Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

Dissertation zur
Erlangung der naturwissenschaftlichen Doktorwürde
(Dr. sc. nat.)
vorgelegt der
Mathematisch-naturwissenschaftlichen Fakultät
der
Universität Zürich
von

Francisco L. Brazile
aus den USA

Begutachtet von
PROF. DR. ROBERT WEIBEL
PROF. DR. KURT BRASSEL

Zürich 2000
Die vorliegende Arbeit wurde von der Mathematisch-naturwissenschaftlichen Fakultät der Universität Zürich auf Antrag von Prof. Dr. Robert Weibel und Prof. Dr. Kurt Brassel als Dissertation angenommen.
Contents

Contents i
List of Tables v
List of Figures vi
Preface x
Abstract xi
Zusammenfassung xiii

1 Introduction and scope 1
  1.1 Motivation ........................................ 1
  1.2 The Goals of this Research .................. 2

2 The Cartographic Process 3
  2.1 Basics ........................................... 3
  2.2 Structure Recognition and Map Design ........ 6
  2.3 Decoupling Structure Recognition and Generalization from Map Design .... 7
    2.3.1 Graphicacy ................................ 9
    2.3.2 Generalization in Map Design ............. 10
    2.3.3 Generalization in Map Creation ........... 10
    2.3.4 Generalization in Map Design Again ....... 10
  2.4 Conclusion .................................... 11

3 Digital Cartography: Procedures and Perspectives 13
  3.1 Introduction .................................. 13
  3.2 Early Digital Representations ................. 13
  3.3 Increasing Demands of Early Digital Systems ...... 15
  3.4 Moves Towards Context-based Generalization .... 17
  3.5 Generalization Operators ..................... 18
  3.6 Multiple Representations ..................... 22
    3.6.1 Early Analysis .......................... 22
    3.6.2 Multiscale Databases .................... 23
    3.6.3 Update Propagation ....................... 23
    3.6.4 MRDB described .......................... 25
  3.7 A Review of Existing Generalization Systems .... 27
    3.7.1 ARC/INFO: with RevPG and FTI ............. 27
### 4 A Framework for an Evaluation-based System

4.1 Introduction .................................................. 32
4.2 Assumptions .................................................. 32
4.3 Framework Overview ........................................... 34
   4.3.1 The Main Processing Stages .............................. 34
   4.3.2 A Basic Example Visualized on a Sample Map Image ... 35
4.4 Detailed Description ........................................... 40
   4.4.1 Initialization Stage – Rules, Map Infrastructure, and Calibration ... 40
   4.4.2 Optimization Stage ....................................... 44
   4.4.3 Final Reporting Stage ................................... 46
4.5 Conclusion .................................................. 47

### 5 Feature-based Spatial Partitioning to Support Evaluation

5.1 Introduction .................................................. 48
5.2 Motivation for Partitioning ................................... 49
5.3 Basic Methodology ............................................. 51
5.4 Detailed Methodology .......................................... 51
5.5 Assumptions about Continuity and Homogeneity ........... 56
5.6 Possible Extensions ........................................... 58
5.7 Conclusion .................................................. 59

### 6 System Design for Integrating Feature-based Partitioning, Measures, and Automated Evaluation

6.1 Introduction .................................................. 60
6.2 Basic Requirements ........................................... 60
6.3 An Object Oriented Design ................................... 61
6.4 Interoperability: Data and Rules Schema ................... 63
   6.5 Data Structure-based Symbolization Problems ............ 66
      6.5.1 Stacking Order ......................................... 67
      6.5.2 Double-lined Road Casing Intersections ............... 68
      6.5.3 A Unique Entity Type Can Require More Than One Symbology ... 68
      6.5.4 Dash Spacing of Separated, Topological Line Segments Produces Poor Cartographic Results ... 68
      6.5.5 The Contiguity Problem of Road Segments ............ 70
      6.5.6 The Overlapping of Casings on Forked Roads .......... 70
      6.5.7 Interoperability of Symbology Instructions .......... 71
      6.5.8 Object Oriented, Java 2, and System Specific Issues .... 71
6.6 Conclusion .................................................. 73
## Testing the Integration of Feature-based Partitioning, Measures, and Automated Evaluation

7.1 Introduction .................................................. 75
7.2 The Availability and Selection of Measures .................... 75
  7.2.1 When to Apply Measures .................................. 75
  7.2.2 What Kinds of Measures Exist ........................... 76
7.3 What Should Be Measured ...................................... 77
  7.3.1 Linear Features and the Network Graph .................. 77
  7.3.2 Partitions .................................................. 78
  7.3.3 Individual Features ....................................... 79
7.4 What should be Evaluated ..................................... 80
7.5 Conclusion .................................................... 83

## Conclusion

8.1 Results ....................................................... 85
8.2 Suggested Improvements ...................................... 86
  8.2.1 Philosophical – The Turing test for Generalization Systems .................. 86
  8.2.2 Strategic Directions ....................................... 87
  8.2.3 Tactical Implementations ................................. 88
8.3 Outlook ....................................................... 90
  8.3.1 Linkages Between Geoprocessing Techniques .............. 90
  8.3.2 Interoperability .......................................... 91
  8.3.3 Conclusion ................................................ 91

## Bibliography

A Glossary ....................................................... 104

B Selected Prototype APIs ....................................... 107
  B.1 License ...................................................... 107
    B.1.1 Copyright ............................................... 107
    B.1.2 List of conditions ....................................... 107
    B.1.3 Disclaimer .............................................. 107
  B.2 Package `ch.unizh.geo.antelope.app` ........................ 107
  B.3 Classes .................................................... 109
    B.3.1 Class `Line_primitives` .................................. 109
    B.3.2 Class `Node` ............................................. 110
    B.3.3 Class `Node_list` ....................................... 111
    B.3.4 Class `Dataset` .......................................... 114
    B.3.5 Class `Cloner` .......................................... 118
    B.3.6 Class `Line_poly_flt` ................................... 119
    B.3.7 Class `Face_list` ......................................... 121
    B.3.8 Class `Partition_list` .................................... 122
    B.3.9 Class `Face` ............................................. 124
    B.3.10 Class `Transform` ........................................ 127
    B.3.11 Class `Project` ........................................... 129
    B.3.12 Class `Line_segment` ..................................... 140
C Prototype XML Data Type Definitions

C.0.13 A sample project file ........................................... 143
C.0.14 A sample data file with points and attributes .................... 144
C.0.15 A sample partition specification file ............................ 149

Index .......................... 151
List of Tables

2.1 The amount of available graphic area, relative to a source map at 1:25,000. ....... 5
3.1 Kilpeläinen’s MRDB Representation Levels ............................................. 26
6.1 The Six Packages of the Antelope generalization prototype. ..................... 61
6.2 The Object Classes in the antelope.app Package. ................................. 62
6.3 Unique road entities found in the Austin East, 1:24,000 SDTS DLG-3 data set after the importation to ARC/INFO. ................................. 67
7.1 Categories of geometric measures ....................................................... 76
7.2 Categories of binary topological relationships in $\mathbb{R}^2$ (source: Kilpeläinen, 1997; Kainz, 1995). .................................................. 77
List of Figures

2.1 An airphoto of Jegenstorf, Switzerland at 1:25,000, cropped from (Swiss Society of Cartography, 1977, fig. 5). ................................................................. 4
2.2 A topographic map of Jegenstorf, Switzerland at 1:25,000 (cropped from, Swiss Society of Cartography, 1977, fig. 8). ................................................. 5
2.3 A topographic map of Jegenstorf, Switzerland at 1:50,000 (cropped from, Swiss Society of Cartography 1977, fig. 7). In this case, only 1/4 of the graphic area of the 1:25,000 map is available to show the same basic features. ............................................. 6
2.4 A topographic map of St. Gallen, Switzerland at 1:25,000 (from Spiess, Swiss Society of Cartography, 1990, fig. 9a). ...................................................... 7
2.5 Selection of streets for 1:200,000 with symbology (inner) casing widths in black. (from Spiess, Swiss Society of Cartography, 1990, fig. 9b). ......................... 8
2.6 The basics of the street network are fixed into place (from Spiess, Swiss Society of Cartography, 1990, fig. 9d). ..................................................... 8
2.7 The remaining settlement structures (left) are added around the street network, seeking to preserve the original feature density and structure, producing (right) the generalized representation (from Spiess, Swiss Society of Cartography, 1990, figs. 9f, 9h). .................................................. 9
2.8 Here (left) a fundamentally correct generalization is hampered by a poor execution of symbology. Right, the same area with better execution of symbology fine-tuning and presentation (source: Baumgartner, 1990). ................................................... 12

3.1 The Lang algorithm. Figures a-c show the first iteration for a complete look-ahead. Figure d depicts the resulting line segments, with eliminated vertices shown in white. (quoted from source: Weibel, 1997a). .................................................. 14
3.2 The Douglas-Peucker algorithm. a) Initial base line with furthest vertex (v10). b) First split into two parts, again with furthest vertices (v4, v14) shown. c) Second split of left part. Vertices v2 and v3 are now within ε, which the second part must be split further at vertex v7. d) Corridors that were eventually generated along the original line. Vertices which were eliminated are shown in white. Note that the final result happens to be the same as for the Lang algorithm for the given line and ε (quoted directly from source: Weibel, 1997a). .................................................. 14
3.3 Hypothetical spatial distribution of lakes that demonstrate a lack of cartographic discrimination in rule-based expert systems. (source: Beard, 1991). ................. 17
3.4 The system, proposing to displace a road (as line segments), would be able to compute the effects on nearby features and propagate displacement vectors (or trigger other operators) as needed (source: Ruas and Plazanet, 1996; Weibel, 1997a). ............... 18
3.5 Omitting example: a. an urban area in New Delhi, India at 1:25,000 (source: Delhi Guide Map, 1:25,000 Third edition, updated 1984, Survey of India). b. the same area at 1:50,000. The shaded buildings from a. were omitted in b. 19

3.6 Displacement example: a. The buildings in Figure 3.5b. are too close together. They can be separated from each other, shown in b., by at least the minimum object separation or legibility distance, which is a value dependent on other visual factors but is usually greater than or equal to 0.3 mm. 20

3.7 Consolidation and Typification example: a. The buildings from Figure 3.6b. create an illegible image when the map scale is reduced to 1:100,000, shown in b. c. demonstrates that buildings can be combined and building clusters typified, which usually involves maintaining the semantics of a cluster by preserving (and enlarging) some buildings in the cluster, though often not the original number of buildings. 20

3.8 Area Simplification example: a. The hypothetical lakes from a, Figure 3.3. b. The lakes are reduced to the new scale and, c. simplified to one lake slightly larger than the minimum polygonal area. 21

3.9 Line Simplification example: a. The Chinese coastline at approximately 1:40 million (source: China and Its Neighbors, Early 1970s, copyright Paul J. Pugliese. In, Kissinger, Henry, 1999. Years of Renewal. New York: Simon and Schuster, p.137). b. The coastline, boundaries, and rivers, at this scale and for this general map theme (shown without labels), can be greatly simplified to avoid the appearance of coalescence. 21

3.10 Data added to a geographic database is propagated to a master database, as well as interacting with methods for each representation that determine which modules need to be cartographically refreshed. The incremental generalization modules only refresh partitions that have been affected by the master database update (from Kilpeläinen, 1994). 24

3.11 A. shows the MRDB at level 3. B. shows the representation at level 3 of a cartographic database. C. shows a cartographic database after generalization operations took place (from Kilpeläinen, 1997). 26

3.12 An illustration of the negotiating network of the agents responsible for features in the dataset (graphic published in Hardy, 1999; originally derived from AGENT internal reports, 1997). In this case, one agent is representing a primary partition (see Chapters 4 and 5) and would be used to communicate evaluation statistics such as density to other partitions agents, who may be determining if the map was generalized consistently. 30

4.1 A high-level state diagram of a generalization machine which incorporates neighboring features context and run-time evaluation. 33

4.2 A sample U.S. Geological Survey topographic map image, Austin East, Texas, originally at 1:24,000, here at 1:100,000. This figure is shown to indicate the general cartographic form of an urban area as published by a North American national mapping agency. 36

4.3 The primary partitioning features are identified from specifications. For this example, they include major roads, major rivers, and map boundaries. The major roads were collapsed to centerlines. For indexing purposes, this is okay. Note the narrow, river-fed lake shown at the bottom is not collapsed. Also, two road segments selected from the high priority feature classes are very close together. The partition formed by them at other bounding features will undergo post-processing. 37
4.4 Additional constraining features are added during this process, such as the determination of urban/rural settlement boundaries. This will be used to characterize the impression of the settlement and help constrain displacement of indexed features.

4.5 All non-linear features are indexed into (source) partitions formed by all linear features forming topology (roads, rivers, railways, etc. but not contours). The zoom-in window highlights two target partitions, shown 1 and 2, which will need to accommodate the buildings indexed to their corresponding source partitions. The target partitions lost graphic area due to symbology changes that occur during generalization.

4.6 The primary partitioning linear features are generalized for the target scale. In this case, a target scale of 1/2 the original, approximately 1:50,000 (shown here 1:200,000) is performed.

5.1 A UML activity diagram describing the basic processing steps taken for partitioning. Note, many details and procedures are encapsulated in each activity box. Solid bars represent synchronization points and asterisks represent multiple process instances.

5.2 The urban focus area (Austin East, TX) at approximately 1:24,000. Most map symbols shown here are similar to USGS specifications, though no labels are present.

5.3 The focus area (centered rectangle) and surrounding features reduced to 1:50,000. The original 1:24,000 symbol widths remain (no generalization).

5.4 Schematic of the focus area (centered rectangle) showing a buffer around the primary road segments and casings. Shaded areas represent source, primary partitions.

5.5 Map of focus area showing 1:24,000 source, secondary partitions (each non-overlapping shaded area). Buildings are shown with their containing partitions.

5.6 Map of focus area showing 1:50,000 secondary, target partition outlines. Two partitions are shown shaded along with the unmodified building features belonging to those partitions. The top shaded partition will be relatively easy to process and generalize, compared to the bottom shaded partition.

5.7 A linear feature, r, is not used for partitioning. a. A partition boundary, b, bisects r. b. r1 was displaced in a direction that violates important visual topology with r2. c. r1 and r2 were smoothed inconsistent with the global shape of the original line.

5.8 Features A, B, C, D would all normally be shown if their allocated partition size were large enough. One design decision is to ignore B and show features with enough space. Another decision, though expensive, is to recreate the partitioning by displacing the river in a deliberate direction in order to mitigate the needs of features A, B, C, D.

6.1 The Project class contains most of the subclasses and structures responsible for delegating GIS, partitioning, and generalization routines. Note that separate geometries are kept for source and target data.

6.2 Centerlines in the data base symbolized as parallel line casings intersect at road junctions, A. The desired impression for these casings is B.

6.3 Postscript image generated from reverse engineered symbology instructions to look like the corresponding, published USGS topographic map. This area is the main campus of the University of Texas at Austin.

6.4 A double-line casing road with no area fill intersects a higher priority double line casing highway with no area fill, redrawn from Map of Europe, 1:5,000,000, Copyright Kümmery+Frey, Bern, Switzerland. Yet the lower class road is shown in higher priority figure ground. At this scale connectivity should be indicated as in Figure 6.2, part B, regardless of road class, as was performed elsewhere in this Map of Europe. Digital systems would, in general, need the ability to handle this case.
6.5 The dashing pattern required by some entities such as narrow highways was interrupted due to the segmented nature of the underlying object. The desired pattern is shown, B. ................................................................. 70
6.6 Contiguity problem stemming from disjoint line segments, A., when a single polyline is expected, B. ................................................................. 71
6.7 Road casings drawn in one shade and then overdrawn in another by line segment. The graphical representation of the roads do not join properly, A. The desired representation is B. ................................................................. 71
6.8 Java Application that can read and write vector topographic data, plot on-screen, create topology, perform feature-based partitioning and do rudimentary generalization and evaluation. In this revision, the left panel shows USGS data symbolized at the source scale on screen at 1:24,000. The right panel shows the data at the target scale of 1:50,000. (Figure reproduction is not to scale.) ........................................ 73

7.1 Examples of safe-regions, from (Peng, 1997). **Left.** The building object in the shaded region can be displaced to another position still enclosed in the shaded area and still maintain neighborhood relationships. **Right.** Similarly, the building to the right has its own allowed region, the shaded area. (Fei, 1996) cautions that the emphasis should be placed on to where the building should be displaced after symbology expansion and not in which area **could** the building be displaced, ignoring symbology expansion. The safe-region measure can still be useful when using a triangulation of symbolized features, addressing this problem. ........................................ 79
7.2 Primary partition for both source and target map. ........................................ 81
7.3 A secondary partition in Laverune, France with 18 indexed features (buildings) and a symbol density of 18%. ................................. 82
7.4 A secondary partition in Asia (New Delhi, India). ................................. 82
Preface

I could not have made any meaningful attempts at studying cartographic generalization without the assistance of many experienced and creative individuals, both directly and indirectly.

I am at first grateful to Prof. Dr. Robert (Rob) Weibel for his strong support in matters scientific, engineering, financial and light-hearted and his unfailing belief in my abilities as a junior researcher. Dr. Geoff Dutton provided the remaining senior instruction needed to carry out my day-to-day research. Because Geoff and I had to share an office, it was hard for him to escape the multitude of my pestering questions. Yet, he somehow managed to get his own work done and consult with me on my own technical problems. My thanks to personnel directly senior to me at the University of Zurich would not be complete without also sharing thanks with Prof. Dr. Kurt Brassel. He too shares that gift of instant understanding of problems technical and strategic to an almost embarrassing degree. His unfailing dedication to providing sufficient financial support for the staff was highly appreciated as well. I also wish to thank the Swiss National Science Foundation (SNF) and the University of Zurich, Department of Geography for the financial support provided to conduct research between June 1996 to June 2000. Along those lines I would also like to thank Heidi Preisig who as administrative assistant to our research group performed miracles to insure that the funding, travel, health matters, library books and journals, and post were all available when needed.

I would also like to thank all of my senior colleagues with whom countless discussions concerning generalization and whose advice proved immeasurable. Prof. Dr. William Mackaness is delightful to work with. Likewise, Prof. Dr. Marc van Kreveld continuously provides a precise clarity of understanding algorithmic issues in vector-based mapping and generalization. Dr. Jean-Phillipe Lagrange shared his views of generalization on numerous occasions.

And to my senior peers and other trench fighters who lead the day-to-day crusade to solve generalization and other technical GIS issues. I personally have learned directly from: Dr. Nicholas Regnaud, Dr. Anne Ruas, Mats Bader, Alistair Edwardes, Dr. Mathieu Barrault, Dr. Christof Baejjs, Prof. Dr. Francis Harvey, Sébastien Mustière, and Regula Ehríholzer.

I also wish to thank people who have offered useful advice on general issues concerning GIS, generalization, remote sensing, interoperability and data dissemination including Dr. Britta Allgöwer, Dr. Caroline Westort, Daria Martinoni, Dr. Andrej Vckovski, Dr. Felix Bucher, Dr. Eva-Maria Stephan-Lippert, Audi Bachmann, Dr. Ivo Leiss, Dr. Michael Schaepman, and Dane Ohe.

I lastly wish to thank my family, Prof. Dr. Robert P. Brazile, Robert E. Brazile, and P. Jason Brazile, who have provided a continued level of support, in many realms, but here most relevantly in a continuing lifetime of exposure to solving computer-based problems.

-Zürich, Switzerland
July 22, 2001

-F.
Abstract

This dissertation deals fundamentally with analyzing the geographic, communicative, and technical processes involved in generalizing map data for cartographic presentation. Due to the many processes involved, several aspects of generalization are reviewed. The first aspects are those related to the manual tradition of cartography and explore the ways that generalization has been formalized. Traditional cartography and generalization have been thought of as a combination of science and art, and so this study seeks to showcase the boundary between the two. Those aspects which are scientific are then illustrated further through principles of modeling the cartographic process. Aspects of structure recognition and measuring are described in detail, to indicate operational procedures that have been used by cartographers, as well as to indicate the direction for digital systems. Transformations that are applied during generalization, the so-called generalization operators, are also reviewed. These operators seek to modularize tactical aspects of the generalization process, form the foundation to migrate towards fully automated solutions through human guidance, and in some instances can be applied programmatically. Ultimately, however there are shortfalls in the structuring of the entire generalization process.

These shortfalls were addressed by recent research efforts to model the entire process, aiming to create holistic generalization solutions. The pursuit of a holistic generalization framework which could be implemented, at least in part, became the aim of study to be presented in this dissertation. A framework was designed that could model the entire strategic aspects of generalization and demonstrate how procedure structuring and ordering could support an automated solution. The framework was also meant to address the problem of providing a common basis for comparison of differing automated generalization approaches. This framework required the analysis of the manual process, the automated methods that were available, and the direction research was taking these methods. One fundamental element to this framework required the development of a system of organizing and limiting the amount of computing that would be required to achieve reasonable solutions. This element, partitioning the data by topological methods, required further expansion to demonstrate integration and to show the dependencies such a framework would have on a way to constrain solutions.

Through partitioning the data in the proposed manner, strategic level control can be achieved through the generalization process. This control is illustrated via theoretical propositions. Partially in order to demonstrate the feasibility of such a partitioning scheme and also to achieve an understanding of what is needed to pursue a holistic treatment of geographic map data through basic research, a software system was designed which could implement the partitioning system, as well as basic features of common geographic information systems, such as topology and statistical measures. The results of the implementation show that partitioning as described is feasible and illustrates how partitioning and basic evaluation measures can be integrated. The success of the basic concepts have been reinforced through adoption by a separate, but ongoing applied research project, AGENT, described in the body of the report.

Lastly, a philosophical approach towards generalization is discussed that highlights the need to continue to formalize the implicit knowledge held by cartographers, as well as create a system that
can be modeled in a theoretical way, as to provide rigor to implemented solutions for generalization. A final observation of the nature of data acquisition and updating proposed that future research should also address the entire geographic data handling process as modern systems will no doubt rely on the successful integration of such techniques.
Zusammenfassung


Zuletzt wird ein philosophischer Zugang zur Generalisierung gesucht. Eine solche Betrachtung
betont die Notwendigkeit, das implizite Wissen der Kartographen weiter zu formalisieren, um ein theoretisch begründetes System schaffen zu können. Eine abschliessende Analyse der Datenerfassung und -nachführung kommt zum Ergebnis, dass künftige Forschung nicht nur einzelne Schritte, sondern den ganzen Prozess der geographischen Dantenverarbeitung berücksichtigen muss, da moderne Systeme ohne Zweifel auf der erfolgreichen Integration verschiedener solcher Techniken beruhen.
Chapter 1

Introduction and scope

1.1 Motivation

Generalization is a complex research issue that has existed as a design problem to some degree since the invention of maps. It seems a broad, comprehensive and automated solution continues to remain elusive. Though given assumptions about the state of current computer technology and the directions researchers are taking cartographic generalization into the near future, in what ways could or should automated generalization procedures be useful to today’s producers of maps? To find the answer to this question, those in the research community are exploring and refining frameworks and methods for solving cartographic generalization issues. In the process, researchers have cited the need for consistent and reasonable methods for evaluating generalization implementations and the quality of their results (Mackaness et al., 1995).

A perfect quality evaluation methodology would allow the high level issues concerning cartography and map design of both the automated and manual results to be directly compared. It would also allow low level procedures to be compared and evaluated, regardless of implementation. In pursuit of meeting these needs, the main goals of this dissertation are to establish a quality assessment framework and outline some helpful methods. In order to get there, much first has to be said about the philosophy and infrastructure that would be needed to compare common types of maps. Very simple cases of combined map elements and procedures are evaluated or theories from the literature brought in. However, the complexity of adding additional map elements often increases exponentially, and a question of scalability arises.

To combat this increase in complexity, a framework is proposed in this dissertation that can reduce the generalization process as inferred from average generalization cases to a few very specific and well-defined cases. These average cases can best be described as what most of the people want most of the time for a specific map product: urban topographic maps at a scale of 1:25,000 and scales derived from 1:25,000, that will serve as base maps for other map uses including thematic mapping. An analysis of how well these average cases can be inferred from the well-defined cases is presented.

To find specific methods for assessing quality in generalization, common elements and procedures of digital cartography were analyzed and evaluated, independent of their participatory role in the larger body of a generalization framework.

Some strategies for solving specialized generalization cases are given. These are essentially variable and cannot be known or well-defined ahead of time. The ability to break these variable generalization cases into their component well-defined or average case parts is explored. It is important to note, however, that the evaluation and assessment of cartographic design is a subjective process.

Fortunately, some degree of this design subjectiveness is shared by a large population of current
map authors and readers, if inferred by assessing which maps in recent history have traditionally been considered successful. Success can be measured in both the economic terms of map sales, in the psychological terms of human testing, and in the acknowledgment of how much aesthetics are affordable given the resources allocated for a map’s production.

Incorporating all such successful considerations is non-trivial and can be a source of debate in the part of this dissertation that describes ways to differentiate map aspects or qualities (qualitative measures) and goals for algorithmic output. Similarly, even the language and definitions used to describe generalization events and procedures suffer (and benefit) from variety. Here, too, the proposal and limited use of new terms are given in hopes of providing some rigor and clarity to the pursuit of insight in generalization quality.

1.2 The Goals of this Research

Delineating broad cartographic design procedures into non-overlapping tasks is subjective and sometimes seems arbitrary. Yet in doing so, it is possible to create a perspective that simplifies the engineering task of automating and evaluating cartographic generalization, or generalization, and focuses efforts on clearly stated problems for automation. The basic goal of this dissertation is to do exactly that kind of delineation, namely:

- to assess the potential for automated context-based generalization;
- to propose a strategy for context-based generalization that relies on self-evaluation;
- to determine how to construct the evaluation infrastructure, via prioritized information partitioning and measures;
- to perform implementation studies and interpret other empirical work that gauges the success and scalability of the proposed strategies, some of which have been adopted and integrated by other research and commercial efforts.

In order to establish supportive reasoning for the proposed strategies and methods, it is first necessary to present some basic definitions, assumptions, and a summary of the background and state of the art of generalization up till the time of the proposed methodology.
Chapter 2

The Cartographic Process

2.1 Basics

Maps are created primarily to demonstrate and emphasize spatial relationships. The act of cartography is the art and science of creating maps. “Cartography” and “maps” themselves have been newly defined by the International Cartographic Association (ICA) in 1997. Cartography is the discipline dealing with the conception, production, dissemination and study of maps.

A map is a symbolised image of geographical reality representing selected features or characteristics, resulting from the creative effort of its author’s execution of choices, and is designed for use when spatial relationships are of primary relevance.

Even though these are new definitions, they still reveal a close relationship to the traditional view of map design as a personal, subjective process and leave the construction procedures and dissemination media un-specified.

The 1973 ICA definition for generalization remains unchanged.

Selection and simplified representation of detail appropriate to the scale and/or the purpose of the map (International Cartographic Association, 1973, 173).

‘Digital generalization’ has been defined in (McMaster & Shea, 1992, pg. 3-4) and also emphasizes map purpose and intended audience, implicitly outlining quality as fitness for use.

Digital generalization can be defined as the process of deriving, from a data source, a symbolically or digitally-encoded cartographic data set through the application of spatial and attribute transformations. Objectives of this derivation are: to reduce in scope the amount, type, and cartographic portrayal of the mapped or encoded data consistent with the chosen map purpose and intended audience; and to maintain clarity of presentation at the target scale.

Topographic base maps for medium-range scales (approximately 1:25,000 to 1:250,000), the focus of this study, are often derived from aerial photographs taken from airplanes. These aerial photographs, or airphotos, are used in the map generation process in order to determine information (Swiss Society of Cartography, 1977) such as

• Where are the watercourses?
Figure 2.1: An airphoto of Jegenstorf, Switzerland at 1:25,000, cropped from (Swiss Society of Cartography, 1977, fig. 5).

- Where are the main roads?
- Where are the railroads?
- Where are the paths through the forest?
- Where is the highest point?
- How many houses are in the settlement?

and are generally absent of lettering. Figure 2.1 shows an example airphoto of the village of Jegenstorf in Switzerland, approximately 1:25,000.

The topographic mapping process seeks to extract the most generally useful information, suppressing the unimportant. Details which are not readily visible in the airphoto are emphasized (Swiss Society of Cartography, 1977). Thus, a topographic map generally contains

- Roads, railways, and other transportation classifications
- Urban settlements
- Contour lines
- Rock outcrops and important surficial geologic features
- Lettering and height values
and is already a kind of generalization of the information shown in the source photographs. Figure 2.2 shows an example of a black and white topographic map of Jegenstorf, Switzerland at a scale of 1:25,000. In practice, the source airphotos are often at a larger scale than the desired topographic map. For example, the scale of the photos would be from approximately 1:10,000 to 1:15,000 for a 1:25,000 sheet.

For some purposes, such as regional planning, environmental studies, and military applications, topographic maps at scales smaller than 1:25,000 are needed. To create a topographic map at half the scale, at 1:50,000, presents a challenge to cartographers because at this scale change, already $3/4$ of the original graphic area is lost. This value can be derived from these simple formulae:

\[
\text{scale factor} = \frac{\text{target scale denom.}}{\text{source scale denom.}} \quad (2.1)
\]

\[
\text{available graphic area} = \frac{1}{(\text{scale factor}^2)} \quad (2.2)
\]

Figure 2.3 shows an example of a generalization of Jegenstorf to a scale of 1:50,000. Scale then plays a large role in determining the amount of information that can be presented in the final map image.

There are a number of procedures that trained cartographers can apply intuitively in order to show map features at successively smaller scales. Taking many variables into account, such as purpose,
Figure 2.3: A topographic map of Jegenstorf, Switzerland at 1:50,000 (cropped from, Swiss Society of Cartography 1977, fig. 7). In this case, only 1/4 of the graphic area of the 1:25,000 map is available to show the same basic features.

legibility, the color and contrast of the map symbology and special lighting conditions, the scale change, the presentation media, and reader expectations, these procedures help to include as many of the same basic features shown at the target scale as practical. These procedures are described in the next chapter in Section 3.3, after a brief introduction to the evolution of digital cartography and generalization.

2.2 Structure Recognition and Map Design

Structure recognition (Brassel & Weibel, 1988; McMaster & Shea, 1992) is the process of determining what to generalize and why, and when to generalize it. Brassel, Weibel, and Dutton (Brassel & Weibel, 1988; Weibel, 1997a; Weibel & Dutton, 1999) recommend integrating structure recognition into an overall framework to determine the generalization objectives and map controls (cf. Section 4.4.1 for a list of map controls). McMaster and Shea (McMaster & Shea, 1992) use structure recognition, an aspect of what they call cartometric evaluation, to identify the typical geometric conditions that would trigger a generalization procedure,

- congestion, the crowding of features
- coalescence, when features seem to fuse together
- conflict, when feature symbology overlaps
- complication, as aspect of special-cases which are not easy to identify apriori
- inconsistency, when features are wrongly transformed differently under similar cartographic conditions
imperceptibility, when features become too small to be visible.

They further point out the subjective nature of, not only the definitions of these conditions, but the varying environment in which these conditions can occur. They ask whether formal definitions of these conditions be defined that can be applied universally and what kinds of measures can be used to detect them. Typically, a trained cartographer would perform procedures to avoid causing any of the above conditions to occur. Speiss demonstrates an implicit, manual process that skillfully combines structure recognition, evaluation, design and execution in just a few steps (Spiess, 1990).

The process begins with a topographic map, Figure 2.4, that will be used to create a 1:100,000 representation, as well as a 1:200,000 representation. A selection of the streets for 1:200,000 is superimposed on the current map with the inner road casing already at the target symbology width, Figure 2.5. This base structure is then in turn used to identify the graphic aspects of the road network, Figure 2.6. Given the implicit constraints of the road network, the cartographer could then complete the rest of the image by adding buildings and features that are representative of the original settlements, Figure 2.7.

2.3 Decoupling Structure Recognition and Generalization from Map Design

This section comes in response to a question often heard, at least in the mid 1990s, in practitioners’ circles. The question is, are researchers trying to fully automate map design? The answer today is not necessarily.

During basic cartographic map design and creation, cartographic generalization often occurs, but not always. Map design can mostly be considered a larger, parent process that employs cartographic generalization under certain conditions. The case where map generalization occurs without map design is called database or statistical generalization cf. (Weibel & Dutton, 1999) and will not be treated in this dissertation. Because those conditions where cartographic generalization is needed occur so often, some practicing cartographers feel generalization is just a part of the map design process and should not be thought of separately in the context of automating map production.

However, delineations between map design and cartographic generalization are indeed present and suggest a separate treatment for the two subjects. The separate treatment arises because cartographic...
Figure 2.5: Selection of streets for 1:200,000 with symbology (inner) casing widths in black. (from Spiess, Swiss Society of Cartography, 1990, fig. 9b).

Figure 2.6: The basics of the street network are fixed into place (from Spiess, Swiss Society of Cartography, 1990, fig. 9d).
generalization may lend itself more easily than the general principles of map design to automation efforts, at least initially. This intuition is based on the concept that some, perhaps many, cases for generalization, especially in the creation of topographic maps, can be well-defined. However, total automated solutions to either generalization or map design are informally considered optimization problems and are NP-complete. So, following the trail of generalization as an optimization problem (Cromley & Campbell, 1992; Ware & Jones, 1998; Lonergan et al., 1999), finding distinctions that ease computability is especially warranted. But first, an alternative explanation of the final product, the map, is presented.

A cartographic map is a graphic image that employs symbols and light variations to denote variations on, over, or under a finite and determinant surface space. A cartographic map is useful to a human map reader when the size of the graphic image is smaller than the size of the determinant surface space and contains graphic elements which communicate only those variations which a map reader can distinguish and deem important.

Useful maps often are used to communicate to a map reader the position of ordinary, physical variations such as roads, rivers, buildings and forests vertically near a surface space, usually on a land surface or under a water surface. Useful maps may also be used to communicate to a map reader the position of virtual variations vertically near a surface space. These variations are still quantifiable but less tangible or positionally certain, such as bird migration routes, political boundaries, or thematic statistics. Variations that are used on the map are called map features. Cartographic map design then is the act and process of determining which of these variations are required to be represented in a cartographic map and how to graphically represent those variations in order to satisfy a well-defined map reader purpose. This distinction between physical and virtual variations aids slightly the composition selection process, as reported later. The map designer will follow rules which govern human ergonomics as applied to map interpretation. These rules are part of the language of graphics, human perception, and human psychology and are sometimes called graphicacy.

### 2.3.1 Graphicacy

Determining how to graphically represent variations necessarily requires thorough knowledge and experience with graphicacy, or the use of form and graphics as the primary medium for communicating information to human readers (Balchin, 1976). Cartographic maps are thought to be the most
information-dense graphics produced for human readers (Bertin, 1967). These information-dense graphics exercise and tax the skills of human map readers the greatest and are hence one of the most challenging classes of graphics to engineer.

If a map is well designed, it is likely the reader will infer the intended map meaning and so the map successfully fulfilled the intended map purpose. Optimizing variation selection and graphicacy expectations, often called visual variables (Bertin, 1967), for creating a successful cartographic map requires a human map designer. However, there are many aspects to general map design and creation principles and tasks that can be facilitated with computer hardware, software, and cartographic intelligence formalizations and algorithms. One such aspect that needs to be facilitated, and possibly automated to a great degree, is cartographic generalization.

2.3.2 Generalization in Map Design

Cartographic generalization during map design occurs when the variation representation size is selected given at least a constraint of map scale. Map scale is the ratio of the size of a graphic image of a cartographic map compared to the size of the represented finite and determinant land surface. Sometimes, map scale limitations of map graphic area, intuitively thought of as a graphic real-estate limitation, cause a map designer to use a graphic representation on the map which may not resemble the variation’s original shape, if tangible as for physical variations. An example of this substitution for physical variations could range from making square a rectangular building symbolization to the use of a locally-defined graphic icon, like a flag icon, to represent a government building. For virtual variations, a dashed line may represent a political boundary or a stack of dollar signs in the map graphic may represent thematic data, such as a country’s gross national product (GNP). The procedures available to rectify generalization problems have been traditionally grouped into logical units, described in Section 3.5. The remaining selection of visual variables, like symbols, colors, and line weights, that best communicate the map information is part of the broader act of map design. When those symbols or map graphic objects are illegible due to overlap or too close a proximity for a human map reader to distinguish, cartographic generalization must be employed to help resolve this kind of real estate problem.

2.3.3 Generalization in Map Creation

Cartographic generalization during map creation occurs when the amount of detail for a variation’s shape must be determined given map purpose requirements for resolution and the communication effects and interplay of variation resolution on other map elements, which is not the same as the real-estate problem. Sometimes the variation’s original shape can be preserved, but doing so causes interference in the communication process. When excessive detail causes illegibility or distracts from the map purpose, a characterization of the original variation is used. A common example of this characterization is the subway map, which is topologically correct, but not geometrically correct. The distinction between generalization as viewed from the map design process versus the map creation is not extremely important but can be useful when there is a narrow range of display scales considered for the final map. It is possible that changes would be needed in the amount of resolution shown (map creation) before they would be needed in the amount of real-estate (map design) problematic map features consume, especially for on-screen representations.

2.3.4 Generalization in Map Design Again

Sometimes the amount of variation detail present in the map is basically correct, though the final result is poor. This deficiency is probably a result of a poor execution of the basic, desired symbology, fine-tuned for the specific needs or constraints of the map. Baumgartner, (Baumgartner,
1990), graphically summarizes this concept through examples, Figure 2.8. In Figure 2.8, left, misin-
interpretations and misunderstandings can arise, in part because the weighting of the linear network
structures are not consistent. Figure 2.8, right, streamlines some of the overly complicated symbol-
ology artifacts and presents a better overall map. Baumgartner concludes that the graphical design of
the final product is thus also an aspect of generalization, though the opposite is argued here. Gen-
eralization is an aspect of graphical (map) design and is needed sometimes when a map is designed
and sometimes when a map is created or executed. Though, map design should always result in an
aesthetically acceptable map, or more properly, a map optimized for communication effectiveness.
Following this argumentation, it seems reasonable to automate tasks of map generalization which are
more mechanical in nature and, in fact, separate from map design and its susceptibility to intangible
computational formalization.

2.4 Conclusion

The topographic map design process traditionally begins with the acquisition of imagery via airpho-
tos, electronic sensors, or through surveying. The compilation of these materials in geographically
overlapping pairs allows the production of a land model indicating basic elevation. Features on the
ground surface are also identified and extracted from the imagery. These features range from natural
features such as rivers, lakes and forest tracks, to man-made features such as roadways and buildings.
At that point, an inherent inventory exists which contains the geographic elements that could be
mapped. This inventory is not necessarily in digital form as features can be mapped directly to
analog media such as mylar. The next step is for a cartographer to answer the essential questions of
map purpose by designing a map which contains all the necessary features in a manner which can
be understood by a map reader. Using graphics as a language for communication has been called
graphicacy and is distinct from other kinds of communication such as words or numerical reasoning.
At first steps, cartographic representation can be a basic task. If the map is of a sufficiently large
scale, the outline of all included features can be represented directly on the map. As the map scale
decreases, a congestion of feature symbology tends to render a small scale map illegible. At that
point, generalization is needed to adjust the feature symbology, position, or selection of features on
the map. Structure recognition is a term to describe the process of identifying how the essential
features interact and what cartographic conflicts can occur. It is argued in this chapter that the
procedural aspects of map design and specific actions of map generalization can be separated and
are not necessarily the same process, though cartographers tend to combine both steps. Thus, even
though map design currently requires human intervention to answer essential questions of purpose,
map generalization tasks can begin to be automated by following basic structure recognition or eval-
uation tasks and triggering cartographically intelligent responses. Major aspects of the evolution of
automated generalization are described next.
Figure 2.8: Here (left) a fundamentally correct generalization is hampered by a poor execution of symbology. Right, the same area with better execution of symbology fine-tuning and presentation (source: Baumgartner, 1990).
Chapter 3

Digital Cartography: Procedures and Perspectives

3.1 Introduction

Digital and numerical generalization has been a research topic receiving wide-spread attention for almost 40 years. Many strategies for automating generalization have been proposed that successively increase the understanding, scope, and digital application of cartographic knowledge. In tandem, there have also been continuous debates about the potential limits for automating generalization. Some people quietly stress that the task is too intricate and computers can only assist humans in building meaningful maps. Wherever the limit to automation lies, it has not been reached and so work has continued at a regular pace. This chapter reviews some of the traditional perspectives toward automated generalization and then presents some additional perspectives on the cartographic process.

3.2 Early Digital Representations

One obstacle to full automation of generalization lies in how the problem had been formalized and modeled with technology. Early formalizations and rules reflected traditional human-oriented procedures and perceptions which were far too sophisticated for the computer technology available at that time. It was clear that to make progress in digital generalization, compromises favoring existing technology would have to be made.

For example, Rhind in (Rhind, 1973) argued for the redefinition of cartographic procedures to be as efficient and narrow in scope as possible to accommodate the then available technology. In about the same period, algorithms to accomplish specific, narrow goals, such as line smoothing or point reduction of lines appeared in cartographic literature, such as (Perkal, 1966), (Lang, 1969) and (Douglas & Peucker, 1973). Figure 3.1 refers to the Lang algorithm which filters out data points based on an epsilon tolerance level and using a look-ahead of five points. Figure 3.2 schematizes the Douglas-Peucker algorithm which also uses a tolerance band measured perpendicularly from a base line comprised of a sub-segment of the original line. Similarly, first steps at mathematical and structural formalization were described, as in (Töpfer, 1974) and (Morrison, 1974). To many cartographic artisans, the sum total of these compromises yielded results too primitive or economic costs too high to accept for production systems and so cartographic practice continued on an analogue track.
Figure 3.1: The Lang algorithm. Figures a-c show the first iteration for a complete look-ahead. Figure d depicts the resulting line segments, with eliminated vertices shown in white. (quoted from source: Weibel, 1997a).

Figure 3.2: The Douglas-Peucker algorithm. a) Initial base line with furthest vertex (v10). b) First split into two parts, again with furthest vertices (v4, v14) shown. c) Second split of left part. Vertices v2 and v3 are now within $\epsilon$, which the second part must be split further at vertex v7. d) Corridors that were eventually generated along the original line. Vertices which were eliminated are shown in white. Note that the final result happens to be the same as for the Lang algorithm for the given line and $\epsilon$ (quoted directly from source: Weibel, 1997a).
However, some mapping agencies responsible for very large land masses determined that these compromises made an acceptable starting place and so they started to amass vast amounts of data stored in digital format (Morrison, 1999a). These collection efforts triggered research in improving digital cartographic systems. For example, Morrison in (Morrison, 1999b) points out that cartographic journals did not exist before 1964 in the U.S., Canada or the U.K. and were created to help fill this need. Eventually, cartographers were able to replace analogue tools with digital tools and shift to digital systems during the 1970s and 1980s. Today, it is fair to say that digital cartographic systems are the primary tool for production cartographers in developed nations and many developing nations, with a few exceptions.

3.3 Increasing Demands of Early Digital Systems

As technology increased, so too did expectations of generalization automation. Once digital systems started to replace analogue methods, expectations were placed on software engineers and cartographers to automate as many tasks associated with generalization as possible. Besides the various software needed to compute, store, and plot cartographic information, various artificial intelligence (AI) techniques such as rule-based expert systems, neural networks, and genetic algorithms were cited as possible enabling technologies for generalization. Peuquet discussed techniques for using AI techniques on large geographical data bases (Peuquet, 1983). Zoraster et al. describe a feasibility study and literature review which preceded the development of a classified system to automate much of the manual production performed by the U.S. Defense Department at the time (Zoraster et al., 1984). Robinson and Jackson presented a paper on using an expert system for map design (Robinson & Jackson, 1985). Nickerson and Freeman presented a paper on a rule-based system to generalize USGS Digital Line Graph (DLG) vector data (Nickerson & Freeman, 1986). Mackaness and others suggested that a map design expert system could be designed by decomposing steps that included generalization algorithms, symbology assignment, feature placement, and evaluation (Mackaness et al., 1986). Monmonier presented an expert system that tracked features and detected symbol overlap and used skeletonization (Monmonier, 1986). McMaster, Shea, and Keller proposed the use of neural networks in generalization (McMaster & Shea, 1992) and (Keller, 1995).

In the short-term, these AI-based designs have not yet proven to be sufficient solely on their own to provide a comprehensive solution to the generalization problem, agreeing with the forecast in (Brassel, 1984). Each would require a significant level of problem specification, constraints, and evaluation mechanisms to work (McMaster & Shea, 1992). The reason for this can be deduced from the following explanations. Mark in (Mark, 1991) demonstrates a typical directive issued to cartographers concerning roads for the generation of a topographic map series:

- Private roads, access roads, and driveways less than 500 feet (152.4m) in length will not be shown unless of landmark value in areas of sparse culture (USGS Topographic Division, 1964).

- All streets in populated places will be shown regardless of length (USGS Topographic Division, 1964).

Which he translates into a hypothetical rule language as:
Where a feature with an importance attribute of 0 will never be shown, while 1 will always be shown. As far as database generalization is concerned, these rules provide a sufficient mechanism to determine which features to select from a database for the final image. However, due to the cartographic conflicts that can arise, the total required generalization effort does not rest solely on selection, as featured in an example in (Beard, 1991a). She cites the following rule:

If a lake or pond would have an area less than the minimum polygon area, do not include it unless there are five or more other small lakes within a radius of 10mm, in which case replace these small lakes with one small lake slightly larger than the minimum polygon area (Nickerson & Freeman, 1986).

Admittedly, this is a complicated rule with many ambiguities encapsulated in the proposed solution. However, she highlights the lack of cartographic discrimination this rule would have in effect on a series of hypothetical lakes, redrawn in Figure 3.3. She points out,

(Figure 3.3) shows a distribution of five lakes exhibiting different configurations of size, shape, and spacing and presumably other descriptive characteristics such as depth or turbidity. In case (a), the stated rule will cause the lake to be replaced by one lake exceeding the minimum area threshold. Case (b) despite the different size and spacing configuration, generates the same result: the five lakes are replaced by one lake. Case (c) shows a similar size and space configuration to (b) but the absence of a fifth lake will cause no lakes to be shown. Case (d) illustrates potential problems in defining the appropriate neighborhood for consideration. In this case, a fifth lake falls just outside the 100 mm radius specified by the rule and thus causes none of the lakes to be shown.

It is argued here that the single most important factor needed to be able to use AI techniques to automate the generalization process is to be able to determine and define a cartographically acceptable notion of structure, or context, a long-term ambition, and to integrate context-based operators with non-contextual operations.

A preliminary distinction between context-independent (independent) and context-dependent (context-based) generalization has been summarized in (Weibel, 1997a), and originates from work described in the next section. An acceptable notion of context is one that includes awareness of nearby cartographic features, such that when changes are considered to one feature, the impact on
nearby features is known to the system. Displacement is often cited as an example of a specific context-based operation. Figure 3.4 shows a case where, after road displacement, the side-effects to nearby buildings are known to the system, which responds intelligently (in this case, with a propagation of displacement vectors). Context-independent operations are generally insensitive to conflicts occurring in neighboring features that could arise after application.

Strategies for automating generalization are now beginning to converge on context-based systems, though the minimum notion of context is still open for debate. Richer contextual evaluation is thought to offer more sophistication in handling problems, but to also increase the solution complexity.

### 3.4 Moves Towards Context-based Generalization

It is difficult to pinpoint the exact origins of the distinction between what is now often called “contextual” and “independent” approaches in the literature. Establishing topological data structures to represent the cartographic information was an important step toward contextual systems. The DIME file, an early topological data structure, was developed by the U.S. Bureau of the Census for storing census block information (Cooke & Maxfield, 1967; Census Bureau, 1969; Peuquet, 1984; Cromley, 1992) and provided adjacency information between points and areas. Various improvements and implementations since then have been developed (Peucker & Chrisman, 1975; Brassel, 1978; Corbett, 1979; Cromley & Campbell, 1984; Morehouse, 1985).

Essentially, topology in modern cartographic systems can offer adjacency information between nodes (0-cells, where a cell is a connected subset), arcs (1-cells), and faces (2-cells). With this information, neighboring features are known to the system and hence algorithms can be developed that preserve topological relationships during generalization, when possible. Topological information can also aid intelligent geometric operations.

Dettori and Puppo in (Dettori & Puppo, 1997) give a framework for defining which specific generalization operators imply topological changes and which merely metric changes. Although this paper presents an interesting and promising framework, there are some debatable issues. For example, to limit the context of simplification algorithms to $\epsilon$ homotopy, which although valid from
Figure 3.4: The system, proposing to displace a road (as line segments), would be able to compute the effects on nearby features and propagate displacement vectors (or trigger other operators) as needed (source: Ruas and Plazanet, 1996; Weibel, 1997a).

a narrow view of scope (and context), implies that these algorithms should not take into account other features. In fact, during simplification, lines can self-intersect or intersect other features, altering topology. Topologically correct versions of simplification algorithms have been presented in the literature (Saalfeld, 1999) to address elements of this problem. The usefulness of topology to automated generalization is explained in more detail in Chapter 5.

Meanwhile also during the 1970s, triangular element data structures, which form triangulated irregular networks (TIN), were proposed for representing terrain data (Gold et al., 1977; Peucker, 1978; Peucker et al., 1978). A TIN model contains data points and their corresponding elevation. TINs allow a constant time operation to determine elevation for any point and a constant time operation for finding adjacent triangles for any given triangle (van Kreveld, 1997).

This data model helped to import expertise on Delaunay triangulation and constrained Delaunay triangulation to the computer mapping community. Uses of Delaunay triangulation and its dual, the Voronoi model, have since been used to compute proximity information in between cartographic features during generalization processing and for managing topology and time-based updates in general (Okabe et al., 1992; Gold & Edwards, 1992; Pilouk & Tempfli, 1993; Gold, 1994; Jones et al., 1995; Ruas, 1995; Ware & Jones, 1997; Peng, 1997; Jones & Ware, 1998; Ruas, 1998b). Other alternative approaches to implementing contextual operators using the raster data model, raster-vector hybrid models, raster-based morphological operators, and the Voronoi have been proposed (Li, 1994; Su & Li, 1995; Li & Su, 1996; Su et al., 1997; Li et al., 1999).

### 3.5 Generalization Operators

There are a number of procedures that a trained cartographer would consider when creating a map after reducing scale relative to an original source map. These manual procedures have been divided into *operators* with the goal of rigorously defining the processes involved (Robinson & Sale, 1969;
Figure 3.5: Omitting example: a. an urban area in New Delhi, India at 1:25,000 (source: Delhi Guide Map, 1:25,000 Third edition, updated 1984, Survey of India). b. the same area at 1:50,000. The shaded buildings from a. were omitted in b.

Hake, 1975; Brassel, 1984; Buttenfield, 1985; Nickerson & Freeman, 1986; Beard, 1987; DeLucia & Black, 1987; Shea & McMaster, 1989; McMaster & Monmonier, 1989; McMaster & Shea, 1992; Rieger & Coulson, 1993; Ruas & Lagrange, 1995; Plazanet, 1996; Weibel & Dutton, 1999). These operators are being continuously refined and formalized, but stem from some basic cartographic ideas. One can consider that:

- Objects can be omitted (cf. Figure 3.5).
- Objects can be displaced (cf. Figure 3.6).
- Objects can be consolidated and typified (cf. Figure 3.7).
- Areas can be simplified (cf. Figure 3.8).
- Lines can be simplified (cf. Figure 3.9).
- Detail can be emphasized.
- Symbology can change.

However, while performing any of these procedures, either by hand or through algorithmically driven generalization operators, there are many types of constraints that should be considered. Although many of these types of constraints have been enumerated, there are still cartographic situations that have not been foreseen and corresponding rules or guidelines stipulated.
Figure 3.6: Displacement example: \textbf{a.} The buildings in Figure 3.5b. are too close together. They can be separated from each other, shown in \textbf{b.}, by at least the minimum object separation or legibility distance, which is a value dependent on other visual factors but is usually greater than or equal to 0.3 mm.

Figure 3.7: Consolidation and Typification example: \textbf{a.} The buildings from Figure 3.6b. create an illegible image when the map scale is reduced to 1:100,000, shown in \textbf{b.} \textbf{c.} demonstrates that buildings can be combined and building clusters \textit{typified}, which usually involves maintaining the semantics of a cluster by preserving (and enlarging) some buildings in the cluster, though often not the original number of buildings.
Figure 3.8: Area Simplification example: a. The hypothetical lakes from a, Figure 3.3. b. The lakes are reduced to the new scale and, c. simplified to one lake slightly larger than the minimum polygonal area.

Figure 3.9: Line Simplification example: a. The Chinese coastline at approximately 1:40 million (source: China and Its Neighbors, Early 1970s, copyright Paul J. Pugliese. In, Kissinger, Henry, 1999. Years of Renewal. New York: Simon and Schuster, p.137). b. The coastline, boundaries, and rivers, at this scale and for this general map theme (shown without labels), can be greatly simplified to avoid the appearance of coalescence.
3.6 Multiple Representations

An unfortunate aspect of digital cartographic production is that map series produced for a certain scale often are created independently of existing map series at larger scales. Allowing this redundancy in production introduces discrepancies across map sheets covering the same area and duplicates effort. Regardless, this route is often taken in part due to a lack of technology to share a single database for multiple scales. “Since problems in updating digital geographical databases have become a major impediment to the effective use of a geographic data production environment, multiple representation databases have become one of today’s key research topics (Müller, 1991)”. Although multiple representation may refer to different facets of creating a unique representation for a particular geographic area, such as the same area at different times or the same area at different scales, this review focuses on multiple representation as an intermediate or separate technological step towards integrating the map generation process. In this respect, there is major interest to define the lacking functionality to support a single, master database that can be used to create maps at various scales and perhaps even on the fly or in a time acceptable to someone submitting a request and waiting for a result.

3.6.1 Early Analysis

Multiple representation initiatives began in response to needs from private mapping vendors and national mapping agencies to limit the redundancy inherent in producing separate map products of overlapping geographic areas (Buttenfield, 1989). At the time, many problems were cited with geographic data storage, retrieval, representation, updating, and dissemination and were discussed together. Major problems such as error propagation models, metadata and lineage, interoperability issues, database and cartographic generalization, and computability issues were all discussed under the research theme. Eventually, each of these issues were split into separate research aims. This analysis focuses on what has become a common interpretation of multiple representations study, that of managing differing representations primarily due to scale change. Multiple representations frameworks not only provide a workable first step in managing large datasets and production efforts in an orderly fashion, but also contribute to the understanding of generalization in a few ways.

Symbology Changing Events

First, determining the scale or theme dependent variables in maps of the same area at different scales to utilize a singular database has led to insight into the generalization process itself. Müller states that the necessity to understand at which scales or range of scales spatial processes occur is one of the driving forces behind generalization (Müller, 1991). Buttenfield et al. followed the symbology changes that occur with scale changes in published maps and noted some trends that can be exploited in cartographic rule bases (Buttenfield et al., 1991). There are several operators such as consolidation, typification, and collapse which can be commonly applied during specific scale changes. Furthermore, the general observation that diagnostic type tasks seem to be relatively easy while applications requiring synthesis, like layout, seem difficult was made during multiple representation studies.

Feature-based Partitioning

Additionally, Buttenfield et al. determined that “transportation features were deemed to be very important to the inventory of features because they retain a high visual prominence on topographic maps at smaller scales (Buttenfield et al., 1991).” They went on to build a tessellation from the transportation network and indexed non-linear features inside these tessellations. This tessellation
concept is exploited later in this report to help structure the generalization process through partitioning the data based on linear features found in the database, as seen in Chapter 5, which when combined with rules, operators and algorithms, begins to enable multi-scale representations from a single source.

### 3.6.2 Multiscale Databases

The first automated approach to multiple representations consisted of storing separate datasets in a single database. A different representation was produced for each desired scale range, yet each visualization could be accessed from a single interface. This multiscale database model was useful because it offered the user the desired functionality of being able to zoom in or out and have well constructed maps for different scale ranges. However, this model, though still in use, is inefficient to support because it does not eliminate the redundancy issue of linking features across scales from a single data source. This inefficiency has led to research in associating geographic features in different data sets in the same database.

Bruegger, Frank, and Kuhn propose a system of multiple, topological representations connected by hierarchical faces (Bruegger & Frank, 1989; Bruegger & Kuhn, 1991; Bruegger, 1995). The lowest level representation shows all features, while higher level representations show only high priority and other dominant features. The features are displayed through several resolutions. Other studies also pointed out the use of aggregation hierarchies, such as (Casasnovas & Molenaar, 1995). Multiple representation studies pointed out the need to preserve topological consistency across different datasets covering the same area. Laurini and Thompson showed that positional error and topological consistency should be monitored across representations (data sets), which requires matching features across data sets (Laurini & Thompson, 1992).

### 3.6.3 Update Propagation

Many studies have been performed which seek to link features and areas found in separate datasets, despite the representation and to allow the updating of one master dataset and propagate the changes to smaller scale representations. Buttenfield reports on an initiative to determine a formal dimension-independent approach for multiple, hierarchical representations of features (Buttenfield, 1989). Various database designs have been proposed which would effectively allow a single dataset at one scale to have multiple representations. These designs involved many problems of matching features. van Oosterom and Schenkelaars demonstrated the first GIS that could manipulate a single database over a large range of scales, using reactive data structures, though the initial results did not match professional cartographic quality levels (van Oosterom & Schenkelaars, 1996). Devogele et al. prepared a multiple representation database using two French National Geographic Institute (IGN) data sets (Devogele et al., 1996). The first section of their study required that the data schemas corresponded. The second section associated objects in one data set with their corresponding objects in the other data set.

Kilpeläinen defined the aspects of a multiple representation data base (MRDB), by demonstrating object behavior over four different scale levels (Kilpeläinen, 1997). When a feature is added at the largest scale, the feature is also propagated to the other representations. This process can be envisioned in Figure 3.10. Kilpeläinen and Sarjakoski propose a concept similar to incremental compilation in compiler design: when a modification to a data set is needed, only the affected module is updated (Kilpeläinen & Sarjakoski, 1995). This method of updating has been called incremental generalization. This modularity of the dataset requires the existence of discrete partitions of the data set. Kilpeläinen in (Kilpeläinen, 1997) notes that, “The problem of how to divide the generalization task into modules is similar to the problem of the displacement of generalized objects studied by (Ruas & Plazanet, 1996)”. This need for partitioning to limit displacement, and constrain updates to only
Figure 3.10: Data added to a geographic database is propagated to a master database, as well as interacting with methods for each representation that determine which modules need to be cartographically refreshed. The incremental generalization modules only refresh partitions that have been affected by the master database update (from Kilpeläinen, 1994).

Affected regions is expanded further in Chapter 5. In order to modularize or partition the problem, cartographic features need to be subdivided by the conflict types they can spawn. Aasgaard presents conflict sets that seek to define the interaction of various conflicts (Aasgaard, 1992). However, Kilpeläinen notes that these conflict interactions are dependent on how the module division (or partitioning) is specified (Kilpeläinen, 1997). Peng extends the theory for conflict set interaction by specifying the existence of boundary features which would be present in any partitioning (Peng, 1997). Peng dubs the subdivision of geographic features, generalization units.

Mechanisms to associate features via connectivities or references between features across different datasets have been defined in (Kilpeläinen, 1997; Timpf, 1998). In the ideal case, there would be a base data set which contains unique references to all geographic features. This data set would be at the largest scale concerned for the geographic area. Timpf states that subsequent representations should be generated from strictly hierarchical representations (Timpf, 1998). That is, the 1:250,000 sheet was derived from the 1:100,000, which was derived from the 1:50,000 sheet, and so on. Kilpeläinen allows sheets to be derived from multiple ancestors in the scale hierarchy (Kilpeläinen, 1997). In current practice for many NMA's however, map sheets are not derived from any larger scale map sheets in the same series. They are often generated via independent or separate base material and have a good deal of inconsistency across sheets covering the same area at different scales. Additionally, Kilpeläinen in (Kilpeläinen, 1997) notes that dependencies among objects cannot be fully managed with the simple hierarchical methods because a higher level of structural intelligence is needed. The intelligence would assist analysis of the dependencies among features and also how the impact of operators affects those dependencies. This intelligence can be introduced by combining the techniques proposed by (Ruas, 1995; Peng, 1997; Brazile, 1998) and discussed in Chapter 5.
3.6.4 MRDB described

To summarize the features that a multiple representation database (MRDB) would contain, Kilpeläinen describes the requirements and a model that would comprise an MRDB system (Kilpeläinen, 1997). The requirements are to provide

- a data model and data structures that manage separate representations
- an object (feature) directory
- interconnect levels
- flexible representation levels
- indexing mechanisms
- a level of decision processing
- support for user queries.

The reasoning dimension and the representation dimension she points out, highlight the differences between database generalization and cartographic generalization. An MRDB is primarily responsible for managing database generalization. Thus, the model she presents follows the following guidelines:

1. A MRDB occurs in a model generalization environment
2. The data in a MRDB is organized in terms of levels
3. Geographical data at each level are organized as objects and relations between objects
4. The system supports the use of topological relations between features
5. The system can be object oriented, which is conducive for cartographic data
6. Different representations of the same object at various levels are linked with bidirectional connectivities, or references
7. Reasoning or process control guides the use of database generalization operators, which are aware of the utilization and maintenance of the bidirectional references.

A distinction between a MRDB and a cartographic database, she summarizes, is that “no overlapping conflicts (occur in the database) generalization phase because there is no real geographic extent and objects do not appear in graphic space.” The utility of this definition she emphasizes with Figure 3.11. Here, the final cartographic database can expect to apply such cartographic generalization operators as simplification, smoothing, (re)symbolization, exaggeration, and displacement. In Figure 3.11, she points out that positional accuracy has been violated while the topological relations have been preserved. Kilpeläinen further proposes a measure for semantic accuracy, which in effect points out that two objects far from each other should still lie proportionally far from each other after generalization transformations have taken place (Kilpeläinen, 1997).

She further states that a distinguishing element of an MRDB to the user is the conceptual resolution level. The grade of conceptualization of the object increases with the grade of the level relative to the base level. This means that the representation of an object changes from level to level. As an example, a building may be represented by a complex polygon, while at the next smaller scale, or higher level, the representation is a simple polygon. Further still, the building may be a point at the third level, and part of an aggregated area at the fourth level. It is interesting to note that she identifies approximately four levels that represent the major symbolization stages of a geographic feature in a topographic database.
Table 3.1: Kilpeläinen’s MRDB Representation Levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Scale</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>1:5,000</td>
<td>are the most complete and accurate data including the attribute data</td>
</tr>
<tr>
<td>Level 2</td>
<td>1:20,000</td>
<td>real-world abstraction; data are derived from the base level</td>
</tr>
<tr>
<td>Level 3</td>
<td>1:100,000</td>
<td>real-world abstraction; data can be derived from the base level and/or Level 2</td>
</tr>
<tr>
<td>Level 4</td>
<td>1:500,000</td>
<td>real-world abstraction; data can be derived from the base level and/or Level 2 and Level 3</td>
</tr>
</tbody>
</table>

Figure 3.11: A. shows the MRDB at level 3. B. shows the representation at level 3 of a cartographic database. C. shows a cartographic database after generalization operations took place (from Kilpeläinen, 1997).
3.7 A Review of Existing Generalization Systems

3.7.1 ARC/INFO: with RevPG and FTI

Environmental Systems Research Institute (ESRI)’s ARC/INFO is a general purpose GIS which computes geographic data topology and allows the symbolization and editing of data. The U.S. Geological Survey (USGS) developed a series of scripts and programs by the early 1990s called RevPG using ARC/INFO as the base software system. RevPG allowed cartographers to symbolize Digital Line Graph (DLG) features and aid in the data capture process, extracting information from USGS Digital Orthophotos (USGS National Mapping Division, updated 1995). This legacy application contains approximately 150,000 lines of ARC Macro Language (AML) code to perform tasks to extract data, symbolize it on screen, alter individual feature symbology, and thus generally allowing the Revision of cartographic Products to be Generated (RevPG). The aim was to support all possible features in a DLG data set and demonstrate examples of how to work with trouble features. Accurate symbolization of the USGS data (or any national mapping agency data) is hardly a trivial task and this program took a few years of applied research and experimentation to develop.

The USGS is currently developing a new computer application for feature extraction to update the technology found in RevPG, called Framework Tools Interface (FTI). The FTI effort is replacing Product Generation Environment (PGE), which was meant for DLG-E data, though the symbolizing routines are being reused from RevPG (Cress, 1999). Maintaining the procedures in the legacy AML code for symbolization is not ideal. The problem stems from the fact that “business” logic or the generalization rules are integrated into the software platform, rather than being platform independent. Peng points out that using a rule-based system rather than embedding logic into the procedural language is better for a number of reasons, including providing more flexibility in accommodating users with differing requirements and allowing changes to the logic that do not require often costly source code modifications (Peng, 1997). He does counter that a rule-based (or constraint rule-based) approach is not better than a procedural-based approach in the sense of geographic analysis and geometric transformation, and, he leaves off, is probably more complicated to initially engineer. However, once built, an expert system is more flexible to extend.

Not only should the rules to generalize data be made more interchangeable, but the procedures to symbolize the data across platforms and networks should also be provided. Research attempts to do this are described in Section 6.4.

Otherwise, ARC/INFO version 7 without the RevPG technology offers only a few generalization operators and algorithms by default, such as line filtering and some centerline detection commands (though the new ARC/INFO version 8 introduces extended functionality). Independent researchers have implemented generalization functionality on top of the standard ARC/INFO distribution before version 8, but (Schlegel & Weibel, 1995) point out the standard ARC/INFO data model lacks features that would assist the generalization process. However with the recently released version 8 of ARC/INFO, ESRI has re-designed the data model for an object oriented system, which has the desired functionality to support generalization, called for in (Schlegel & Weibel, 1995). ESRI has also continually refined their ArcView product, which has a separate code base from the legacy GIS data model in older versions of ARC/INFO and converges with the technology of other object oriented GIS vendors.

3.7.2 CHANGE

CHANGE is a generalization system developed at the University of Hannover that uses batch algorithms to carry out the mechanics of generalization, leaving expert knowledge to help orchestrate and refine the strategy (Grüneich, 1992; Grüneich, 1995). The program generates maps with fully automatic, yet approximated generalization solutions. Interactive editing is then used afterwards
to produce a map with desired properties. This system does not integrate topology to perform the transformations and is known to primarily be used with large scale cadastral data and street plans, up to scales of 1:25,000. It has been estimated that CHANGE automates up to approximately 50% of the generalization workload (Baella et al., 1994).

3.7.3 MGE Map Generalizer (MGMG)

The Map Generalizer package in Intergraph’s MGE provides trained cartographers the ability to interactively edit features and use a toolbox of operators (Lee, 1995). The parameters to the operator algorithms are set manually and in real-time. The system performs best for independent (non-contextually based), linear filtering and simple area feature generalization. Although the program was virtually retired for a brief time except in research, further software development has resumed under the new product name DynaGen (Smith et al., 1999).

3.7.4 URNAGS and ISNAP

Peng describes a system that generalizes linear data by iteratively selecting candidates from a decision tree, updating the topology to reflect changes, and performing a cursory evaluation (Peng, 1997). The program is implemented in C and linear data is imported from ARC/INFO. The system focuses on classification hierarchies, linear feature length and sinuosity. The system can also reinstate features that were automatically removed. Peng proposes that this type of system could be used during structure recognition to help determine context, especially for patch areas and hydrographic networks (Peng, 1997).

Peng also extended the Formal Data Structure model (FDS), originally developed by Molenaar (Molenaar, 1989; Molenaar, 1986; Molenaar, 1991; Molenaar, 1995), to create an Enhanced Formal Data Structure (EFDS) that supports adjacency relationships (Peng, 1997). Peng used this data model to create ISNAP, which implicitly uses unconstrained and constrained Delaunay triangulation to perform evaluation. Peng demonstrates the use of ISNAP for the following nine generalization applications (Peng, 1997):

1. determining and inquiring adjacency relationships
2. checking the spacing and spatial conflict detection
3. perform object aggregation
4. finding safe-regions, or space in which an object can be displaced while maintaining relative proximity with neighboring features
5. performing object displacement
6. propagating displacement vectors among clusters of objects
7. performing object exaggeration
8. detecting feature patterns
9. Digital Terrain Model (DTM) relief generalization
3.7.5 MAGE

Developed at the University of Glamorgan (UK), MAGE relies on a triangulated data structure, the Simplicial Data Structure (SDS), to compute proximity information between cartographic features (Ware et al., 1995; Jones et al., 1995; Ware & Jones, 1997; Jones & Ware, 1998; Ware & Jones, 1998). MAGE has been used for the generalization of large-scale topographic map data from the Ordnance Survey of Great Britain (Jones et al., 1995). Generalization operators were developed that directly access the SDS, and include amalgamation, building simplification, collapse or the deconstructing of road centerlines, and object exaggeration. Jones et al. further points out that triangulated data structures facilitate the maintenance of topological relationships (Jones et al., 1995).

3.7.6 Plage, Stratège

The French National Geographic Institute (IGN) developed a context independent generalization platform called Plage (Lecordix et al., 1997; Plazanet, 1996; Mustière, 1995). Plage is used primarily to test ways of performing structure recognition and generalization on lines and includes the ability to determine inflection and critical points on linear features, as well as smoothing, simplification, and so-called line caricature.

Stratège offers a richer system for performing context-dependent generalization and was developed also at the IGN. Ruas in (Ruas, 1998b) points out that the system, implemented in Lisp and utilizing experiences derived from Ilog’s expert system shell Smeci (Ruas & Lagrange, 1995), is object oriented and enables rules to drive object behavior. She cites the goals of Stratège to focus on the selection, aggregation, and displacement operators, and also to conduct experiments on the execution of strategic generalization goals and guiding dynamic behavior, especially for urban topographic contexts.

3.7.7 LAMPS2

LAMPS2 provides an object oriented database, base classes of geographic features, and also generalization methods that can be applied on those feature classes, via the standard Generalisation module (Hardy, 1996). Objects can both be assigned pre-defined, by varying representation behavior for set scales and other map controls, or a cartographer can choose an interactive interface to guide generalization operators on specific features in the database to create a professional map. Interactive operators used on multiple objects, such as aggregation, typification, and displacement are provided, as well as operators on singular objects, such as collapsing, refinement, exaggeration, and simplification (Hardy, 1999).

The system also provides the possibility to program dynamic behavior for the features in the form of a procedural, user application language. Operators can be chained together, or sequenced, on specific sets of features that were identified programmatically. This capability has led to further research, in conjunction with the AGENT project, described below, to automate generalization orchestration.

There have been a few trials of the LAMPS2 system in generalization research. João uses the LAMPS2 base system to perform empirical studies to measure the effects of generalization algorithms on geographic features (João, 1998). Kilpeläinen used the system to demonstrate model generalization and creating and storing multiple representations of geodata (Kilpeläinen, 1997). Harrie uses the LAMPS2 system to demonstrate the feasibility of propagating updates between cartographic data sets, with some amount of success (Harrie, 1998).
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

Figure 3.12: An illustration of the negotiating network of the agents responsible for features in the dataset (graphic published in Hardy, 1999; originally derived from AGENT internal reports, 1997). In this case, one agent is representing a primary partition (see Chapters 4 and 5) and would be used to communicate evaluation statistics such as density to other partitions agents, who may be determining if the map was generalized consistently.

3.7.8 LAMPS2 and AGENT

There is a collaborative EU funded ESPRIT project (LTR 24929) that is tasked to develop methodology for automated generalization. The project lead is with the French National Geographic Institute (IGN). The project consortium is using the LAMPS2 software as a base system, and adding new object functionality via the formulation of measures, automated evaluation, partitions, and generalization strategy. The research partners include the University of Edinburgh, the French National Polytechnic Institute in Grenoble, and the University of Zurich, where this author is situated. AGENT is unique in that it extends the object oriented technology a step further by adding collaborative and communicative ability to the features themselves through agent techniques, in addition to adding to the existing toolbox functionality of generalization operators and measures, some of which have been formalized (AGENT: Lamy et al., 1999; AGENT Consortium, 1997; AGENT Consortium, 1999c; AGENT Consortium, 1999a; AGENT Consortium, 1999b).

Figure 3.12 shows agents, represented by the spheres, and their corresponding negotiation network. In this case, one agent is responsible for the primary partition that comprises the center of the map, bound by two major roads. The agent in this case might be responsible for communicating evaluation statistics such as density or the description of building selection to other partitioning agents. The agents may communicate amongst themselves to determine if the separate partitions were generalized consistently.
3.8 Conclusion

First steps towards automated generalization began in the late 1960s with simple but useful goals such as the filtering of lines for improving on screen or plotting performance. At about the same time, formalization of generalization was taking place as researchers attempted to define the requirements, procedures, and algorithms that would be needed to generalize a map. Various frameworks that included all the stages necessary to generalize a map were presented at a very high level, while empirical work analyzing cartographic data in digital form was also conducted. One thrust of this empirical work studied the cartometric evaluation of data sets via geometric measures. The other main research was conducted on performing database or statistical generalizations, attempting ways to categorically reduce the number of features shown at a particular scale. For additional overviews of perspectives, procedures and terminology, of both database and cartographic generalization, see (Brassel & Weibel, 1988; Buttenfield & McMaster, 1991; Müller et al., 1995; Weibel & Dutton, 1999). Results on these experiments were then reintegrated into the formalization attempts and delineation of basic generalization operators were being enumerated. Researchers and practitioners implemented these operators with computer algorithms that could be reused on different data. With a combination of these algorithm driven procedures and manual intervention, product generation was digitally assisted. Ways to combine these separate generalization products were proposed, via multiple representation frameworks. Also, with the aid of computer technology some studies were conducted which recorded the movements of cartographers in an attempt to better understand and formalize the domain knowledge of cartography and generalization. Large rule bases were created which sought to explain how to handle commonly occurring generalization problems. These rule bases were then combined with artificial intelligence techniques in an early attempt to automate generalization, though they often lacked the ability to process cartographic datasets holistically.

Eventually, more complex cartometric evaluation led to the pursuit of context-based approaches and ways to reduce human intervention. Context, with respect to cartographic generalization, is the notion that procedures on sets of cartographic features should somehow be directly sensitive to neighboring features and also perhaps indirectly towards other semantics or constraints. Context-based generalization systems then seek to solve generalization problems practically in their entirety for a particular region of a map. When context-based procedures are coupled with adequate rules for symbology, sufficient structure recognition, real-time evaluation of generalization operators, and reporting, a complete, automated generalization system begins to take form. Such a combination of methods and procedures is envisioned and is presented next as a context-based map generation framework.
Chapter 4

A Framework for an Evaluation-based System

4.1 Introduction

Automated cartographic generalization has become a complex process to illustrate en bloc partly because of intricate production flow and the many data structures, concepts, and algorithms involved. This complexity also prohibits algorithm and system designers from easily appreciating and integrating geographic context during software research and design. Generalization researchers have proposed different ideas and components about how generalization could be automated, as cited in Chapter 3. However, these ideas are often originally presented with either a high level of abstraction describing extremely basic procedures, or are applied with narrow scope and great detail to a specific problem, often without indicating any integration or holistic generalization issues. In addition, many researchers are citing a need for a way to assess and compare the results of their algorithms, through a quality evaluation system perhaps, and also to compare their process orchestration. The quality evaluation system should also be able to assess solution quality at different levels of granularity. Any proposed design should specifically identify the main processes that a working, evaluation and context-based system would require.

This chapter seeks to build a framework for an evaluation-based system with a contextual awareness of neighboring features, by providing a visualization of the generalization process at just enough detail to include the required high-level operations in automated generalization and indicate how those operations are sequenced together. This chapter adds to that sequencing a place holder for integrating run-time quality evaluation of solutions based on geographic . This framework was initially proposed in (Brazile, 1998).

4.2 Assumptions

The proposed design extends and integrates previously published frameworks and models for generalization (Brassel & Weibel, 1988; McMaster & Shea, 1992; Ruas & Plazanet, 1996) and serves two main purposes. The first is to present a metaphor which provides an overview of the complex processes in digital generalization. As generalization means many things to many people, it is useful to consolidate the most commonly occurring concepts into one integrated system. That consolidated system could form the basis for agreement or discussion for a context-based generalization platform. Also encased in the proposed metaphor are concepts for integrating quality criteria creation and
Figure 4.1: A high-level state diagram of a generalization machine which incorporates neighboring features context and run-time evaluation.

Analysis for use in run-time decision making. The aspect of quality being monitored and controlled is the level of fitness for use or how well the desired product conforms to requirements, both high level such as serving specific map purposes through common communicative means and low level such as the original quality and accuracy of the source data and the degree that the accuracy can be preserved to maintain the map purpose. These notions of quality as they related to generalization are described further in (Guptill & Morrison, 1995; Ehrliholzer, 1995; Ehrliholzer, 1996).

The second main purpose of the proposed generalization system is to present a basic design framework for an actual machine capable of generalization decision support and automation. This proposed design shows the atomic actions that will need to be performed by the machine. The atomic actions are illustrated indirectly as states a machine should reach during processing. The general sequencing of these states is also indicated.

Figure 4.1 is meant to help in visualizing the state transitions. Each ellipse in the diagram refers to a state and is numbered for identification. A state diagram is used instead of a process oriented diagram because the process or procedural details can be initially abstracted from consideration in order to explain the overall workflow. Differing implementations could simulate the machine as represented by the state diagram.

This machine design includes ideas for a decision support system by offering accountability, process logging, run-time quality analysis, and user oriented solution visualization prior to execution. However, a future aim is that once the machine has been given the initial criteria, it can create a reasonable solution without human interaction. To meet this aim, the design of the system should incorporate the possibility of full automation of each atomic action which occurs after the user has given high-level, guiding parameters at the beginning.

A subset of this design has been implemented and tested to determine if measures proposed in the literature are useful for run-time evaluation and to see if the proposed process organization and

33
feature partitioning is viable. The partitioning concepts are described in Chapter 5 and the implementa-
tion is briefly described in Chapter 6. The test cases focus specifically on interaction between small-scale linear features (road network) generalization, hydrography and settlements. These cases allow the evaluation of a few published measures from the different measure classes described below. Tests of new measures and partition implications could be extended by future research.

Additionally, no known generalization technology matches yet the sophistication suggested in the proposed machine design, though some projects are converging in this direction (AGENT: Lamy et al., 1999). Many algorithm, processing, and sequencing details are still under research and development. So too are formalizations and criteria needed to determine comprehensive quality methods. However, based on current research directions, this initial design and the subsequent partial implementation, with a focus on the software infrastructure to support run-time quality evaluation, should build onto the body of generalization knowledge and validate the overall metaphor and possibly the utility of this initial design.

### 4.3 Framework Overview

#### 4.3.1 The Main Processing Stages

This description provides a summary of necessary machine operations, if the machine existed in its most complete form. There are three main processing stages in the machine: initialization, placement optimization, and final reporting.

In the **initialization stage**, the requested output and generalization controls in the form of parameters and high-level constraints are confirmed or reset by the user. Then the machine attempts to resolve the high-level constraints and priorities into computable constraints and priorities. Afterwards, the machine builds the associated and auxiliary data structures and checks for the completeness and to some extent the correctness of input data such as topology and attribute encoding to continue processing. Upon approval, the machine then rates the initial map with the given symbology and initializes the array of quality measures, some based on this initial rating. Then, the machine creates a new map space which will be used as an intermediate step for feature placement. The set of highest priority feature classes will be committed to the target map via this intermediate map space. Upon placement of each feature class, a series of checks and warnings will be performed which will in part be used to build an associated final report later during subsequent processing. After the highest priority feature classes have been placed, the map is partitioned primarily by one of two major methods, namely by an object-primary system, where linear features that form topology important to map readers suggest likely partitions. The other form of partitioning, space-primary system where the partitions are formed according to some geometric tessellation, if used would be used transparently to the generalization process. After the map is partitioned, the machine moves to the next main stage of lower priority feature class placement and space optimization.

The **optimization stage** is where the most difficult generalization decisions take place. The few, highest priority feature classes are assumed to have been placed successfully. Then the machine must determine how to place the remaining features in the lower priority feature classes, if possible. The partitions are analyzed for information density or complexity and for each partition, the machine proposes a number of potential solutions. These solutions are stored in a strategy table and ordered according to the degree of matching the user selected guidelines. The potential solutions may be visualized or modified at this step if the user desires. Then, the selected or highest ordered solutions are executed and checked, while the changes are stored in a log. Also for each completed partition, a partition report is built which contains information useful for the final quality measure array. In general the proposed solutions will attempt to utilize global rules and constraints unless either the user has accepted a high degree of local solutions, assumed to create an inconsistent and inferior
map, or global rules are too rigid or undescriptive and the option to reform global rules has been rejected. However, if the machine cannot propose an adequate set of solutions, it generally will first attempt to modify the global behavior of the machine and will restart processing of all partitions at the beginning. Once all partitions are complete the machine moves to the last stage of final report building.

In the final reporting stage, the machine will create the final quality report which creates an assessment of the target map and compares all the classes of quality measures of the source and the target maps. These quality measures cover a broad range of map information, including geometric, topologic, semantic, aesthetic and map fulfillment measures. If however, there were unresolvable problems in the earlier stages and the machine must terminate unsuccessfully, processing will occur in the final reporting stage which attempts to provide clues for overcoming the fatal errors. In either case, upon completion of the final reporting stage, the machine terminates.

4.3.2 A Basic Example Visualized on a Sample Map Image

This section demonstrates some procedural elements of the proposed design with a hypothetical map generalization example. Partitioning, which this proposed design relies on heavily, is explained in fuller detail in Chapter 5.

Figure 4.2 shows a typical U.S. topographic map, originally symbolized and published at 1:24,000, minus accompanying text lying outside the neat line. The original graphic is 84.1 x 118.9 cm. The primary partitioning feature classes are identified from specifications and other sources. For this map, they include major roads, major rivers, and map boundaries. This step is visualized in Figure 4.3.

Sometimes additional constraining features are needed to build partitions and limit the scope of processing. Figure 4.4 adds urban boundaries as a constraining feature which could be computed in a number of ways. The shape of the settlement boundaries should be used and then generalized. To compute a de facto urban boundary that may not agree with the incorporated boundaries in the data set, a procedure to aggregate all buildings and settlements and combining them in some form with an epsilon tolerance, buffering, or Minkowski summation operation, is needed.

Next, all non-linear settlement features, such as buildings and landmarks, are indexed into source partitions formed by all linear features forming topology. Figure 4.5 shows all linear features that form topology and all urban settlements that fall inside the resulting partitions. The zoom-in window highlights two target partitions and the buildings that were associated within each corresponding source partition. The target partitions differ from source partitions because the amount of graphic area has changed in this case due to the enlargement of symbology for the new scale. If the linear network changes due to linear feature selection (removal), then source partitions would be matched to target partitions. The linear feature network would be likely to change in areas where the road or alley network is dense and the scale change is intermediate, for instance 1:10,000 to 1:100,000.

Because primary partitioning features are not likely to be removed during generalization, the worst case for the displacement of objects and the resulting alteration of topology is limited to a primary partition. Hence, Figure 4.6 shows that the primary partitioning features, which are assumed to be fairly regularly distributed in urban topographic maps, are the first features to be generalized. The remainder of the linear feature network would then be generalized and placed inside the primary partitions. The new target partitions would be populated with features indexed into their corresponding source partitions. For each secondary partition, additional generalization operations such as building simplification, displacement, exaggeration or typification would then occur. Then, the quality and measuring mechanisms would insure the desired level of consistency. How the quality mechanisms and the map generalization engine and infrastructure are incorporated is described next.
Figure 4.2: A sample U.S. Geological Survey topographic map image, Austin East, Texas, originally at 1:24,000, here at 1:100,000. This figure is shown to indicate the general cartographic form of an urban area as published by a North American national mapping agency.
Figure 4.3: The primary partitioning features are identified from specifications. For this example, they include major roads, major rivers, and map boundaries. The major roads were collapsed to centerlines. For indexing purposes, this is okay. Note the narrow, river-fed lake shown at the bottom is not collapsed. Also, two road segments selected from the high priority feature classes are very close together. The partition formed by them at other bounding features will undergo post-processing.
Figure 4.4: Additional constraining features are added during this process, such as the determination of urban/rural settlement boundaries. This will be used to characterize the impression of the settlement and help constrain displacement of indexed features.
Figure 4.5: All non-linear features are indexed into (source) partitions formed by all linear features forming topology (roads, rivers, railways, etc. but not contours). The zoom-in window highlights two target partitions, shown 1 and 2, which will need to accommodate the buildings indexed to their corresponding source partitions. The target partitions lost graphic area due to symbology changes that occur during generalization.
Figure 4.6: The primary partitioning linear features are generalized for the target scale. In this case, a target scale of 1/2 the original, approximately 1:50,000 (shown here 1:200,000) is performed.

4.4 Detailed Description

Each of the three main stages, initialization, optimization, and reporting, have been broken down further into states and are described here in more detail.

4.4.1 Initialization Stage – Rules, Map Infrastructure, and Calibration

State 1. Start (generalization machine exists)

In the beginning, the generalization machine exists and is available for processing input data. It begins by first accepting user selected parameters for global controls and constraints.

State 2. Global rules formed/reformed

Entry into this state may come at the beginning of the life of the machine run or later, after processing indicates a reformulation of options is required.

To achieve this state, many production details and guidelines are needed by the system. First, the source data sets are provided or indicated to the machine. Then the user indicates all of the initial generalization controls. The list of controls include

- map purpose
- source scale and desired target scale
- reading conditions
- graphic limits
- symbology table
• output device capabilities
• output specifications
• accuracy requirements
• degree of local autonomy authorized
• other input parameters, input data, meta-data

These user-guided controls will be used to determine the default graphic limits for individual feature classes and the symbology used in the map.

Next, available feature classes and their sub-categories will be prioritized. The machine proposes the necessary ordering of feature classes for prioritization based on map purpose and other controls, but the user may adjust these priorities. The prioritization step that occurs before State 3 will determine which feature classes are selected and placed first (highest priority), which lower priority feature classes are required, and for some less rigorous applications, which remaining feature classes are desired but not required.

A possible prioritization scheme (though not used in this study) would intersperse feature class types; for instance, class ‘A’ roads (transportation) may be placed first followed by buildings of class type ‘large’ followed in turn by class ‘B’ roads, etc. Lower priority feature classes may also be ordered but their placement rank for final map will be computed holistically and where the ranking would often be equivalent. A more likely prioritization and placement scheme, indicated in Section 4.3.2 and described in full detail the next chapter, prioritizes first within geometric types, linear, area, and point features.

At this point in the state transition, additional options may be specified or original controls revisited. The minimal dimensions for each feature class or other additional graphic limits may be altered. Priorities or constraints for symbology choice may be set. Because the computable constraints are determined automatically, in an interactive system, additional constraints might be needed. So, additional high-level constraints to accuracy or graphic representations can be provided or changed such as special cases for topology preservation (e.g. to increase or decrease the recognized topology rules), semantic preservation (e.g. to insure that certain symbology characteristics describe a semantic the system may not be configured to detect), additional measurable aesthetic requirements (e.g. such as consistency indicators) or measurable map purpose fulfillment requirements (including accuracy, theme: make sure all ball parks are indicated on a school sports map regardless of size or general priority, etc.). Conflicting priorities or parameters may be determined already at this step, in which case the user can re-specify them.

When all user selectable parameters have been indicated, the machine translates the non meta-data options into computable constraints and processing guidelines. The user may visit and edit these constraints or re-specify and reapply the module which determines the computable constraints. But once this transition is completed, the constraints will remain fixed. However, as will be indicated in the Optimization Stage in 4.2.2, a solution may violate some constraints.

State 3. **Hard and soft constraints initialized**

The concept of generalization constraints was proposed in (Beard, 1991b) and extended in (Weibel, 1997b; Lagrange, 1997; Ruas, 1998b). Generalization constraints are limits which indicate the minimum requirements for managing space, establishing neighborhoods of objects, preserving topology, or establish other information for map features. A hard constraint may not be violated and defines the unassailable needs of how a feature may be placed on the final map. A soft constraint suggests a placement guideline – perhaps a limit which should not generally be violated, but may be
relaxed under certain conditions. Other discussions of constraints for generalization can be found in

There may be an overall percentage of soft constraints which may be relaxed, according to the
level of local autonomy granted by the user before State 2. This percentage would translate to a
scoring system used when the machine computes the holistic positioning of lower priority feature

Some constraints may be determined automatically. By analyzing the global rules and priorities
the machine could utilize a pre-existing lookup table of guidelines for hard and soft constraints. The
use of look-up tables in a similar context has been discussed in (Jasinski, 1990; Weibel et al., 1995).

It should be pointed out that the overall design quality of the finished map could depend heavily
on the way soft constraints are managed. Again, it is assumed allowing too much local autonomy
will produce an inconsistently treated map. Also, to insure consistency, partitioning information
could be used to group like partitions and to help insure like treatment. However, to what degree
violating soft constraints impacts automated map design quality is not yet known. Initial results
from the AGENT project are starting to determine empirically the acceptable versus unfavorable
amounts soft constraint violations that should be allowed.

Assuming reasonable constraints have been computed and agreed upon, the machine is free to
build the supporting data structures necessary for processing.

State 4. Map infrastructure formed and approved

To achieve this state, the supporting data structures will need to be built and initialized. If the
machine is performing a second or later run because of an earlier problem with map constraints, then
all map infrastructure tests will need to be repeated because the requirements may have changed. The
machine will inspect the supplied source data sets and any requirements will be checked. For instance,
if a specific feature set topology is required, a topological integrity check would be performed. Other
special needs for object proximity, semantics, or other qualities would be created or inspected here as
well. User interaction may be needed to rectify any problems, such as data set cleaning or providing
auxiliary map data intelligence.

When each of these qualities meets the requirements, the machine is assumed to have all of the
infrastructure needed to perform the first steps of feature placement of the highest priority features
into the new map space. However, before that initial placement commences, a quality rating and
calibration of quality methods on the original data set would typically come next.

State 5. Calibration of checks/initial rating built

During the transition to this state, an initialization of the quality methods occurs. This important
step performs a large number of measures on the source data set, both at logical levels and at
geometric levels when relevant, where the original data set is rendered with a pre-specified symbology
and then subsequently assessed. The quality methods used would be arranged in a hierarchy similar
to the hierarchy of constraints described in (Weibel, 1997b), building from (Beard, 1991a). Also
included in the quality method hierarchy are data quality checks, similar to those presented in
(Guptill & Morrison, 1995) and considered further by (Brändli, 1997).

The measures are performed on the map as a whole and not on partitions or subsets of the
final map. Partitions may be used to simplify and organize processing and are described in State 9.
However, various partition statistics may be compared with the source map measures.

Following these experimental lines and the strategy decried in State 11, a specific strategy for handling conflicting
design constraints has been implemented for a multi-agent generalization system in the AGENT project. AGENT
is an E.U. ESPRIT project which was commenced approximately a year after the start of this thesis and involves
the French National Geographic Institute (IGN), Laser-Scan Ltd., the University of Zurich, INP Grenoble, and the
University of Edinburgh.
The results of all measures are stored in a quality measure array or structure, described in a subsequent section, and are used for a basis of comparison when all processing has finished and the final map is assessed.

**State 6. Initial feature class built at target scale**

This state is added to the machine in part to show the condition that some high priority features will likely not be subject to the transformations that other lesser priority features will undergo.

The most important features in the highest priority sub-feature class are symbolized and added to the new, initially blank map space. All the generalization operations needed to add these features are performed here as well. These added map features will not be moved or deleted in any subsequent processing unless the machine initialization stage is revisited and new priorities and feature classes are specified.

Note the concept of anchoring certain high priority map features is a system design choice and does not significantly impact the quality measuring system. Basically, upon completion of committing features to the new map space, a series of checks and warnings are performed and relevant values stored for the quality measure array and the final report. This process, in fact, occurs next and is indicated by State 7.

**State 7. Checks and warnings assessed for the quality measure array**

Measures and checks on the new map space are performed for reaching this state. Checks for topology, geometric integrity, and semantic alterations may be required.

If any serious problems arise, the machine proceeds to a failure assessment mode (this transition not indicated in Figure 4.1). If the problem can be solved easily with user intervention, then, the machine repeats the checks and warnings. If not, the machine provides advice for resolving the problem and returns to State 2 or ultimately terminates unsuccessfully, State 20.

A special case after State 7 is that the new map has only one or a few high priority feature classes and these are not subject to the major generalization processing of the optimization stage. When this case occurs, the machine would proceed to check other quality criteria, State 16, and would progress directly toward completion.

Otherwise normally the machine proceeds to State 8, and back again for checks.

**State 8. Remaining high priority feature classes built**

The remaining high priority feature classes are symbolized, generalized, and placed into the new map space, similar to what was needed for State 6. Quality measures are performed again, as shown by the transition to State 7.

**State 9. Map area partitioned**

After the high priority features have been generalized properly and placed into the new map space, the remainder of the features will need to be generalized and added. A published concept for processing map data sets for generalization first partitions the map space based on natural breaks or clusters, as proposed in (Ruas, 1995). The form of partitioning desired for this design was described in the previous subsection and is also described in detail in the next chapter. Essentially, if major roads were the highest priority feature class, then primary partitions form when roads are used as the boundaries, and secondary partitions are formed by all linear features, which include primary features. The map features that occur in the space delineated by the roads are then generalized. As there could be hierarchies of roads placed in the map, there could also be hierarchical partitions, most likely processed bottom-up, top being the root, where each higher-order partition subsumes
one or more pre-generalized partitions. However, there might be problems with having to integrate
an arbitrary number of partitions, as described in the next chapter.

The partitioning scheme may also use a space-primary scheme (geometric tessellation), although
mostly transparently, that is as a spatial index to speed up geometric search and data retrieval as
found in (Samet, 1990). It is assumed that a space-primary partitioning cannot be reasonably used for
process organization in the way that an object-primary system could be. This is because tessellation
boundaries are very likely to interrupt linear features. However, there might be enough descriptive
data and quality methods to arrange map features in a way that is consistent with the target global
rules. Also, a separate analysis function that crosses partition boundaries could be designed which is
able to find a divided cluster and test to see if each division was processed consistently and propose
repair partitions if not. This approach, however, is not discussed further.

In general, partitions may be processed independently. If for some unforeseen reason, not all linear
features can be used to build the partitioning, it is possible that linear features will cross partition
boundaries, as in Figure 5.7. In this case, the partition data structures may contain auxiliary
information that indicates how to integrate adjoining/neighboring partitions, such as indicating
anchor nodes and gradient vectors for these features that cross boundaries. Otherwise, following the
general assumption of the independent processing of partitions, the machine continues to process
and generalize each partition.

4.4.2 Optimization Stage

“Optimization” in this context is not meant to imply a strictly well-defined optimization problem.
It rather implies a general sense of finding adequate solutions based on constraints, measures, and
computed possibilities.

State 10. Information density found for each partition

After (or during) the partitions have been formed, the machine will analyze each partition as
overlaid on the source map (or source partition) for suitability and trends. One important measure
is the density function of each source partition in the source map. The function could also add
all requested feature classes with the proposed symbology to a temporary map space or provisional
rendering and determine indices of information density. Other logical data sources may be checked
too for finding the information density. This index will help track feature density across the map.
For convenience to a user, these indices may even be visualized as in (Mackaness, 1995b, fig. 2),
which shows a 3-d surface hovering over a town. Where there are more map features, and hence more
information, the hovering surface is higher. The clutter function proposed in (Jansen & van Kreveld,
1998) and the density analysis pre-processing step in (Ruas, 1998b) also explores the concept of
measuring cartographic information density for use in generalization systems.

State 11. Generalization solutions computed

The optimization of placement of all remaining features in the remaining feature classes is com-
puted for the given partition. Essentially, this is where the most difficult sequencing of generalization
procedures occurs. The machine calculates in what can be visualized as a temporary map space to
determine a solution and then stores that solution in a solution table. In general, there will be
multiple solutions for each partition.

These solutions vary in part according to the user-specified guidelines from states in the ini-
tialization stage. For instance, an upper limit on displacement, a soft constraint, may have been
specified in an earlier stage. When the machine goes to process a partition, however, it may propose
exceeding the indicated displacement allowance. So it may first propose and store a solution that
respects the displacement allowance, and rate it highly according to its fit to the global criteria, but
poorly according to some other criteria such as proximity of neighboring features. Later, it may propose and store a solution which exceeds the displacement allowance and rate that solution highly according to proximity but poorly according to following the global guidelines.

It will attempt these solutions by a rating of appropriateness given to them before they were entered into the solution table. If the first solution is found acceptable, then there is no other reason to try the remaining solutions. Otherwise, the remaining solutions will be tried. In the preceding example, the first solution which preserved the displacement soft constraint may fail an aesthetic measure, in which case remaining solutions from the table would be attempted. Perhaps the next solution which exceeded the displacement soft constraint may pass more tests and measures and provide a better overall solution.

The partitions with the highest information densities, yet under the density where all features are combined into a solid block (roughly 60-80% dense), are assumed to be the partitions which present the biggest computational challenges. In an interactive system, these partitions can be singled out for more attention.

State 12. Proposed solutions stored in strategy table

The strategy table holds proposed generalization solutions (tactics) for the given partition. There is one record per solution, where each record contains all the transformations necessary to arrive at the pre-proposed solution in addition to various authorization indicators, checks, and ratings.

This state indicates that at least one suitable solution is in the strategy table and that the machine may continue by attempting the solution. However, in practice, while the machine continues forward, there may be more solutions offered in the background and stored in the table. Probably the machine would cease adding new solutions when a partition has been marked as completed by the same monitor process which determines parallel synchronization and other concurrency issues.

State 13. Strategy table consulted

In an interactive system, the solutions proposed in the strategy table may be visualized. The user could select each solution and the machine could render the partition with the new changes and present this image to the user. The user may also edit the solution record if minor instructions need to be tweaked.

In an automated system, the machine would access each entry in the order of decreasing overall rating, unless a differing sub-rating is preferred for ordering.

State 14. Changes executed/stored in change log

The machine executes the next available solution from the solution table or the solution selected by the user and stores the resulting features on the intermediate map (pre-target). If using a hierarchical partitioning, this intermediate map will only be finally integrated and committed as the target map when the geographic area represented by the partition is no longer part of any other partition. Each resulting partition will be used as input to higher level partitions. Otherwise, in the more general assumption of partitioning choice, the linear features are all generalized first, and the partitions “filled in” with the indexed features.

The changes, both procedural and descriptive, are stored in respective change logs. These change logs may be viewed by a user or perhaps used subsequently (post-machine) to amass data for processing-trend analysis or generalization operator use analysis.

State 15. Checks and warnings assessed for the quality measure array
Similar to State 7, because changes occurred in what will ultimately become the final map space, various checks and warnings are performed. In this case, the checks and warnings are primarily for analysis of the partition, but also are performed as the partition is integrated with the intermediate and ultimately target map space.

State 16. **Preliminary criteria checked**

This state is meant to signal a confirmation stage before committing the changed partition to the list of new target map space. It would be here that additional non-critical checks or measures could be performed, such as map-fulfillment or aesthetics measures. These are indicated as non-critical because some global rules may be configured to ignore these indicators. If however these indicators are activated and the partition fails the tests and there are more solutions in the table which may be attempted, then the machine reverts back and re-attempts a solution from the table for that partition. Otherwise, that partition is completed and a partition quality report is built.

State 17. **Preliminary/contributing report built**

By this stage, the partition has been analyzed and a range of measures computed. These are collated and may be accessed directly by the user or used when the final map reports are computed.

State 18. **Failure assessment performed**

In the event that there was a failure with executing solutions from the strategy table or that none of these solutions were acceptable, then the machine performs a failure assessment. In this failure assessment, three paths may be followed.

The first path is that a minor modification may be recommended and the user allowed to make this change and re-execute a modified solution. In a worse case based on this path, the machine may have failed some test but elected to keep the solution anyway, in hopes that the area will be reprocessed in a higher-level partition.

The second path is that the global constraints were too restrictive and should be relaxed in the global rules specification for State 2. This second option reflects the machine’s bias toward modifying global rules when problems are encountered. It could be that a local solution can be found without changing global rules and starting over. However, perhaps a global rule would be more appropriate and if proposed, would create a better designed map. This bias, however, would only be by default and a user could override this behavior.

The third path is that the machine cannot continue and must abnormally terminate processing. If this path is followed, then the machine tries to create a descriptive report, State 19b, and will also try to indicate the causes for failure and possible remedies.

4.4.3 Final Reporting Stage

State 19 A. **Final report built (Success)**

If this state is reached, in the general case, it means that all the partitions are finished and the last partition was committed into the final target map.

Primarily, a report was built that describes the quality of the total overall effect of generalization. It also contains data which compares how feature classes and conditions changed with regard to the original source map rating. The desired product is a report which shows how qualities, both design and data, of the map were preserved, improved or changed for the worse.

State 19 B. **Final report built (Failure)**
This state was reached because the machine could not continue processing and a user could not intervene with acceptable modifications to any proposed solution. The machine still builds a final quality report, to the degree possible and also tries to build a failure report which indicates causes for failure and possible alleviation.

**State 20. Failed and terminated**

The machine could not finish successfully and terminates.

**State 21. Succeeded and terminated**

The machine finished creating the new generalized map, creates the appropriate reports, and terminates. Post-processing of the finished map may occur in another system.

### 4.5 Conclusion

There are several high-level issues involved in designing a generalization framework that incorporates some level of contextual awareness and evaluation. First the overall process of generalization should be decomposed into basic steps or stages, which hopefully can be handled independently of each other and then reintegrated. This process decomposition was assumed via a form of a partitioning of the dataset, where partitions could be treated separately. With this assumption in place, a generalization state machine was described which demonstrated steps on evaluating the dataset and preparing it for treatment. The generalization machine design incorporates methods for assessing the quality of solutions at different granularity levels, specifies how those methods could be supported via generating an adequate processing infrastructure, and how those methods are sequenced together. The concept of evaluating the source data set using cartometric and other techniques was envisioned here, as well as ways to compare these measurements with the final target representation. In this manner, ways to introduce and enforce consistency of treatment were added to the functionality of the state machine. The machine design also encapsulates the basic decision paths that should be taken in the event of problems. This way of introducing consistency and evaluation into the entire generalization process is argued here to help create a system that can automatedly generalize topographic map data. This machine design takes advantage of new algorithms that offer improved and constraint-based ways of handling map features (Ruas, 1998b; Højholt, 1998; Harrie, 1998). Finding input parameters that trigger meaningful solutions is still a major aspect of research in generalization. However, following assumptions about the structure of urban topographic data sets, this framework supports these new constraint-based methods via partitioning based on linear features found in the representation, which is described next.
Chapter 5

Feature-based Spatial Partitioning to Support Evaluation

5.1 Introduction

This Chapter describes problems and motivations for organizing map space into partitions for map generalization. This organization facilitates the sequencing and organization of the software procedures carried out by controlling modules, in this case agents designed to automate the generalization process. A partition in this context is a decomposed spatial unit created after identifying and generalizing linear features found in the map database. The faces formed by these linear features create containers where non-linear features such as buildings can then be placed and transformed if necessary. A partition seems to be most useful in urban contexts where feature density is high and between scale changes where major features are not iconified. A partition in this case helps to manage the generalization process, particularly concerning topology, density, and feature arrangement, unlike spatial indexes like quadtrees. This document proposes procedures for creating partitions and continues to motivate the problem of integrating partitions and examines difficult cases.

Map partitioning has been proposed as a means for organizing and decomposing map space into smaller, though still geographically meaningful divisions (Buttenfield et al., 1991; Leitner, 1993; Ruas, 1995; Brazile, 1998; Harrie, 1998; Ruas, 1998c). These partitions help in organizing and sequencing generalization procedures and allow some simplifying assumptions particularly concerning displacement. There are potential benefits to using partitions such as providing a possibly necessary geometric framework for constraining the placement of map features and providing a basis for computing feature density statistics, often needed for constraints or goal satisfaction.

There are some generalization situations where partitioning does not facilitate determining a solution. Some clusters of features can span partition boundaries. A cluster then in the context of this thesis is a spatial extent independent of partitions, used to organize those generalization situations which are not topologically based, yet where relationships with nearby features can be defined. For instance, through clustering, the alignment of buildings separated by partition boundaries or preserve relative proximity to neighboring features could be maintained. Even though this clustering problem is not a topological, Peng et al. defines a mechanism to determine what they call the safe regions of features that is reminiscent of explicit topology (Peng et al., 1995). This concept assists in determining further inner-partition constraints about the distance features can be displaced (Peng, 1997) (cf. 6.2.4 and 7.3.3). It is then assumed here that preserving these alignment and gestalt

---

1This Chapter is partially based on Brazile, F. Edwardes, A., Weibel, R. and Mackaness, W., Organizing Map Space for Generalization through Object-Primary partitioning (to be submitted).
characteristics in medium and small-scale topographic maps is a separate problem from preserving topology and, when trade-offs are necessary, of secondary importance (Monmonier, 1991). However, as these situations can affect the overall aesthetics of a map, an eventual integration of clustering and partitioning would be desirable.

There are other possible benefits from using this kind of partitioning, such as in creating an evaluation system. The partitions could be used to organize a quality report on how the map features have been altered during the generalization process (Brazile, 1998), though this role will not be discussed here. Instead the focus here is on the computational need for partitioning, detailing basic procedures for creating partitions and clusters, and considering difficult or degenerate cases.

5.2 Motivation for Partitioning

Partitioning for generalization support is briefly illustrated in (Brazile, 1998) primarily as a mechanism to preserve topological relationships in complex cartographic situations and retain or emphasize priority cartographic elements. Partitioning is also mentioned there for use in divide and conquer strategies. The basic hypotheses where partitions are useful are presented as well as how the creation of these generalization-oriented partitions differ from traditional uses of spatial partitioning, summarized in (Samet, 1990), which are generally used for optimizing range queries, pre-testing for line intersection, and indexing point features.

Topology

Perhaps the single most important reason to employ partitions is to help manage topology during and subsequent to generalization. Creating topological areas, or faces, from linear map features, or edges, provides information on the topological relations of geographic features. By specifying object adjacency and inclusion, topological face data structures help when retaining non-linear features inside the containing topological face after generalization. For instance, by creating faces from edges, it is possible to consistently determine whether a building positioned between two roads before road generalization, are still correctly positioned afterwards. This property is explained further under the section Displacement, though it is already assumed that preserving topology at successively smaller scales is often more important than preserving absolute positional accuracy.

Metrics

Partitions as topological structures can provide additional information, or metrics, which can be used to create goals when an automated generalization system dynamically modifies constraints, including constraints on displacement. Partition-based metrics may also be used when creating or executing procedures on classes or other feature groupings. Partitions assist in the satisfying of metric constraints by providing spatial reference units to create statistics such as relative density (Jansen & van Kreveld, 1998), or to use when making top-down decisions on operator priority (Peng, 1997; Ruas, 1998c).

In addition to providing a geometric framework for the placement of map features, partitions also provide a cartographic framework with which to constrain the displacement of map features. The enlarged or displaced map features, for instance, should not be displaced outside of the containing area suggested by the partition topology. However, the partition is not just used as a binary topology test – if the feature is enlarged and needs to be displaced, the partition metrics can be helpful in indicating the relocation vectors to available space.
Strategy

The partitioning system proposed here emphasizes high priority features, expects their early and largely fixed treatment, and then contributes to lesser priority constraint satisfaction which does not interfere with high priority features. Priority related to partitioning is also useful in satisfying such global, and elusive, constraints as style, theme and ensuring consistent treatment of features throughout the map.

Process decomposition

The usefulness of many contextual operations is likely to be bound by the number of objects involved in a structure or relevant to a context. Partitions thus provide a structure with which to limit the combinatorial expansion of processing a set of map features represented as inter-related objects. If partitions can be fixed, then the generalization process becomes a matter of computing a solution for many small partitions with few features rather than one large map with all features. This process decomposition becomes especially important for novel, but computationally expensive approaches, such as Multi-agent systems and other artificial intelligence related techniques.

Displacement

In many topographic maps of urbanized areas, it is easy to see a network of roads, rivers, railroads and other linear features that cross each other to make up a parcelled view of the landscape. When these linear features are retained at smaller scale mapping, starting after 1:24,000, it is not unusual to require transformation and simplification of these features, where according to (João, 1998), a national mapping agency (NMA) will focus at smaller scales on preserving topology over the absolute accuracy of all features. She notes that it is usual for NMAs to choose roads as the highest priority feature class to retain and draw, after survey points. She points out two exceptions though, such as the Dutch and Portuguese National Mapping agencies, which choose another linear feature, rivers, to have the highest placement priority. When these linear features are simplified, segments of these lines are naturally displaced. Any linear or point features found in the area between the previous segment position and the new segment position will also have to be displaced to some degree in the same direction to preserve topology. Therefore, displacement must be a “local” operation, where the context of the immediate cartographic situation is analyzed. The constraining assumption is therefore suggested that high priority linear features, once in the desired position, should not be moved again. And after the remaining linear features have been generalized, let us similarly assume they are in their proper position and should not be moved again. Thus, the remaining area, computed as the area bounded by closed cycles of linear features and adjusting for the symbology width of the lines, becomes the available cartographic space to work with to place all remaining features.

It is then possible to manage the computation of new displacement vectors for these remaining features or to find other cartographically acceptable adjustments, such as typification. This adjustment management is possible because important, though perhaps not all, related factors effecting the new positions of the features have then been identified. A good solution for placing features in the remaining areas is the subject of other ongoing research in generalization. The current approaches to displacement offer some interesting solutions, c.f. (Genin & Donnay, 1997; Mackaness, 1994; Bundy et al., 1995; Ruas & Plazanet, 1996; Højholt, 1998), though problems remain. For instance, there is not yet a guarantee that the placed features will retain any inter-feature relationships, such as gestalt patterns inside or across partitions, though those issues are not addressed in this research. Finding the constraining area via partitioning insures that important topology characteristics can be maintained, regardless of choice of the displacement algorithm.

For situations where displacement is not enough, this constraining area still plays an important role in finding acceptable solutions. Perhaps the scale change is too great and not all non-linear
features in the partition can be retained. In that situation, it is likely algorithms for amalgamation, characterization, or selection will be invoked. It will be necessary then to compute the density of features in constraining area in order to find a new feature representation which respects the previous scale representation (Ruas, 1995; Brazile, 1998). This area provides the necessary frame of reference for computing feature density and is still the topologically acceptable space for the new representation. Next a way to compute the constraining areas in situations appropriate to urban topographic maps is proposed.

Note also in the situation where selection or characterization may be invoked, the aim is to preserve the “look and feel” of the map at the smaller target scale to the best degree possible. There is no established way to preserve that feel. However, there have been a few proposals (Ruas, 1995), (Brazile, 1998) that essentially call on finding the distribution and density of features in what was the previous spatial extent before the features were generalized, and trying to preserve these values in the target scale mapping of the corresponding area. This method of before and after area statistics is discussed in Chapter 7, and it is suggested that the source area that forms the partition will be an essential component of topographic-style generalization methodology.

5.3 Basic Methodology

The activity diagram shown in Figure 5.1 explains the basic procedures used for partitioning a map. To create the partitions, first identify the partition defining features. The role of user-supplied map specifications will play a part in this determination. As an example, however, major highways and thoroughfares are chosen as the high-priority, primary partitioning features and all linear features, the priority features plus the secondary roads, railroads, and rivers, as the secondary partitioning features. In this case, there are multiple classes of secondary partitioning features, though there could be only one (or none) depending on the features present in the target scale. It is an assumption that at the base hierarchical level, in this example the secondary partitioning level, that all linear features will be included in the partitioning. This assumption arises from the difficulty of merging partitions formed by generalizing linear features independently of each other, as detailed in the section on continuity and homogeneity. Conversely, the primary partitioning level was designed to emphasize the cartographically important notions of feature priority, where linear features often dominate a topographic map after a scale-change. By subserviating all other features to this prioritized network, the cartographic importance or pattern (gestalt) of this network is to some degree maintained.

As indicated by Figure 5.1, a relationship is found between all non-linear map features, buildings for instance, on the map and their respective corresponding, non-overlapping extents or partitions. Then generalize and symbolize the features that created the partition boundaries, first primary features then secondary features, if present. This new, target-scale network of linear features then suggests new partitions, which are computed, in this case, based on the graphic area remaining after highway and road symbolization. Remaining features, here buildings, are then related to the target scale partitions. Primary partitions are thus used as specified, primarily to implicitly organize the secondary partitions and reinforce the semantics of the map based on the prioritized treatment of the primary linear features. For each secondary partition, the buildings are generalized and symbolized, yet their final position remains inside the allocated space of the containing secondary partition, thus maintaining topology in the new generalized map of highways, roads, and buildings.

5.4 Detailed Methodology

As an example of this methodology applied to a real data set, consider the map shown in Figure 5.2. Figure 5.2 is a typical U.S. topographic data set representing an urban area at 1:24,000 scale. In
Figure 5.1: A UML activity diagram describing the basic processing steps taken for partitioning. Note, many details and procedures are encapsulated in each activity box. Solid bars represent synchronization points and asterisks represent multiple process instances.
Chapter 5

Figure 5.2: The urban focus area (Austin East, TX) at approximately 1:24,000. Most map symbols shown here are similar to USGS specifications, though no labels are present.

This map, there is a road network of high priority highways and lower priority residential roads. There are usually also other linear features such as railroads, city administrative boundaries, major streams, and minor or intermittent streams. Along with the linear features, there are also polygonal features such as buildings and area tints representing, for instance, parks or forested areas. There are also additional graphic items present such as labels and special icons which will not be discussed. To simplify the problem and show those features which will most resemble the expected urban data sets, contours have been ignored, as well as area tints and any special features not essential to the topology of the map.

To readily visualize the total generalization problem associated with scale reduction, see 5.3. This figure shows the focus area at a simple scale reduction of the data set from 1:24,000 to 1:50,000 with no generalization algorithms applied. From this image, it is clear that the buildings are often too small and need to be enlarged, if possible. Besides enlargement, building displacement, characterization, or selection is possible.

To determine if enlargement is possible, the area available to the buildings in the partition at the target scale should be calculated. When the available area is calculated, constraints to the alteration or displacement of the buildings also need to be specified, in case enlarged buildings need to be merged, moved, or typified. A stable partitioning scheme will provide both the available area and the displacement limits, which allows topology preservation.

To preserve the topological relationships, partitions are formed using all the linear features, and then partitions that do not meet certain criteria, such as desired size or shape, are merged or cleaned. Unfortunately, due to the presence of access and feeder roads, exit and entry ramps, and divided highways, the road centerlines cannot be directly used to create the partitions. The width extents of the road casings and divided roads are used to determine road centerlines, and there have been several proposals on determining centerlines from casings (Olson, 1995; Christensen, 1996; Thomas,
These extents can be approximated by buffering the road casings or divided highways with a very small value ($\approx 5m$), and using the resulting area outside the buffer as the source partitions. These non intersect regions will form the source, primary partitions Figure 5.4.

Following the presented methodology, the buffer is computed for all of the primary linear features at the source scale, with a small offset, and likewise for the secondary partitions. The resulting areas then represent the source partitions, Figure 5.5. All buildings are then referenced to these source partitions by computationally associating each building inside a secondary partition.

After determining in which partition each feature belongs, the linear features used in partitioning can then be generalized, and the new, resulting partitions computed – the generalization of these linear features is not discussed here, though it is not necessarily a trivial step. The new, possibly modified, geometry of the primary linear features is used to create the extents for the target, primary partitions. Similarly, the generalized primary and secondary linear networks (and map extents) are used to find secondary partitions. Map features, such as buildings, are then re-referenced to the target partitions (Figure 5.5). If the target scale partitions do not match the source scale partitions possibly due to linear-feature selection for the target map, building features from the source scale partitions are then referenced to another target scale partition. This second referencing step is to preserve the evaluation of topology, whether it can be preserved or at least manipulated in a desirable fashion. One such manipulation may be that some buildings belong to a new secondary partition, but still are inside the original primary partitions. When all non-removed features are referenced to a target partition, then the partitions can be used to find the graphic area available to the building symbols.

To find the available graphic area, a buffer can be utilized again, this time, around the symbolized linear features. By buffering the symbolized linear features, the target partitions then indicate, not just topology, but how much available graphic area can be used to help compute the needed
Figure 5.4: Schematic of the focus area (centered rectangle) showing a buffer around the primary road segments and casings. Shaded areas represent source, primary partitions.
Figure 5.5: Map of focus area showing 1:24,000 source, secondary partitions (each non-overlapping shaded area). Buildings are shown with their containing partitions.

modifications with buildings or other non-linear map features. Example target partitions outlines are shown in Figure 5.6.

In Figure 5.6, two partitions are highlighted. The top partition and its associated buildings will probably allow enough space at 1:50,000 for the buildings to be simply squared, displaced, and the smaller building possibly selected. The bottom partition will require complex generalization analysis and processing, probably including a combination of displacement, selection, typification or aggregation.

5.5 Assumptions about Continuity and Homogeneity

The preceding example did not emphasize the complexity of multiple secondary linear features, such as rivers and railroad tracks, which would be used to create the secondary partitions. In fact, all lower priority linear features to be used for the secondary partitioning should be found, generalized, and then used to create a buffer from these linear features using symbol widths.

This step of combining all secondary linear features into one partitioning stage is a design assumption and is taken in response to the following problem: suppose a linear feature were not used in the partitioning process and thus is a candidate feature for simplification, displacement, or selection. This linear feature is then likely to be bisected by a partition, which can cause problems. This situation is shown in Figure 5.7.

In Figure 5.7a, the linear feature, possibly a river, r, is bisected by a partition boundary, b. Figure 5.7b represents a worst case for treatment of feature r because the integrity of the river has been broken. Other equivalent poor solutions are the elimination of either r1, but not r2 (or vice-versa), or the inconsistent smoothing of r, shown in Figure 5.7c, producing two troughs (or two
Chapter 5

Figure 5.6: Map of focus area showing 1:50,000 secondary, target partition outlines. Two partitions are shown shaded along with the unmodified building features belonging to those partitions. The top shaded partition will be relatively easy to process and generalize, compared to the bottom shaded partition.

Figure 5.7: A linear feature, $r$, is not used for partitioning. a. A partition boundary, $b$, bisects $r$. b. $r1$ was displaced in a direction that violates important visual topology with $r2$. c. $r1$ and $r2$ were smoothed inconsistent with the global shape of the original line.
Figure 5.8: Features A, B, C, D would all normally be shown if their allocated partition size were large enough. One design decision is to ignore B and show features with enough space. Another decision, though expensive, is to recreate the partitioning by displacing the river in a deliberate direction in order to mitigate the needs of features A, B, C, D.

peaks), instead of the original trough, then peak, configuration in Figure 5.7a. Without explicit controls in place to prevent this inconsistency from happening, these undesirable case scenarios are possible. There are, however, design possibilities to handle this linear intersection problem without forming partition boundaries from lesser priority linear features. Though, let us assume it will in practice be simpler to use all secondary linear features in the secondary level partitioning. This problem is also treated by Peng who, to paraphrase, suggests linear-units in the map should all be grouped together (by conflict-type) to avoid the inconsistencies of interrupted linear features (Peng, 1997).

Some resulting partitions may have irregularities and will not meet the required criteria such as desired shape or size. These criteria are in fact not so simple and stem more from problems with features that would normally be shown if their allocated partitions were not too small. A post-processing phase is then needed to determine if the partition size or shape can be modified to the benefit of the impeded feature. This phase is where the optimization complexities of generalization become obvious and expensive.

For example, if the curved river in the left group of Figure 5.8 is somehow displaced to the benefit of feature ‘B’ (the river conflicts with building B’s symbology), will features ‘A’, ‘C’, or ‘D’, etc. then be impeded? A quick institutional decision might be to eliminate the lesser priority conflicting feature such as ‘B’ from the final map, and note the inconsistency in a report. This decision may not produce acceptable cartographic results. The expensive and probably desired decision remains to mitigate the needs of all affected features ‘A’, ‘B’, ‘C’, and ‘D’ through optimization, until all that need space and can have space in their partitions, do. Possibilities relying on least square methods, such as the finite element method, may prove fruitful in this capacity (Højholt, 1998) because partition boundaries can be defined as rigid linear constraints.

5.6 Possible Extensions

What this methodology so far has not yet addressed, however, are the semantic relationships of features with other like features located in adjacent partitions. This problem is argued here to be less important than the main topology preservation indicated previously, of features and close by linear features, though certainly still impacting the quality of the produced map. A typical case of this problem might be visualized as a cluster of buildings that are aligned with each other in one
partition, and mirror similarly aligned buildings in an adjacent partition. An independent clustering routine would be needed to integrate and verify the ability of clustering techniques such as those proposed in (Peng et al., 1995; Regnauld, 1996; Worboys, 1996; Peng, 1997), in order to place alignment constraints on the clustered buildings and be processed concurrently, despite being in separate partitions.

5.7 Conclusion

Partitioning a data set via linear features present in the source and target representation is suggested here as a way to structure urban topographic generalization and allow a means for process decomposition. This methodology of first generalizing partition boundaries (linear features) and then features contained in the partitions themselves is asserted here to satisfy the desire to characterize delineated urban tracts, help in evaluating track density, to shadow or mimic the structure recognition needed to generalize the urban portions of the map, as well as constrain displacement locally or to neighborhoods delimited by linear networks. This methodology also allows us to emphasize the semantic structure of the primary and dominant features of the map by treating them first, and grouping the processing of all other features afterwards.

However, there are still a few open problems concerned with partitioning for generalization support. The ability to define the feature classes and individual features used in the partitioning at different levels, managing the varying nature of secondary linear feature generalization, managing clusters interrupted by partitions, and managing features in the map that are simply problematic. Such problematic cases include cul-de-sacs, where a local grouping adjacent to a road cul-de-sac can be unaesthetically rearranged, yet traditional topological relationships have not been altered. However, Peng et al. propose approaches for structural mechanisms that handles cases such as this (Peng et al., 1995; Peng, 1997). These difficulties aside, it is proposed here that partitioning shows promise as a necessary element in current strategies of context-based, urban generalization. A system design which demonstrates the characteristics of linear feature partitioning and applies evaluation measures on partitions is presented next.
Chapter 6
System Design for Integrating Feature-based Partitioning, Measures, and Automated Evaluation

6.1 Introduction
Chapter 4 describes a conceptual framework for cartographic generalization aimed at specifying the general workflow needed and which allows automated evaluation. Chapter 5 introduces the concept of feature-based partitioning that can help organize geographic space and facilitate generalization transformations, with a particular emphasis on urban topographic mapping. This chapter describes a system design and features that are necessary to demonstrate the utility of the partitioning and evaluation framework, connecting the concepts with a logical software design. This chapter also introduces some questions and assumptions that would influence the system design. Several key features have been implemented in a prototype system in order to determine the overall feasibility of the integrated partitioning and evaluation approach.

6.2 Basic Requirements
Urban settlement cartography and generalization practiced by formal groups such as state mapping agencies or commercial map publishers, follows laid-out rules and has well specified needs. The goal for the implementation of a prototype system was to respond to this production environment by being able to handle and process, draw, generalize, and evaluate the urban topographic mapping data that is representative of urban settlements, using the background, framework, and concepts presented in previous chapters. A digital system that meets these needs should basically be able to,

1. Read and write digital, vector topographic map data
2. Symbolize the data on screen or on paper as it appears on published maps, before generalization
3. Create the processing infrastructure needed to (roughly) generalize and begin to evaluate an urban data set, which includes,
   • Accepting or generating basic map controls and target symbology instructions
   • Indexing existing features into partitions
Table 6.1: The Six Packages of the Antelope generalization prototype.

<table>
<thead>
<tr>
<th>Package name</th>
<th>Text description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app</td>
<td>Database and Application methods</td>
</tr>
<tr>
<td>gui</td>
<td>Features relating to the graphical user interface</td>
</tr>
<tr>
<td>buggy_jfc</td>
<td>Core Java Foundation Classes that need to be rewritten</td>
</tr>
<tr>
<td>data_io</td>
<td>Parsing code to read data files in XML format</td>
</tr>
<tr>
<td>feature_classes</td>
<td>A base class and classes for each entity type, such as road</td>
</tr>
<tr>
<td>antelope</td>
<td>The master package that contains all sub packages</td>
</tr>
</tbody>
</table>

- Generalizing linear features needed for partitioning
- Performing structure recognition and cartometric evaluation
- Analyzing partitions for density and homogeneity
- Placing features in target partitions (intelligent selection and more)
- Measuring and reporting changes in the representation.

The cartometric measures would be used during several stages of the characterization, processing, and evaluation of the geometric conditions listed in Section 2.2. However, these measures generally fulfill low-level needs, and require continual refinement and understanding to select meaningful parameters. A description of the available measures and an evaluation of some measures relevant to urban topographic generalization is provided in the next chapter. The focus here is to describe the above needs further to clarify what behavior is expected from the system.

6.3 An Object Oriented Design

An implementation in an object oriented language like Java to store the data by classes and symbolize the data to screen or PostScript by the author would be useful for three reasons. First, newer formats for data dissemination by NMAs seem to be heading towards features that can be modeled as software objects. Second, the LAMPS2 system used for the AGENT project stores features in an object oriented data base. Designing a class hierarchy in Java that is compatible with the Lamps2 system, would allow synergies to take place, in addition to being able to work without software license restrictions when operating off-site. Third, programming a representation in Java would allow the author the ability to add and isolate only those computational methods needed to carry out the objective of partitioning and evaluation experiments needed for this dissertation.

Thus, this section describes some of the important data modeling and handling issues that make up the system prototype. The Java implementation can be found on the CDROM that accompanies this thesis. The prototype design contains 6 packages, listed in Table 6.1. The app contains the GIS engine, database and methods used for experimentation and will be described in some detail. Selected APIs (Application Programming Interfaces) can also be found in the appendix that explain the other packages.

Table 6.2 lists the classes found in the app package. These classes hold the associated data structures used to perform GIS functions. These classes can be understood better when viewed in context of their parent container, the Project class, shown in Figure 6.1. The aim of the prototype is straightforward, namely to fulfill the Basic Requirements from the previous section. Data and Partitioning features are read from files with the XML parser and reader and stored in the Dataset. From there, a UI control can trigger the creation of topology. Topology is kept through the concept
### Table 6.2: The Object Classes in the *antelope.app* Package.

<table>
<thead>
<tr>
<th>Class name</th>
<th>Text description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloner</td>
<td>Makes a deep copy of arbitrarily complex objects.</td>
</tr>
<tr>
<td>Dataset</td>
<td>Holds all of the cartographic features and geometries and has most of the routines to compute all line intersections for the geometries for when the features are first read into the system and do not have topology defined yet.</td>
</tr>
<tr>
<td>Face</td>
<td>Holds a topological face or partition and can store references to records for that face.</td>
</tr>
<tr>
<td>Face_list</td>
<td>A Container to hold faces or partitions.</td>
</tr>
<tr>
<td>Line_poly_flt</td>
<td>Provides a polyline (line with multiple points) for a topologically based system that has a start node, end node, and intermediate points that do not intersect other polylines, where coordinates use floating point precision (not Double).</td>
</tr>
<tr>
<td>Line_primitives</td>
<td>Holds all of the shared primitives of lines that belong to a topological universe.</td>
</tr>
<tr>
<td>Line_segment</td>
<td>A simple 2-D floating point precision line used to test intersection during topology construction, and as such is temporary in the system.</td>
</tr>
<tr>
<td>Node</td>
<td>A topological node of any incidence.</td>
</tr>
<tr>
<td>Node_list</td>
<td>Contains and performs methods on nodes during topology building.</td>
</tr>
<tr>
<td>Partition_list</td>
<td>A master container to hold partitions and index features for those partitions.</td>
</tr>
<tr>
<td>Project</td>
<td>Holds a project, which includes a dataset and structures for storing and managing topology.</td>
</tr>
<tr>
<td>Transform</td>
<td>A class to hold cartographic transformation and projection methods.</td>
</tr>
</tbody>
</table>
of shared primitives. That is, if nodes are within a certain snap distance, they are stored as the same node. Line segments are tested for intersection. Any line segments that intersect in the same topology layer must do so at nodes, so if a node doesn’t already exist, it is created, and the feature is split. The end result contains a list of shared line and node primitives. From these, a list of topological faces are also generated. Any time a feature changes geometry, the topology is updated, due to this node–line primitive structuring. Computing topological faces requires traversing the nodes and shared line segments, looking for the shortest cycles, in arbitrarily either the right hand direction or the left hand direction. Any time a cycle is found, a lookup must be performed to see if that cycle already exists in some form, though it could start at any node. If it does not, it is a new face, and stored in a container. Instead of using relatively expensive Math.sin() calls, the theta function in (Sedgewick, 1992) can be used to determine which segment extending from the tested end node has the shortest angle.

In order to symbolize the data, each feature has its own symbolization instructions embedded in the class, that are invoked with the identical method call. Namely, the graphic user interface (GUI) paint method iterates through each feature class in the features table, and again through each feature in that feature class (to respect stacking order). For each feature, the feature’s paint(Graphics2D g) method is called, where the graphics context g is the open graphics window in the user interface. All features extend a super class (a Java Abstract class) called Geo feature. Iterating over any container object, and retrieving the contents means retrieving the contents and casting them as a Geo feature. Because partitions can have their own topological layer, they need their own copy of the topological shared primitives. Node incidences can change and so on, so the base topology should not change when experiments with partitions are conducted. In order to populate the partitions, a container containing references only to features to be indexed is created. Then, for all partitions, each feature in the container is checked with a polygon in polygon test.

The largest performance bottleneck lies clearly in the line intersection routines, used to compute topology, needed for both dynamic topology and partitioning. \( O(N\log N) \) plane sweep algorithms were tested and deemed much faster, though a Java implementation derived from (Laszlo, 1996) had too many special simplifying assumptions about the possible intersections encountered (which did occur in about 10% of all possible intersections tried on a 10,000+ segment USGS data set). The geometric library LEDA (Mehlhorn & Näher, 1994; Mehlhorn & Näher, 1995; Schirra, 1997) has robust line intersection code which could be rewritten to work in a Java environment. Otherwise, a brute force \( O(N^2) \) algorithm can work for small \( N \).

Even though the functionality of the entire system has been broken down into packages and classes, it seems the coupling is still high and the number of dependencies between classes monolithic. In general, heuristics for reducing dependencies are not well understood, though it is said using package diagrams aids this process (Fowler & Scott, 1997).

### 6.4 Interoperability: Data and Rules Schema

Getting data into and out of the system can be facilitated by providing the data in XML format using one of the widely available XML parsers. XML, or Extensible Markup Language, is a language designed to facilitate data transmission between disparate computer systems. It allows the designer to specify a data schema in a generic definition, and to share this schema across the Internet. The first kinds of schemas in XML were called DTDs, or data type definitions. XML currently provides a fuller schema language which is a functional superset of XML DTDs, though DTDs were used for this study. There are freely available programs by major vendors to allow one to read a datafile which conforms to the specified DTD, needing only to write a minimal amount of parsing code. Most of the parsing work is done by the freely available software. This section introduces the DTDs used for the prototype. IBM’s xml4j.jar was used as the parsing engine.
Figure 6.1: The Project class contains most of the subclasses and structures responsible for delegating GIS, partitioning, and generalization routines. Note that separate geometries are kept for source and target data.
This section shows the DTDs used for the dissertation prototype. The first DTD, \texttt{project.dtd}, contains the strict definition of those elements which are useful to formulate a GIS project. In this case, key value pairs, such as for storing scale and geographic extent information, the specific XML files which contain the raw line data, and the stacking order in which to draw the geographic features on screen. Sample XML data files are included as examples in Appendix C.

The next DTD, \texttt{geodatln.dtd}, is the schema of a cartographic line object. It also has a section for key value pairs for storing any attribute information and a section for the data points. The attributes are bundled with the line object because some systems reusing this data might want to draw or analyse the data as it is read from a network, where some objects will be processed before all data is transmitted. This factor is important for transmission over the Internet, as well as for the smooth integration into the Object Oriented reference prototype.

The last DTD, \texttt{part_spec.dtd}, allows one to specify the features that will be used for the partitioning portion of this dissertation. Essentially, for each partitioning level, the IDs of the features needed for partitioning are specified. For the purposes of this research, they were acquired through automated scripts, extracting all linear features that should form the partitioning features. Ideally, these features should be generated from observations from the database, such as all linear features of a particular hierarchy.

A weakness of the current prototype data exchange is that some attributes are expected to be present in the data, yet they have not been hard-coded into the DTD. This weak-typing feature is useful for rapid development, as features can be added or removed to the XML data files without recompiling parsing code. However, industrial strength software should be modified to be more robust. Once data is determined to be essential, it should be mandatory in the DTD. Creating generic DTDs or XML schemas broadly applicable for geodata is an open problem.

\begin{verbatim}
<?xml encoding="US-ASCII"?>
<!ELEMENT document (parameters, data_files, stacking_order)>
<!ELEMENT parameters (key, value)+>
<!ELEMENT key (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT data_files (xml_files, ser_files)>
<!ELEMENT xml_files (xml_geodatln)*>
<!ELEMENT ser_files (ser_feature_class)>
<!ELEMENT xml_geodatln (#PCDATA)>
<!ELEMENT ser_feature_class (#PCDATA)>
<!ELEMENT stacking_order (class)+>
<!ELEMENT class (#PCDATA)>
\end{verbatim}

\texttt{project.dtd}
The USGS, for instance, has released large data sets to the public in the Spatial Data Transfer Standard (SDTS), digital line graph (DLG-3) format. This data is well suited for some types of automated generalization research due to the varying nature of the North American geography and landscape covered in the U. S. and the depth of coding detail in the schema. Similarly, data sets released by the IGN provide a chance to analyze geography and generalization issues more prevalent in Europe, such as a higher urban density of buildings and narrower streets.

Each data set would then be written to a separate XML file for each unique entity type that exists in the source schema. For example, Table 6.3 shows the unique road entity types in an example dataset in USGS SDTS DLG-3 format. For each record, a matching file was created that contained all features of that entity type, such as one XML file with all Class 3 roads.

Other themes, modeled by the USGS in a series of digital files for each theme are hydrography, boundaries, man-made structures, contours, roads, railroads, survey points, and vegetation. These files were found available free of charge on the Internet in SDTS DLG-3 format. (http://www.usgs.gov/).

### 6.5 Data Structure-based Symbolization Problems

The implementation studies of this thesis, described in Chapter 7, required the analysis of digital data sets released from national mapping agencies such as the USGS or the IGN. Writing scripts to convert the data into XML format was usually straightforward and enabled the prototype to draw wire frame vector data. In general, full symbolization with symbols and symbol widths that are to scale is important for the study of generalization for two reasons. The first is to properly identify the visual or computational conflicts such as congestion, coalescence, and imperceptibility. The second reasons is that the evaluation of the success of generalization operations is only possible with full symbolization, due to the graphical nature of symbol feature interaction. However, instructions or
Table 6.3: Unique road entities found in the Austin East, 1:24,000 SDTS DLG-3 data set after the importation to ARC/INFO.

<table>
<thead>
<tr>
<th>ENTITY_LABEL</th>
<th>Text description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1700005</td>
<td>Cul-de-sac</td>
</tr>
<tr>
<td>1700201</td>
<td>Primary Route, class 1, symbol undivided</td>
</tr>
<tr>
<td>1700202</td>
<td>Primary Route, class 1, symbol divided by</td>
</tr>
<tr>
<td>1700203</td>
<td>Primary Route, class 1, divided, lanes separated</td>
</tr>
<tr>
<td>1700204</td>
<td>Primary Route, class 1, one-way, other than</td>
</tr>
<tr>
<td>1700205</td>
<td>Secondary Route, class 2, symbol undivided</td>
</tr>
<tr>
<td>1700206</td>
<td>Secondary Route, class 2, symbol divided by</td>
</tr>
<tr>
<td>1700209</td>
<td>Road or street, class 3</td>
</tr>
<tr>
<td>1700210</td>
<td>Road or street, class 4</td>
</tr>
<tr>
<td>1700211</td>
<td>Trail, class 5, other than four-wheel drive</td>
</tr>
<tr>
<td>1700213</td>
<td>Footbridge</td>
</tr>
<tr>
<td>1700215</td>
<td>Perimeter of parking area</td>
</tr>
<tr>
<td>1700217</td>
<td>Road or street, class 3, symbol divided by</td>
</tr>
<tr>
<td>1700222</td>
<td>Road in transition</td>
</tr>
<tr>
<td>1700402</td>
<td>Cloverleaf or interchange</td>
</tr>
<tr>
<td>1700405</td>
<td>Nonstandard section of road</td>
</tr>
</tbody>
</table>

mechanisms to accurately symbolize all of the data as originally published on hardcopy maps or even to the screen do not seem to be available (Brazile, 1999). Therefore, symbology instructions had to be reverse engineered, by referring to already published copies, consulting published guides in some cases, and asking experts in others. In the symbology section of the implementation, it was discovered that some items in the schema require differing symbology and some entity types are dependent on the underlying data structure.

### 6.5.1 Stacking Order

The first problem was to determine the appropriate stacking order of features. Experimentation and cross-checking the published sheets revealed a solution. Drawing in order, where lower numbered features first, and subsequent features are drawn on top of previous ones,

1. urban and wooded area tints
2. water areas
3. contour lines
4. water outlines
5. lower and higher class roads, railroads
6. building areas
7. airport runways
8. power stations
9. substation power lines.
6.5.2 Double-lined Road Casing Intersections

The second problem is the intersections of roads that are represented as double-line casings. Figure 6.2 demonstrates the default behavior, left, when linear are merely symbolized as linear features, and the desired behavior, right. Figure 6.3 demonstrates the problem further. Examples of this situation have also been found on maps of other scales and purposes, such as the Kümmerly+Frey 1:23,000,000 Weltkarte. On the published Kümmerly+Frey 1:5,000,000 map of Europe, there seems to be a possible inconsistency near Halmstad, Sweden, Figure 6.4, which would be avoided if an automated system implemented the desired symbolization in Figure 6.2, right.

The road casing intersection problem could be solved with a restructuring of the data that allows a recursive traversal of the road network. The algorithm begins by selecting all roads to be symbolized and considering them at once. Starting at any road, find the perpendicular to the first road segment, and with a vector in the perpendicular direction with a magnitude of one half the desired symbolized casing width from the centerline, follow the road cluster along the outside cluster, drawing a casing line as the roads are traversed, until the perpendicular of the starting point in the negative vector direction is reached (the beginning). Circular arcs may be used on the convex side of corners to connect the line casings. Then for those situations where interior partitions were created, a mechanism would be needed to visit every line, and on either side of the line, determine if a symbol were present. If a symbol were missing, traverse the interior partition, until all interiors were processed. This algorithm would also need a mechanism to determine when the symbolized casing line self-intersected during graph traversal, and those intersections eliminated, which would occur when two cartographic objects form a steep angle.

In essence, after considering these kinds of algorithms, it seems performing a buffer directly with a parameter width, again, one half the casing symbol width, would achieve the same objective. This buffering was performed and evaluated, but not integrated into the prototype. It shows some promise, though it does add a complicated step in merely symbolizing the data. A comparable solution using road buffering and the medial skeleton to expand centerlines was proposed as an extension to (Christensen, 1996) by (Christensen, 1999b).

6.5.3 A Unique Entity Type Can Require More Than One Symbology

The third problem became obvious upon viewing the resultant images, partially shown in Figure 6.3. Some of the entity labels did not map one-to-one to a specific symbology type and it is cartographically unacceptable to force one symbology for these entities. For instance, entity type, 1700222 – a road in transition, was symbolized on the published map as Class 1, with a highway type symbol, elsewhere as Class 2, with a boulevard type symbol, and sometimes as a dashed trail type symbol. The symbology instructions written for this prototype never succeeded in characterizing those segments correctly in all cases, though this fault was not deemed crucial to perform the generalization experiments. However, a consideration was made on computing the most predominant type the transitional segment represented, via some kind of interpolation, but those ideas were not implemented and might not be sufficiently accurate if they were. Without that ability, the only correct way to symbolize these features seems to be to mark them explicitly as a special case and symbolize each manually.

6.5.4 Dash Spacing of Separated, Topological Line Segments Produces Poor Cartographic Results

The fourth problem was that certain entities represented with dashed symbols were not symbolized correctly. This is due to the cartographic object, for instance a highway, being represented by disjoint line segments (see Figure 6.5; problem appears also in Figure 6.8). Based on the simple super class,
Figure 6.2: Centerlines in the data base symbolized as parallel line casings intersect at road junctions, A. The desired impression for these casings is B.

Figure 6.3: Postscript image generated from reverse engineered symbology instructions to look like the corresponding, published USGS topographic map. This area is the main campus of the University of Texas at Austin.
Figure 6.4: A double-line casing road with no area fill intersects a higher priority double line casing highway with no area fill, redrawn from Map of Europe, 1:5,000,000, Copyright Kümmerly+Frey, Bern, Switzerland. Yet the lower class road is shown in higher priority figure ground. At this scale connectivity should be indicated as in Figure 6.2, part B, regardless of road class, as was performed elsewhere in this Map of Europe. Digital systems would, in general, need the ability to handle this case.

Figure 6.5: The dashing pattern required by some entities such as narrow highways was interrupted due the segmented nature of the underlying object. The desired pattern is shown, B.

The symbolization routines specified in the prototype were not coded to handle the dashed patterns consistently. If the data to be symbolized were restructured via a connected network line made up of several topologically distinct segments, it is likely that the alternating dashed pattern would be correctly rendered.

6.5.5 The Contiguity Problem of Road Segments

The fifth problem arises from how the data is stored, by road segment, which has been split for topological reasons. The problem stems from these line segments being symbolized separately, when instead a single polyline line or network line grouping segments is desired so that spacing or contiguity problems are handled correctly. Figure 6.6 A. shows the outcome of symbolizing each road segment separately; Figure 6.6 B. shows the desired behavior. The problem can also be viewed in Figure 6.8, in the source map panel, noting the highway in the lower-center of figure is interrupted several times.

6.5.6 The Overlapping of Casings on Forked Roads

The sixth problem also arises from how the data is stored, by topological road segment. Taking a highway symbolized with a red interior and black casing, the following problem arises. Figure 6.7 A. shows that a road segment that forks off from a main road, using the same symbology, interrupts the casing. The desired representation is to show the roads are connected, Figure 6.7 B. right.
Chapter 6

6.5.7 Interoperability of Symbology Instructions

Cranston et al. have demonstrated an intelligent way to transfer DLG data to a common interface (as common objects) and back to a native format again using the OpenGIS paradigm (Cranston et al., 1999). However, even with systems implementing this new interoperating paradigm, the needed ability to symbolize the data is missing from any interoperability specifications (Doyle et al., 1999). Doyle et al. further note in their OpenGIS implementation, OpenMAP, that the graphical appearance didn’t meet the requirements imposed on cadastral GIS applications in Germany, due to the omission of interchangeable portrayal specifications (Doyle et al., 1999). The Swiss INTERLIS geodata specification format, in version 2, includes a mechanism to address this deficiency, which would make it easier to exchange graphic views on the data (Keller & Thalmann, 1999), thereby fulfilling a need to be able to share instructions on the usage of data as well as the data itself.

6.5.8 Object Oriented, Java 2, and System Specific Issues

The representation produced from the Java implementation, Figure 6.8, approximated the published map, Figure 4.2, sufficiently closely for the purposes of this study. Surprisingly, the performance of the display paint method was quite responsive. The ability to encapsulate the draw methods in the feature class for any arbitrary feature class, whether they were roads, buildings, lines or areas was very powerful, produced a close approximation to the desired result, and was straightforward to implement. For major highways, for instance, a black line with width five pixels wide could be drawn, and on top of that, a red line with width three pixels overlaid, giving the impression of a black road with a red fill. However, for a production quality cartographic representation, each symbology problem mentioned in Section 6.6 would need to be addressed.

Another problem stems from the implementation of the underlying, system dependent image buffering. In this prototype, two alternatives were tried. In one panel (the source panel), an operating system image buffer was used to store the map representation. The benefit is that the scrolling is immediate, as the entire image is buffered into memory. The drawback is the zoom factor could only
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

work for very small magnitudes, because large zoom factors meant creating memory buffers that exceeded system capabilities. This is, of course, assuming a quad-tree or uniform grid style of tiling the symbolized image is not available. In the other panel (the target panel), the features were all redrawn with every update event. This method was slower on scrolling, as each scroll event caused the entire panel to be redrawn. However, an arbitrary zoom factor could be entered. Special system engineering including image tiling would be required to have both fast scrolling response and an arbitrary zooming capability.

Computing a Correct On-screen Scaling

Remaining problems were found with generating the representations on screen, which are similar to those experienced by (Ditz, 1997). Namely, the pixel is the smallest addressable size, yet a pixel is already approximately .01” on a 17” monitor at 1024x768. Many line weights in the USGS specifications, intended for printing to paper from film, are half that size or less. Additionally, line symbols that must show variation, such as black outlines with a red interior casing, need at least three pixels – two for the black outer pixels and one for the inner red – an exaggeration of at least 50%. An on-screen specification will probably be necessary for future digital products.

Another concern in the Java implementation was how to compute the line widths dynamically, based on the current resolution of the screen and the specified scale factor. There is a function call in Java that provides a resolution, though it was not the correct one because it could not know the size of the monitor.

Problems with Serialization for the 2D Classes

Another problem with Java was that the object class, BasicStroke, which implements different line styles cannot be serialized, or written directly to a file to be read in quickly later, due to potential security reasons because it has private variables. However, each feature class had a draw method and instantiated an object of BasicStroke to draw a symbolized line. That means that the mechanisms employed to save the objects persistently all broke when the BasicStroke functionality was used. That further meant all the source data, in ARC/INFO generated, simple UNGENERATE format data had to be read in, parsed, projected, and symbolized from scratch each time the program was run, slowing the time the program was started to the time all features had been drawn from under one minute with serialization to approximately seven minutes without serialization. A quick solution to the problem was to rewrite the BasicStroke module to allow it to be serialized – a rather trivial effort, though it introduces undesired redundancies into the base Java platform. This re-write was introduced into the buggy_jfc package of the study prototype.

Exploring Object Oriented Design Patterns to Overcome Symbology Problems

In order to overcome the sixth problem, from Section 6.5.6, it seems necessary to draw all the black lines for all highways that are connected, and then draw the red interior lines. This can be accomplished but no longer allows the simplistic design pattern of iterating over all objects in the database and calling their draw method. A specific and separate class or engine would need to be created which can draw each feature class in the table, based on multiple iterations over all features in that class. Likewise, this engine should contain a structure to make explicit the stacking order of feature classes.

Carrying the concept further and having a single class contain all of the possible representations, where a road object’s draw method would merely call a static instance of a drawing engine class with a parameter for that road type can not yet be advised. This idea was considered, in part because it seems it would make central all issues dealing with representation, which would be valuable during
debugging and inheritance. However, it would also seem to violate the spirit of naive object-oriented (OO) design by making the data objects not truly self-contained. In the software engineering field, commonly recurring alternatives to self-contained objects have been found in large, production OO systems. These alternatives have been studied and transcribed into generic patterns (Gamma et al., 1995). Generally OO design patterns, with the exception of the iterator pattern, were not integrated into this prototype design. However, exploring the use of these patterns in GIS implementations seems promising and may already be in use in proprietary systems. Designers should note some of the drawbacks to using patterns, such that they are also more complicated, require a thorough understanding of the complex interaction behavior of geographic data, and seem to not only be programming language dependent, but efficiency dependent on the underlying machine or virtual machine implementation. The reference material available to the author at the time of this experiment did not provide sufficient examples for Java, although they can now be found in the literature (Grand, 1998).

6.6 Conclusion

A software system design was presented that bridges the gap between the theory presented in the framework and some of the important implementation details necessary to create a working sys-
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

tem. This design was implemented in Java, an object-oriented programming language, and basic cartographic functionality was featured in this chapter. The visual results from this prototype design indicate the overall methodology served to represent a majority of the features found on the published map reasonably well. However, some non-trivial symbology issues still remained, such as rendering non-overlapping double line streets from lower class road centerlines found in the data base and also creating randomized icon-based area-fills that remain analogous to the original. However, for measuring and evaluation purposes, the implemented system yields sufficient information for determining symbology space usage. An intelligent model for storing, transferring, and symbolizing vector topographic data in an object oriented model is still an open (and large) problem.

As for the suitability of this system design and data infrastructure to investigate generalization, there are many items to consider. For the USGS data sets, the data were freely available, and many reference maps and some separate symbology instructions could be found either digitally or inexpensively in paper format. The USGS data provided some useful case studies such as in situations where building footprints were shown, tightly clustered, and aligned in identifiable patterns, as in Figure 6.8. Otherwise, the geography and map specifications involved were predominantly North American, such as streets that form an almost regular grid pattern and urban areas that are merely symbolized with a shade rather than retaining all buildings large enough to be shown, even at 1:24,000. At 1:100,000, there are generally no more than five buildings total, yet the entire street network seems to be retained. In the IGN datasets, unlike the USGS datasets, buildings were shown at smaller scales. The IGN data in general, e.g. see Figures 7.2 and 7.3, contain much smaller area features than would be shown on similar map sheets of US areas. Data from other regions outside of the US and Europe were digitized and brought into the system, like the area in New Delhi, from Figure 3.5. Other than specific symbology differences, the data can be structured similarly, allowing this prototype system to use data from different geographic areas. Data covering differing non-US and European regions in digital format at 1:50,000 and 1:100,000 scales exists, according to (Beaulieu & Dohmann, 1997). However, this data could not be acquired for use in this study because it is restricted to U. S. Dept. of Defense personnel. Otherwise, this data might have proven interesting to complement the USGS, IGN, and manually digitized data sets.

With such a system in place, incorporating generalization concepts into a basic digital cartographic system is possible. Utilizing the database schema, partitions can be created from the topographic data from the linear features found in the database. Primary partitions, which characterize major urban tracts as well as the indexing, secondary partitions are formed from topological relations using linear features as boundaries. This partitioning functionality is demonstrated next, as well as the ability to perform measures on partitions.
Chapter 7

Testing the Integration of Feature-based Partitioning, Measures, and Automated Evaluation

7.1 Introduction

The overall aim of this research has been to show how to build the procedural infrastructure that will support the generalization process for topographic style vector data sets. Specifically, this research sought to cover the aspects of decomposing the complete process as it is currently understood, describing basic structure recognition, and proposing ways to support run time evaluation. The route to implement these goals is two-fold. The first aspect is to indicate how to construct the processing infrastructure via partitioning, which when used as described can intrinsically preserve important cartographic characteristics during the scale change or information refinement. The second aspect is to combine this process decomposition with both the starting state of the source map (e.g. the digital representation) as described in the previous chapter and with measures on the newly decomposed partitions and indexed elements.

After successfully partitioning the map into processing units based on the assumptions presented in the previous chapters, it is possible to perform the various characterization, transformation, and evaluation methods that are needed to create a generalized map. This chapter describes how those infrastructure building methods and measures can be combined to assist in the generalization evaluation process and illustrates this relationship with implementation-based examples.

7.2 The Availability and Selection of Measures

7.2.1 When to Apply Measures

Determining the measures that are needed to build a generalized topographic map, and which would thus populate the quality measure array from Chapter 4, is a challenging problem to the GIS research community. Weibel and Dutton point out that cartographic data tend not to be richly structured and that parts of features are rarely coded explicitly (Weibel & Dutton, 1999). Measures are needed to identify structure that is to be preserved (or discarded) during generalization. Even though some data sets are richly encoded, they still may not contain sufficient implicitly coded structure for some of the algorithms. There are multiple stages where measures are needed in generalization, namely
Table 7.1: Categories of geometric measures

<table>
<thead>
<tr>
<th>Proximity</th>
<th>Size</th>
<th>Convexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concavity</td>
<td>Sinuosity</td>
<td>Elongation</td>
</tr>
<tr>
<td>Compactness</td>
<td>Singularity</td>
<td>Regularity</td>
</tr>
<tr>
<td>Repetition</td>
<td>Symmetry</td>
<td>Homogeneity</td>
</tr>
<tr>
<td>Orthogonality</td>
<td>Granularity of detail</td>
<td>Feature removal</td>
</tr>
<tr>
<td>Density</td>
<td>Detail Proximity</td>
<td></td>
</tr>
</tbody>
</table>

during

1. the initial characterization of the general aspect of all features in the source map
2. cartographic conflict detection (cf. Section 2.2)
3. characterization of features in the neighborhood of the conflict
4. run-time evaluation
5. post-evaluation to compare transformed features and areas to original and specified conditions.

Although these stages can be enumerated distinctly, in fact they may overlap, as will the parts of the measure implementations. So, a categorization and description of the available measures is presented next.

7.2.2 What Kinds of Measures Exist

The classification of measures can be grouped roughly according to the classification of constraints encountered during automated generalization. These constraints have been proposed in (Weibel, 1997b) and by (Lagrange, 1997; Ruas, 1998c; Weibel & Dutton, 1998). They are (geo)metric, topologic, semantic and aesthetic constraints. Each of these constraints can then be used to inspire and classify, in turn, geometric, topologic, semantic, and aesthetic measure classes. Another measure class can be created to test map purpose fulfillment.

Inside measure classifications are categories of specific measures for cartographic features. Researchers at the French National Geographic Institute (IGN) have enumerated several measures using a constraint-based classification (Mustière, 1995; Hangouët, 1995). The AGENT consortium has also compiled and developed detailed measure categories and measures that could be used in the generalization process (AGENT Consortium, 1997; AGENT Consortium, 1999a; AGENT Consortium, 1999b). Geometric measure categories are numerous and topological measures are fairly complete, while semantic and aesthetic measure categories are relatively unpopulated. In general the availability of geometric measures follows Table 7.1. For topologic measures there are standard contained-by, connected-to and next-to measures. These general topological relationships in $\mathbb{R}^2$ have been enumerated in (Tryfona & Egenhofer, 1997) and are presented in Table 7.2. Specifically for linear features there is intersection and self-intersection plus graph theoretic measures, such as the depth of a network, graph density, and graph connectivity.

Aesthetics measures are difficult to formalize. Their role is to monitor and facilitate efficient map communication, seeking to combine design aspects for efficient communication as well as possible aesthetic or stylistic treatment. This measure category includes readability measures, pattern preservation measures for shape, size, and distribution, and consistency measures. The pattern preservation measures should primarily indicate overall similarities of distributions of measures comparing source material and graphic results.
Useful semantic measures are also difficult to acquire, but could be based on rules similar to those found in cartographic rule bases. Other semantic measures could be developed on the partitions themselves, deriving semantic information from statistical, metric, and topological information. Generalization systems of the future will need well-defined semantics that may not exist by default as explicit relationships in the source databases. Resolving implicit relationships in databases for these new semantics can be a major source of error (Harvey et al., 1998). Thus, testing for semantic errors when combined feature classes are to be shown in the final map and building measures to help insure intelligent use becomes important.

Clearly, some measures are meant to be applied to singular features such as the minimum length of parts to a building features, while others for collections of features under various contexts or views, such as the overall building density to a partition, which in a sense is a measure of measures, as are other statistics-based measures, such as summations or histograms, mean, standard deviation and coefficients of variance. So, with the availability of so many measures, which measures are really needed to produce meaningful, generalized maps? To answer this question requires a review of the features and processes that are involved during generalization and experimental trials verifying the assertions made here.

### 7.3 What Should Be Measured

#### 7.3.1 Linear Features and the Network Graph

The first time generalization algorithms are employed, before the target partitions are formed, occurs during the generalization of the linear feature network. Road feature selection and displacement is a major topic of research and requires more study before it can be adequately formalized. Essentially, the general road pattern should be altered as little as possible with respect to the final scale and
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

display conditions. Railroads should move almost not at all, and rivers can be be displaced or collapsed, based on sensible rules and institutional guidelines. If some of the linear feature network have overlapping symbology, an algorithm in the spirit of (Nickerson, 1988), but handling \( n \) linear features instead of only two at a time is appropriate, assuming the worst case of rows of mostly parallel features do not cascade into each other, iteratively.

When people look at roads on a map, they are accustomed to seeing highways as one feature, the highway, and streets as an entity from the starting point to its ending point which bears the name of the street. However, in a topologically based system, the data are modeled via discrete line segments, which do not match the logical road with the given name. And in fact, there may be multiple roads, such as divided highways, and access or feeder roads. There are data models which group discrete segments together (e.g. routes in the ARC/INFO system), which are used for transportation and network planning. This discretization of the underlying data segments is an issue for generalization as well. An evaluation mechanism is needed to categorize logical features together so that they can be collapsed during transitional scale changes.

Some branches of the transportation network may need to be pruned. Pruning the network represents a frailty of showing maps at successively smaller scales. Senses for navigation and connectivity, as previously stated, are generally overriding goals for the medium to smaller scale topographic maps. Pruning the network directly competes with this goal, though it can be clearly necessary, for instance to remove alleyways or perhaps bike or walking trails that coincide with paved roads of other uses. Smaller feeder or access roads may need to be omitted from the graph as well. Approaches to pruning the network graph for geographic data sets have been proposed (Mackaness, 1995a; Mackaness & Beard, 1993). Though additional work remains, let us assume such an algorithm exists to perform this pruning and so the next step of the automated generalization process is to work with the resulting linear feature generalized map, without any indexed features. Let us also assume that any changes in the linear feature network connectivity were reported in a report, so that a sense of distortion can be documented. That leaves the remaining non-linear features to process.

7.3.2 Partitions

The most useful measures for each partition are density and proximity. We can group partitions of like density together to signal them for similar processing. To group like partitions together, without the benefit of a sophisticated database schema with land-use information, each partition is only analyzed through density. Some partitions will be empty and thus will have no density. These will be ignored. Partitions that have a very high density, over 84\%, can be easily treated by filling in the partition to show it is entirely built up land. Partitions between 0 and 84\% then are the ones that require sophisticated processing. Rural areas, outside the scope of this dissertation, are for now assumed to consist of very sparse partitions for built-up areas, and possibly completely built-up land tracts, say of forest parcels or crops. To determine the built-up areas, the functional and not necessarily official urban boundary needs to be defined. Methods to determine urban boundaries have been described in (Averack & Goodchild, 1984).

If the scale change is large, for instance 1:25,000 to 1:200,000, and no intermediate maps are produced, then the amount of generalization work is relatively small, once the road network is adequately generalized (and refined). For the far majority of indexed features, only certain predictable impressions are needed, such as squared buildings spaced adequately apart or, for partitions which have a very high density, a completely filled in area bounded by the linear feature network (which may not be symbolized except through the swath it weaves through the ink of completely built-up areas).

If the scale change is small, then the amount of generalization work is relatively large. This is because fine adjustments are needed in an environment where the available space has decreased, but the features to be shown remain fairly constant. During these smaller scale changes, minor modi-
Figure 7.1: Examples of safe-regions, from (Peng, 1997). Left. The building object in the shaded region can be displaced to another position still enclosed in the shaded area and still maintain neighborhood relationships. Right. Similarly, the building to the right has its own allowed region, the shaded area. (Fei, 1996) cautions that the emphasis should be placed on to where the building should be displaced after symbology expansion and not in which area could the building be displaced, ignoring symbology expansion. The safe-region measure can still be useful when using a triangulation of symbolized features, addressing this problem.

fications to features can be expected. The ability for the system to self-evaluate minor alterations is then required, because many alterations will be needed. Using the partitions as a receptacle for density calculations, the current density of the state of processing can be identified and matched to the original density and the allowed target density, for bounds checking. However, performing localized evaluation other than relying on density and proximity as measures, becomes a challenge. Alternative measures for localized extents rely on the creation of subsets of features used for analysis. Clustering facilitates the detection of homogenous characteristics, or the measures found in Table 7.1 (Regnauld, 1996). Other localized measures exist to facilitate which regions are available for displacement during generalization. (Peng, 1997; Peng et al., 1995) introduces the concept of safe-regions, which are similar to other processing units when using triangulation based infrastructure methods. Figure 7.1 shows the available safe-region areas for buildings, if they need to be displaced after a symbol modification. Presumably, if features are combined, their respective safe areas can be combined as well to preserve localized neighboring criteria. Regardless, the features in each partition need to be generalized in a manner appropriate to the target scale.

7.3.3 Individual Features

Individual features are measured first for their size. If a feature does not meet the minimum criteria for display at a particular scale and given a particular symbology, it will either be omitted, grouped with other features through typification, or enlarged directly, if it is important enough. Linear features that were generalized first may undergo simplification and smoothing. Features inside partitions are also measured to determine if the geometric attributes of the feature are too complicated. The feature shape, if the feature is to be retained, can be simplified—8 sides to a building at a source scale can become 6 or 4, and still the impression of a building is communicated at the target scale. Feature symbology can change and this can be controlled by measures as well. Contrast density can also be calculated, separate from geometry density, and is applied to check features of a particular size such as large shopping mall. When feature size exceeds a certain area, the symbology contrast can be reduced.
7.4 What should be Evaluated

The items to be evaluated during pre-processing and descriptive stages of generalization are also the items that need to be evaluated while undergoing generalization transformations. The following basic list describes the features that should be evaluated.

- **Evaluation is needed to generalize the linear feature network.**
  A technique described in a Swiss manual to manually generalize the linear features, mostly the street network, requires selecting the roads that will be symbolized at the fullest symbol width, fill them in from casing to casing with ink, and then with a medium line width, draw the outline of the inked casings (Spiess, 1990). Remove the ink and one is left with the outlines of the roads appropriate for the target width, for small scale changes. The buildings and indexed features in these partitions are generalized and the lowest priority roads are only shown indirectly, by being swathes in a partition, separating buildings. This process is highlighted in Section 2.2, Figures 2.4 to 2.7.

- **Evaluation will be needed to simplify buildings and area parcels not tied to man-made linear features so that the new shape resembles the original shape.**
  Examples of this process are given in (Swiss Society of Cartography, 1977; Parantainen, 1997).

- **Evaluation is desired to detect clusters of indexed features that are not related to partitions.**
  These feature clusters can be determined via techniques proposed in (Regnauld, 1996; Zahn, 1971).

- **Evaluation is desired to determine if and how consistently partition densities and feature proximity were preserved.**
  If the consistency of partition densities is not adequate, then a re-processing effort may be needed. Re-processing could also be needed if target feature proximities do not match the source, or are not consistent among themselves.

- **Evaluating quality is facilitated when the amount of liberty taken to transform the linear feature network is quantified and when the treatment consistency of all partitions can be formulated.**
  This statement should apply to all features in the features network, all features indexed in partitions, and all clusters that were identified and measured, after transformations.

To demonstrate that partitioning as described in previous Chapters, can be achieved and that partitioning provides a framework from which to base measures on, the prototype system was used on various National Mapping agency datasets. Figure 7.2 demonstrates the primary partitioning facility in the prototype system. In this figure, a primary partition is identified in the source data. The graphic indicates a realistic scale change and what the partition might look like at the target scale. The primary partitioning emphasizes the general pattern that will be implied from the stronger visual features that form the linear features network. Figure 7.3 shows a secondary partitioning on the source data set. In this highlighted partition, the density of indexed features relative to the size of the partition is computed in the prototype to be 18%. These two figures indicate that the partitioning concepts presented in previous chapters were achievable for a test dataset in use by map researchers. Figure 7.4 represents an attempt to apply the partitioning as a generalization guidance and evaluation tool on a dataset outside Western Europe and North America. In this case, the dataset represent an area near New Delhi, India. The partitioning functioned and the non-linear features for the highlighted partition were indexed. In all three cases, partitioning is demonstrated to serve to build the infrastructure as described in Chapters 4 and 5.
Figure 7.2: Primary partition for both source and target map.
Figure 7.3: A secondary partition in Laverune, France with 18 indexed features (buildings) and a symbol density of 18%.

Figure 7.4: A secondary partition in Asia (New Delhi, India).
7.5 Conclusion

In Figure 6.8, and the published maps for the area shown in Figures 7.2, the road network seems to be the dominant feature class. Observations, not shown, were made on published maps of the same areas. It was noticed that the cartographers for the USGS 1:100,000 map covering Austin East, TX, displaced smaller, low order roads, but only to a minor degree, and did not seem to displace high order roads such as highways. This was determined by overlaying the digital data of both scales and visually noting the difference. In some cases, the road network graph was simplified by omitting some smaller roads. In general, there were only a few situations where the 1:100,000 road network was altered, when more displacement had been expected by the author. This means that network generalization is likely to require only subtle changes in scale as studied in this thesis. Another matter to consider was that the 1:100,000 map was not derived from the 1:24,000 map. If this data were used in other ongoing work in calibration, a mechanism for matching would have to be implemented, which might be non-trivial. In some cases, this calibration mechanism would be needed also for map updating, discussed in the next chapter. Insight into the designing process did become clearer. The configuration of features on these real maps are complex and for algorithmic and measure testing purposes, a simplification to canned case studies would then be desired to regression test any platform which claimed to holistically solve a generalization situation. Creating artificial experimental data based on these case studies, as well as keeping mind solutions found on published maps, would be useful for identifying calibration thresholds, especially for displacement. However, real data for mapping is ultimately the target of such research and should ultimately be used to establish confidence in future linear feature generalization and displacement methods.

Returning to the question of creating case studies from real data, the basic question arose while using the prototype. What is the smallest set of feature classes that would demonstrate solutions to generalizing a sufficiently complicated environment and is such a question valid. Gower et al. note that the sequence in which processes are performed is an integral part of effective map generalization (Gower et al., 1997), yet too many feature classes would impede obtaining the research objective. It is difficult to commit an answer at this time, yet in the study area again, major roads were the most important structure imposed on the landscape, followed by second class (residential) roads, followed perhaps by water bodies, both lakes and rivers of all sizes, followed by selected buildings. Contours gave a heavy impression on the map due to the brown color, which weighted them more than is necessary, but are not used for topology, and thus not for partitioning. In general, color can have an effect on symbol widths, as pale colors require wider lines and area symbols that might need to be enlarged somewhat compared to equally sized, but darker contrast symbols (Swiss Society of Cartography, 1977). Railroads, transmission lines, and other features do add to the complexity of certain situations, but they could probably be omitted in order to prove if a system could treat cartographic data holistically.

Finally, evaluation and measures that are built into algorithms share much in common. Measures are meant to singularly focus on the processed element at hand, while evaluation implies more a review of the measures that were performed or should have been performed. Thus with this concept of evaluation in mind, this chapter illustrates how, based on the assumptions of generalizing urban topographic maps at intermediate and small scales, partitioning, measures, and self-evaluation can be practically integrated. The prototype results show that partitioning for generalization is possible and thus a subset of the generalization process can be automated, when distinctions are made between generalization and map design. One is not guaranteed with this framework to have a perfectly designed map, but one is likely to get a consistent map. An automated but consistently treated map, one where similar situations are treated with similar solutions, is going to possess aesthetic elements that an automated but inconsistently treated map would not share. Additionally, if human intervention is still required, it can also provide a good first attempt and significantly reduce the amount of time needed to generate a new map series possibly from a single source database, for the
goal of multiple representations.
Chapter 8

Conclusion

8.1 Results

In Section 1.2, the following goals for this research were defined as,

- to assess the potential for automated context-based generalization;
- to propose a strategy for context-based generalization that relies on self-evaluation;
- to determine how to construct the evaluation infrastructure, via prioritized information partitioning and measures;
- to perform implementation studies and interpret other empirical work that gauges the success and scalability of the proposed strategies, some of which have been adopted and integrated by other research and commercial efforts.

The achievement of these goals were met through various research tasks meant to address each topic. One major task of this research was to demonstrate that the very complex task of cartographic generalization could be decomposed into complementary, yet separate steps. Being able to automate generalization is well known to cartographic practitioners as the holy grail to this domain. Indeed some consider generalization to encompass everything that is elusive and subjective about good map making. However, researchers and agencies have made significant efforts to extract production knowledge from domain experts in order to determine what could be automated. Further thought on the subject has been given form and new methods for achieving sub-tasks such as smoothing and displacement. Work for this thesis included decomposing the high level procedures necessary in generalization and integrating these new methods into that framework.

A further task was to show that this decomposition would allow, for certain, well formulated mapping tasks such as topographic mapping in medium to small scales, a path towards automation and integration with existing generalization methods and procedures. The literature is filled with independent methods for executing and testing generalization operators, so a framework that combines these features would begin to permit a holistic, problem-solving approach. This task was met through the design of a framework that integrates the various available approaches. This framework relied, however, on a novel constraining assumption on how and when features are treated.

Yet another task of this research was to provide that structure. A proposed avenue to establish these constraining elements was given through the notion of partitioning features found in the dataset, partially in order to preserve characteristics of the original representation and partially to limit the amount of decisions that must be made. Fundamental questions of partitioning data sets in this
manner were presented that provide the theoretical foundations for partitioning and present some additional questions that might arise from partitioning. One observation from this discussion is that the complement to partitioning, or creating clusters of features through some specific criteria not limited through linear features and integrating their usage with partitioning remains an open problem. For the average case, however, much work can be provided through the partitioning mechanism.

The next task of this research was to consider and design a software system that utilized the proposed concepts. A novel reference implementation was made which bridged the theoretical argumentation of partitioning with the pragmatic nature of generalization by providing a demonstration application or prototype. This application was capable of rendering the problem in a realistic manner, performing rudimentary structure recognition, executing the partitioning concept via generic instructions, and producing scaled representations which could compare the statistics and measures collected between the source and target representations. The prototype was successful in that it showed for datasets in North America, Western Europe and Asia, the partitioning concept for urban topographic maps was feasible and could become a necessary element to automated generalization. Major elements of partitioning have also been shown to be useful for an ongoing, publicly funded production generalization system (AGENT: Lamy et al., 1999). Other aspects of the prototype revealed deficiencies in generic aspects of digital cartographic systems, primarily in the area of symbology computation but also in data exchange and interoperability issues. Further testing of the concepts provided in the prototype to establish additional operational insight of generalization operators remain hinged on the need for formal specifications of cartographic constraints, which are under research in the AGENT project, and also on functional definitions of quality. The various, possible improvements to this direction of research are presented next.

8.2 Suggested Improvements

The concepts and methods shown in this report can be extended in a number of ways to improve the workability, broaden the scope of treatment, and fine tune urban cartographic generalization results. There are interesting possibilities to improve procedures and methods that cover each segment of the generalization process. These have been broken down next into philosophical, strategic or high level, and tactical or low level improvements.

8.2.1 Philosophical – The Turing test for Generalization Systems

British mathematician Alan Turing (1912-1954) once proposed a test to define intelligence. In (Turing, 1950), a person was hooked up via a terminal to a machine and to another human. Only through asking questions, was the person supposed to figure out which counterpart was the human and which was the machine. If the person cannot arrive at a decision within a certain amount of time, such as five minutes, though the time is largely considered unimportant, the machine is intelligent. This test has two components: a philosophical statement – if the machine passes the test, it is intelligent or alternatively stated whatever acts intelligently is intelligent; and a practical component – such a machine will exist in the near future (French, 2000). French points out that the practical component to the Turing test insures the philosophical statement has weight.

This test, the first known of its kind applied to computer systems analysis, itself has limitations. (French, 2000) points out that this challenge tests only human intelligence. That is, a successful machine would answer questions that satisfy the person’s frame of reference, and naturally suggests cultural and other subjective biases. If this argument can be made in the general case of defining and testing intelligence, it can certainly be made for the specific case of defining cartographic intelligence for determining the semantic structure of datasets and computing ideal representations. Therefore,
one improvement in the philosophical argumentation behind cartographic generalization is defining the contextual awareness that makes up the domain, with biases and all. Since the goal of generalization is to produce maps that satisfy human needs, if the machine implements solutions that mimic human procedures, it is a success. Currently, to the author’s knowledge, no such formal definition exists for semantic awareness of features present in a cartographic dataset. Traditional knowledge acquisition in generalization has focused on rule-based or constraint-based knowledge that can be used directly to aid in the use of operators. But the Turing argument applied to generalization suggests a larger definition of semantic awareness or formalization that should extend from the structure recognition phase to the evaluation and testing phase. Establishing a common universe of encapsulated generalization knowledge would help to answer the Turing test for generalization in philosophical and practical ways. Naturally also, to verify the equivalence of the framework and methods proposed in this research with the expected performance of a cartographer using today’s technology (at least up to a certain procedural stage) would validate the utility of proposed automated approaches.

8.2.2 Strategic Directions

Given a satisfactory level of motivation that an automated system can generalize maps according to human needs, how can the approach presented in this thesis be improved? To answer this question re-evaluates the constraining assumptions that were needed to carry out the research. Such assumptions were that the proposed methodology works best for urban topographic areas in scale ranges between 1:15,000 to 1:250,000 and smaller. These assumptions can be discarded for a larger research project in order to produce a convincing description of a truly generic methodology. Additional strategic improvements can be had by refining the goals of the generalization machine. The strategic improvements are:

Achieving Cartographic Realism

Cartographic realism in this case implies a system that passes the Turing test for generalization. Practice should be conducted to build a representation which is indistinguishable from a representation drawn by a trained cartographer (not counting map labels). This can include narrow or canned problems which are generalized by trained cartographers, such as conducted for the OEEPE (Organisation Euopeene d’Etudes Photogrammetriques Experimentales) Working Group on Generalization, and evaluated for the kinds of detailed work necessary to create a cartographic representation (Ruas, 1998a). The most urgent of the feature classes requiring attention are probably road and transportation networks. The scope of operators in this research effort were limited and only assumed to perform certain kinds of transformations, while not focusing on the transformations themselves, only their interactions.

Applying the methodology to a broad range of data

The framework and designs given in this thesis were limited already towards urban topographic data sets. The methodology is largely extendible to other types of datasets due to the overlap in types of representation problems that emerge. However, conclusive work would need to be carried out to make the claim that the methodology presented in this research is broadly applicable outside of urban data sets. The other representations which could be addressed are

- rural areas
- aeronautical charts
- categorical maps
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

- (special) thematic maps.

**Rural areas** have a less complicated transportation pattern and have less features which require intervention. Therefore it is assumed that rural areas do not pose a large problem to generalize. However, some mechanisms to insure consistency can be specified. **Aeronautical charts** contain elements of urban topographic data sets, plus more. Aeronautical charts have even more features which compete for comprehension and imply additional design constraints. They would make a challenging problem to solve in an automated system. **Categorical maps** such as soil maps have other methods for success criteria, often statistically based, than those assumed for urban topographic maps. However, elements of categorical mapping overlap with urban topographic representations, such as when vegetation categories and general topographic features are to be produced for the same representation. Statistical answers to category distribution should not be influenced by design criteria, yet the usual map controls apply when creating any map meant for human interpretation. **(Special) Thematic maps** include those thematic maps that build and show infrastructure that are not part of the better understood transportation networks and urban settlements. These might include virtual or surrogate features meant to convey information which is not to be found on an airphoto. Thematic maps are often produced directly for the final scale, though some amount of generalization is necessary when using composite representations as sources.

**Parallel Processing**

The algorithms that implement some of the generalization operator toolkit generally are robust in that they have survived changes in computer hardware and software. As is often stated in the computer science community, good algorithms on old machines are good algorithms on new machines. The reason for this is that people today generally have access to von Neumann machines. These represent the classic virtual machine where instructions and data are contained in the same memory areas and a single central processing unit extracts instructions from memory, extracts data from memory, performs the instruction, and moves on to the next instruction. The general architectures of the more capable modern systems have for the most part remained similar to this von Neumann virtual machine. However, when one considers that actions can be performed simultaneously, speed ups of great significance are possible. Parallel algorithms for general operations common to GIS have already been proposed and found useful in (Healey et al., 1998). For a role specific to this work, map partitioning creates processing areas which limit the scope of potential generalization decision spaces, such that a divide and conquer approach can be pursued. Parallel treatment of feature-based partitions could introduce time-saving functionality on a machine with parallel processing capabilities.

**8.2.3 Tactical Implementations**

Improvements of a tactical nature suggest ways to make changes to narrowly defined tasks that already rely on formalized methods or regulated procedures to implement, or ways to improve subtasks in a generalization framework, though the distinction between tactical and strategic in generalization can sometimes be narrow. Using the Gaussian smoothing function on a line suggests a tactical improvement over simplifying a line using only the Douglas–Peucker algorithm. Similarly, segmenting a line based on homogenous criteria and tailoring filtering concepts with varying parameter values to each criteria rather than applying a global filtering value for an entire line as it is encoded in the database improves the quality of the generalization (Plazanet, 1996). (Fritsch, 1999) extends this work further by demonstrating how to generalize homogenous segments and retain geometric elements such as general angularity and orientation. There is still room for many improvements in the tactical aspects of generalization. These can be realized in the following ways:
Borrowing Algorithmic Concepts and Approaches From Other Disciplines

One interesting addition to improving the displacement operator was the addition of the Finite Element method for constraint satisfaction problems. The displacement operator was performed using this method brought to the Generalization research community via civil engineering and surveying (Højholt, 1998). The application of this well known method, however, occurred only relatively recently in generalization and suggests there could be more numerical methods or concepts that can aid the implementation of algorithms to support generalization. Additional examples include the use of least squares adjustment to satisfy constraint systems (Harrie, 1998). Also, simulated annealing techniques for optimization of variables to support generalization (Ware & Jones, 1998).

Improving the Knowledge Methods

There are two approaches to creating techniques for guaranteeing functionality of a system. The first way is through black box testing. Black box testing requires generating a large number of tests. If the system passes the tests as defined by the correct answers, then the system is deemed successful. Actually, this form of testing is weak in that exceptions can still exist. This form of testing was used to establish the correctness of the mathematical code on a recent hardware chip (the original Intel Pentium processor). This chip was later revealed to have a bug, or produce an error, because a lookup value that was not tested was incorrect.

The second way, and a more sophisticated approach, requires formalizing that which is to be tested via theorems. These theorems provide rigorous and well defined behavior and interaction of components. Automated theorem provers can even assist in the mathematical proof needed to guarantee true rigor. The very hard part with this approach is in the phase of defining the systems and theorems. It is very easy to get the theorem wrong and prove something rigorously but not at all useful. However, once the correct entities are modeled and theorems defined, the system can be shown correct for all cases. It is currently not uncommon for hardware manufactures to prove the correctness of their mathematical chip-embedded code via theorems to eliminate costly mistakes. Due to the potential rigour that can be realized, a quality evaluation system should take into account the theorems based approach and apply it wherever possible.

Fine tuning usage of satisfactory operators

Many generalization operators use algorithms which are already sufficient to carry out the operation. However, adequate parameters to guide the algorithm are generally needed for each instantiation by most systems today. That this intervention is needed suggests that operator parameters too are probably not rigorous enough. If these could be better stated, then mechanisms to translate these into algorithm parameters automatically can be found, extending ideas published in (Plazanet et al., 1998; Weibel et al., 1995).

Improving the linear feature (e.g. road network) generalization

The linear transportation networks in urban topographic maps form a very significant part of the landscape. After some scale changes, these networks are the only features remaining in the representation. Nickerson proposes a first approach towards automating the generalization of primitives that comprise the linear transportation networks (Nickerson, 1988). However, his approach is simple in that it only deals with the interrelation of two features at a time. Accurate transportation network generalization cannot be realistically bound by such a simplistic assumption. Therefore, sophisticated ways to auto-generalize these primitives are needed. With the advent of such methods, the partitions themselves should become even more useful as the constrained area is more accurate and effective. Improved methods to generalize linear transportation networks are a current topic of
research. Some directions include the usage of, Finite Element Method, and other least squares adjustment related techniques to displace lines (Burghard & Meier, 1997; Højholt, 1998; Harrie, 1999; Sarjakoski & Kilpeläinen, 1999).

8.3 Outlook

In general, there are many technologies and methods that build up Geographic Information Systems. These technologies for data acquisition, dissemination, storage, retrieval, and updating all play a role in determining how to generalize data sets that are becoming available. Thus, a complete system has to be considered when determining the role of generalization. This need for broad understanding can be demonstrated even by isolating the media used to view maps. Today, paper-based products are being replaced by digital representations. The very need for topographic maps with fixed feature sets has been challenged. Historically topographic maps were used to catalog the existence of features, as a kind of implicit database. Modern systems contain digital databases and only require demonstrating the presence of such objects, on demand. On demand images do not necessarily require that all the traditional topographic features be present. So, why go to the trouble that is being done to solve traditional cartographic generalization problems? Changing the problem does not address the fundamental need and it is the author’s view that static representations of maps will be needed even in digital times. If these representations need to show more geographic objects than there is space for, then the problem of generalization remains valid. So, given that the problem really is not going away on its own, how can the total cycle of data acquisition and updating be modernized?

8.3.1 Linkages Between Geoprocessing Techniques

The mid to long-term future of various spatial data handling techniques is converging to enable an integrated approach for mapping, visualization, scientific studies and information storage. Stable and comprehensive techniques for the linkages between the various phases of geodata processing need to be developed. Ultimately, geodata producers and distributors would like to streamline the process from information acquisition to representation and dissemination. In order to do this, three stages that occur during product generation are needed which would allow a rapidly generated vector-based database.

New data are currently acquired via raster-based imaging methods, such as radar, laser altimetry, passive light sensors or photography. However, many geographic and cartographic processes ultimately are modeled and visualized with vector-based representations. Automated feature extraction is the linkage that allows the creation of vector representations from this raster imagery and thus represents the first limiting factor, or dependency in the chain to full geographic data acquisition. The second limiting factor is the process of updating or supplementing already existing data. To integrate multiple data collection phases means propagating vector updates to existing vector data sets. The extent to which this propagation can be done automatically represents the dependency in contributing to the overall automated database generation. The third limiting factor falls under the subject of this thesis, namely, cartographic generalization. Once the data has been acquired via raster methods, converted into vector data, and integrated into a master database, the data will need to be generalized for representations at various scales. Generalization then becomes another dependency in the product generation process that restricts the ability to automatically create data from acquisition to distribution. The solving of each factor is critical to geographic database creation. However, the problem extends further into data sharing.
8.3.2 Interoperability
If the geographic data generation process were completely automated and the three limiting factors were automated, a remaining problem still exists in the distribution chain, namely interoperability. A National Mapping Agency or commercial producer could generate a product, but there would be no guarantee this product could be used efficiently with products generated from other sources. Data interoperability is an integral part of geographic information science because studies require the analysis of many different kinds of variables, which are often products of different sources. The problem aspects of interoperability include incompatibilities in data formats, software products, spatial conception, quality standards, and global models (Věkovskí et al., 1999). For geographic datasets to allow a synergistic combination of information to allow scientific understanding requires a seamless, painless integration of disparate data and information.

8.3.3 Conclusion
So, one possible ideal state of technology would have solved the feature recognition problem, the automated integration problem, the cartographic generalization problem and the resulting interoperability problems. At that stage the geographic information system becomes an almost transparent tool to perform meaningful scientific analysis of geographic processes. However, in reality it is not clear to what extent the human element can be removed from the intervention of manipulating geographic data. Of the major three major stages of production and interoperability, generalization seems to require expert knowledge to the highest degree. In fact, cartographic generalization techniques abound with implicit knowledge and the role of researchers is to convert this into explicit knowledge. Determining this explicit knowledge, even to a limited but still comprehensive degree, would help to enable a fully automated pipeline to support geographic studies. This thesis has tried to contribute to that effort by suggesting ways to automatically build the infrastructure and some necessary methods used to guide generalization. Overall, it has been the author’s goal to contribute to the elimination of generalization from the list of limiting factors to the adoption of GIS as a transparent tool for scientific analysis.
Bibliography


Christensen, Albert. 1999b (September). Widening road centerlines. personal correspondance.


94


Gamma, E., Helm, R., Johnson, R., & Vlissides, J. 1995. Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley.


Parantainen, Leena. 1997. *An Investigation of Forest Generalization by means of a Comparison of the Swiss National Map series, 1:25,000 to 1:100,000 with ideal examples.* M.Phil. thesis, Department of Geography, University of Zurich. in German.


Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization


Weibel, Robert, Keller, Stefan, & Reichenbacher, Tumasch. 1995. Overcoming the Knowledge Acquisition Bottleneck in Map Generalization: The Role of Interactive Systems and Computational Intelligence. Pages 139–154 of: Frank, Andrew, & Kuhn, Werner (eds), Spatial Information Theory – A theoretical basis for GIS (Proceedings of COSIT ’95. Berlin: Springer-Verlag, for Lecture Notes in Computer Science 988.

Appendix


Appendix A

Glossary

Some of the terms and acronyms used in this dissertation are defined below. Many of them are frequently used in the fields of computational geometry, geography, cartography, mathematics, computer science, and spatial analysis. Some terms have been defined, modified, or overloaded to fulfill a role for this research effort.

**airphoto:** A photograph taken, usually perpendicular to the ground surface and from an airplane. Airphotos are used in part as the base material for topographic mapping.

**attribute:** Information associated with an entity. (e.g. A street may be modeled in a data model as an entity. An attribute might be the street name.)

**cartographic procedure:** A step or series of steps a human cartographer would apply to represent an entity on a map.

**Digital Line Graph:** A coding format of the U.S. Geologocal Survey to represent cartographic features in a vector data structure. This format is designed to store the data based on the topological relations of the features.

**entity:** An object which is modeled and manipulated through data structures and software.

**edge:** In a topologically based mapping system, a directed line from a start node to an end node. The start node and end node may be the same.

**face:** In a topologically based mapping system, a (closed) area bounded a directed cycle of edges.

**feature:** A geographic object such as a road or a building.

**formalization:** The act of defining a problem as rigourously as possible such that a mathematical or computational solution can be determined. Many facets of generalization are intuitive to trained cartographers and as such, many cartographic design and generalization problems have not been formalized.

**generalization, cartographic:** The process of arranging or rearranging the graphic (cartographic) elements of a map so that legibility is attained and map purpose respected. It is often associated with changing the scale of the map.
Appendix A

**generalization, database:** The process of selecting, removing, or grouping together entire classes of features from a dataset in order to simplify the elements that will be represented in the final database. Also called *statistical generalization*.

**graph:** In a topologically based mapping context, the set of all nodes, edges, and faces.

**graphicity:** The ability to communicate via graphics, in contrast to literacy, the ability to communicate via words, or numeracy, the ability to communicate via mathematical principles.

**map purpose:** The intended information meant to be communicated through the graphics of a map. Some maps have multiple purposes, such as topographic maps.

**map scale:** The correspondence of a real world measuring unit to the same measuring unit on the map. (e.g. 1:25,000 signifies that 1 measuring unit on the map is equivalent to 25,000 measuring units on the ground surface.)

**multiple representations:** Displaying the same geodata at different scales or at the same scale with a different (usually simplified) symbology.

**multi-scale database:** A database which has redundant copies of geographic features, some of which may be generalized, where the symbology is tailor made for representation at a specific scale.

**node:** In a topologically based mapping system, a point in a graph, where either an edge begins or ends.

**NP-complete:** NP-complete problems are problems that probably cannot be solved in polynomial running time and are hence considered intractable.

**O-notation:** Describes the upper-bound (worst-case) on the running time of a computer algorithm as a function of the number of input elements, $n$. Algorithms can be rated according to their running times. Constant-time access $O(1)$ is the best; High order polynomial-time algorithms, $O(n^k)$, take much longer and superpolynomial, e.g. $O(k^n)$, can generally not be solved practically for large $n$ (running time: $O(1) \subset O(n) \subset O(n^k) \subset O(k^n)$).

**operators:** A generalization operator is a computational function used to perform a subset of or mimic entirely a procedure a cartographer would use to generalize a cartographic feature or features.

**optimization problems:** A class of problems in which there are many possible solutions, each solution is given a score, and any solution that meets the optimal score, either maximal or minimal depending on the formalization, is an optimal solution to the problem. The optimal solution may not be the optimum solution, if one exists, and discrimination is generally not made between solutions which achieve an optimal score. Some problems are too difficult to find the optimum solution, so an optimal solution sometimes suffices.
**Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization**

**self-evaluation:** Describes the nature of a software system to automatically evaluate the results of modifications performed by operators or sequences of operators with an aim to improve the desired appearance of the final map.

**topology:** As it relates to digital mapping systems and in its fullest usage, describes the modelling of features as nodes (also called vertex, zero-cell), edges (arc, chain, one-cell) or faces (polygons, closed ring, two-cell) and explicitly storing or defining all intersections of lines at nodes, the areas formed by all lines, and which areas neighbor each line. These relationships are invariant under continuous transformations.

**transitional scale change:** A scale change where the symbology changes between the source scale and the target scale.
Appendix B

Selected Prototype APIs

B.1 License

B.1.1 Copyright

Unless expressly noted, all of the documentation and software included in this dissertation is copyrighted by Frank Brazile, Copyright 1997, 1998, 1999, 2000 Frank Brazile. All rights reserved.

B.1.2 List of conditions

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of Frank Brazile may not be used to endorse or promote products derived from this software without specific prior written permission.

B.1.3 Disclaimer

This software is provided by Frank Brazile “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall Frank Brazile be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

B.2 Package ch.unizh.geo.antelope.app

Classes
<table>
<thead>
<tr>
<th>Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line_primitives ................................................................. 109</td>
</tr>
<tr>
<td>Node ................................................................. 110</td>
</tr>
<tr>
<td>Node_list ............................................................. 111</td>
</tr>
<tr>
<td>Dataset ............................................................ 114</td>
</tr>
<tr>
<td>Cloner ............................................................. 118</td>
</tr>
<tr>
<td>Line_poly_flt .............................................................. 119</td>
</tr>
<tr>
<td>Face_list ............................................................... 121</td>
</tr>
<tr>
<td>Partition_list ............................................................ 122</td>
</tr>
<tr>
<td>Face ............................................................... 124</td>
</tr>
<tr>
<td>Transform ............................................................ 127</td>
</tr>
<tr>
<td>Project ............................................................... 129</td>
</tr>
<tr>
<td>Line_segment .............................................................. 140</td>
</tr>
</tbody>
</table>
B.3 Classes

B.3.1 Class Line_primitives

Declaration

```java
public class Line_primitives
extends java.lang.Object
implements java.io.Serializable, java.lang.Cloneable
```

Constructors

- `Line_primitives`
  ```java
  public Line_primitives()
  ```

Methods

- `containsKey`
  ```java
  public boolean containsKey(java.lang.Object key)
  ```
  - Usage
    - Determine whether the named key exists in the internal hash.
  - Returns
    - true means this key is a key exists in the internal hash.

- `get_bounds`
  ```java
  public Rectangle2D get_bounds()
  ```
  - Usage
    - Retrieve the bounding rectangle of all the line primitives in the data set.
  - Returns
    - the 2-D bounding rectangle of all general paths in this collection.

- `get_nodes`
  ```java
  public Node_list get_nodes()
  ```
  - Usage
    - This method returns all of the nodes that are found in this list of line primitives. This is useful during partitioning when working with subsets of total geometry.
  - Returns
    - a Node_list containing all nodes used by the shared line primitives.
  - See Also
    - ch.unizh.geo.antelope.app.Node_list (in B.3.3, page 111)

- `get`
  ```java
  public Object get(java.lang.Object key)
  ```
  - Usage
    - Retrieve a shared line primitive.
  - Parameters
    - `key` - The key that refers to the shared line primitive in the internal hash.
• *keys*
  public Set keys( )
  
  – *Usage*
  * Retrieves the keys from the internal object that stores the has of shared line primitives.
  – *Returns* - A Set of keys identifying each of the shared line primitives.

• *print_all*
  public void print_all( )
  
  – *Usage*
  * The debug method is used to print to standard out all the shared line primitives.

• *put*
  public void put(java.lang.String id, java.lang.Object o )
  
  – *Parameters*
  * id - The key to use to refer to this object while stored internally in a hash.
  * o - The object to store internally in a hash.

**B.3.2 Class Node**

**Declaration**

```java
public class Node
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable
```

**Constructors**

• *Node*
  public Node( )

**Methods**

• *get_id*
  public String get_id( )
  
  – *Usage*
  * Return the unique ID for this node.
  – *Returns* - the unique ID for this node.
* Return the meta information relating to this node, such as unique ID and 2-D position.
  
  - **Returns** - The meta information relating to this topological node, such as unique ID and 2-D position.

- **get_position**
  
  ```java
  public fb.Point2D_Float get_position()
  ```

  - **Usage**
    * Return the 2-D point representing the location of this topological node.
  
  - **Returns** - The 2-D point representing the location of this topological node.

- **set_position**
  
  ```java
  public void set_position( ch.unizh.geo.antelope.buggy_jfc.fb_Point2D_Float p )
  ```

  - **Usage**
    * Set the 2-D point representing the location of this topological node.
  
  - **Parameters**
    * **p** - The 2-D point representing the location of this topological node.

### B.3.3 Class Node_list

#### Declaration

```java
public class Node_list
extends java.lang.Object
implements java.io.Serializable, java.lang.Cloneable
```

#### Constructors

- **Node_list**
  ```java
  public Node_list()
  ```

#### Methods

- **add_node**
  ```java
  public void add_node( ch.unizh.geo.antelope.app.Node n )
  ```

  - **Usage**
    * Add a node to the collection of unique nodes.
  
  - **Parameters**
    * **n** - The Node to add to this collection of unique nodes.

- **entry_exists**
  ```java
  public boolean entry_exists( ch.unizh.geo.antelope.app.Node tmp_node )
  ```

  - **Usage**
* Determine whether a node with the given position already exists in this list of unique nodes.

  - **Parameters**
    * tmp_node - The node to check for duplication in the unique node list.
  
  - **Returns** - if a node with the given position already exists in this list of unique nodes.

- **find_adjacencies**
  
  ```java
  public void find_adjacencies( ch.unizh.geo.antelope.app.Line_primitives lp )
  ```

  - **Usage**
    * Compute and find all line segment adjacencies from the list of unique shared line primitives for each node.
  
  - **Parameters**
    * lp - The list of unique shared line primitives.

- **find_rightmost**
  
  ```java
  public Node find_rightmost( java.awt.geom.Line2D.Float line, java.util.Vector list )
  ```

  - **Usage**
    * Find the right most node from a list of line primitives that are adjacent to an end node.
  
  - **Parameters**
    * line - A line.
    * list - An adjacency list of shared line primitives.
  
  - **Returns** - The right most node.

- **get_adjacencies**
  
  ```java
  public Vector get_adjacencies( )
  ```

  - **Usage**
    * Get the Java structure that represents the list of adjacencies to nodes.
  
  - **Returns** - the reference to a list of adjacencies.

- **get_bounds**
  
  ```java
  public Rectangle2D get_bounds( )
  ```

  - **Usage**
    * Get the 2-D bounds of all nodes in this collection.
  
  - **Returns** - a Java structure that represents the 2-D bounds of all nodes in this collection.

- **get_node_at_pos**
  
  ```java
  public Node get_node_at_pos( java.awt.geom.Point2D tmp_pos )
  ```

  - **Usage**
    * Get the node found at the 2-D position passed as a parameter.
  
  - **Parameters**
    * tmp_pos - the position to inspect.
  
  - **Returns** - The node that can be found at position tmp_pos.
• get_node_pos
  public int get_node_pos( java.lang.Object n )

  – Usage
    * Get the numeric index of the desired node.
  – Parameters
    * n - The node to find in the collection.
  – Returns - The numeric index of the node in the collection.

• get_node
  public Node get_node( int i )

  – Usage
    * Get a node found in this collection of unique nodes.
  – Parameters
    * i - The index of the node to return.
  – Returns - The node with index i.

• get_nodes
  public Vector get_nodes( )

  – Usage
    * Return a reference to the Java structure that represents the node collection.
  – Returns - The node Vector itself.
  – See Also
    * java.util.Vector

• node_at
  public boolean node_at( int i )

  – Usage
    * Determine if there is a node in storage at index position i.
  – Parameters
    * i - the assumed index of a node.
  – See Also
    * ch.unizh.geo.antelope.app.Dataset.feature_at(int)

• num_nodes
  public int num_nodes( )

  – Usage
    * Get the number of nodes in this collection of unique nodes.
  – Returns - The number of nodes.

• print_all
  public void print_all( )

  – Usage
    * Print to standard out the position of all nodes in the list.
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

- **remove_node**
  ```java
  public boolean remove_node( ch.unizh.geo.antelope.app.Node tmp_node )
  ```
  - **Usage**
    - Remove a node from the collection of unique nodes.
  - **Parameters**
    - `tmp_node` - The node to remove.
  - **Returns** - True indicates the node was removed from the collection.

- **theta**
  ```java
  public double theta( java.awt.geom.Point2D.Float p1, java.awt.geom.Point2D.Float p2 )
  ```
  - **Usage**
    - The Theta function is in the Sedgwick algorithms book and is computationally cheaper than computing real trigonometric angles used for point comparisons.
  - **Parameters**
    - `p1` - The first 2-D point.
    - `p2` - The second 2-D point.
  - **Returns** - The ordered theta value.

### B.3.4 Class Dataset

The Dataset class holds all of the cartographic features and geometries and has most of the routines to compute all line intersections for the geometries for when the features are first read into the system and do not have topology defined yet. Some of these methods are awaiting recoding for improved run-time efficiency. There are bottlenecks in the line intersection code.

**Declaration**

```java
public class Dataset
extends java.lang.Object
implements java.io.Serializable, java.lang.Cloneable
```

**Constructors**

- **Dataset**
  ```java
  public Dataset( )
  ```
  - **Usage**
    - Create a new Dataset with an empty collection of features.

**Methods**

- **add**
  ```java
  public void add( ch.unizh.geo.antelope.feature_classes.Geo_feature f )
  ```
  - **Usage**
Add the Geo_feature referenced by f.

Parameters

* f - the Geo_feature to be added.

- `break_lines`

```java
```

Usage

* Compute the line intersections and create a list of nodes to represent each intersection a hash of unique line_primitives.

Returns

- in the parameters, a list of nodes and a hash of line_primitives

- `clear_geometry`

```java
public void clear_geometry( )
```

Usage

* Clear the geometry of each feature in the collection.

See Also

* ch.unizh.geo.antelope.feature_classes.Geo_feature

- `delete_node`

```java
```

Usage

* This delete handles only the simple case of deleting the middle node R in the ordered set {S,R,T}. If the number of primitives incident to R is greater than 2, than tough luck. The reason? there would have to be a series of complex behaviours defined for that occurrence.

S——-R——-T

- `feature_at`

```java
public boolean feature_at( int i )
```

Usage

* Determines wheter the Geo_feature indexed by i exists.

Returns

- true if a feature is present at that location

See Also

* ch.unizh.geo.antelope.app.Dataset.get_feature(int)
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

• **find_intersections**
  
  ```java
  public void find_intersections( ch.unizh.geo.antelope.app.Node_list node_list )
  ```

  **Usage**
  * Find the line intersections for all Geo_features. Currently uses an inefficient n*n algorithm.

  **Parameters**
  * node_list - the list of nodes created by this method

  **See Also**
  * ch.unizh.geo.antelope.app.Dataset.segment.Elements()
  * ch.unizh.geo.antelope.app.Line_poly.flt (in B.3.6, page 119)
  * ch.unizh.geo.antelope.app.Line_segment (in B.3.12, page 140)
  * ch.unizh.geo.antelope.feature_classes.Geo_feature

• **get_collection**
  
  ```java
  public ArrayList get_collection() 
  ```

  **Usage**
  * Return a reference to the Dataset collection itself.

  **Returns**
  * the collection itself()

  **See Also**
  * java.util.ArrayList

• **get_copyof_geometry**
  
  ```java
  public ArrayList get_copyof_geometry( java.lang.String id )
  ```

  **Usage**
  * Return the geometry of the feature uniquely identified by id..

  **Returns**
  * An ArrayList of the geometries for that feature.

  **See Also**
  * ch.unizh.geo.antelope.feature_classes.Geo_feature

• **get_feature**
  
  ```java
  public Geo_feature get_feature( int i )
  ```

  **Usage**
  * Gets the Geo_feature indexed by i.

  **Parameters**
  * i - the index into the 0-based collection container.

  **Returns**
  * the indexed Geo_feature

  **See Also**
  * ch.unizh.geo.antelope.app.Dataset.feature.at(int)

• **get_geometry**
  
  ```java
  public ArrayList get_geometry( java.lang.String id )
  ```

  **Usage**
* Return the geometry of the feature uniquely identified by id..

- **Returns** - An ArrayList of the geometries for that feature.
- **See Also**
  - `ch.unizh.geo.antelope.feature_classes.Geo_feature`

**get_iterator**

```java
public Iterator get_iterator() {
    // Implementation
}
```

- **Usage**
  - Return the iterator for the Dataset collection.
- **Returns** - the collection iterator()
- **See Also**
  - `java.util.Iterator`

**insert_new_node**

```java
public void insert_new_node(ch.unizh.geo.antelope.app.Line_primitives lp,
    ch.unizh.geo.antelope.app.Line_poly_flt polyline,
    ch.unizh.geo.antelope.feature_classes.Geo_feature feature,
```

- **Usage**
  - Helper function to break lines, interrupts a polyline by inserting a new node and creating two polylines.
- **See Also**
  - `ch.unizh.geo.antelope.app.Dataset.break_lines(Node_list, Line_primitives)`
  - `ch.unizh.geo.antelope.app.Line_primitives` (in B.3.1, page 109)
  - `ch.unizh.geo.antelope.feature_classes.Geo_feature`

**instantiate_rendering**

```java
public void instantiate_rendering(double zoom_factor)
```

- **Usage**
  - Create a graphic rendering for all features based on the parameter zoom factor. A zoom factor of 1.0 indicates showing the map information at scale on the screen.
- **Parameters**
  - `zoom_factor` - the pixel based on screen zoom factor.

**num_features**

```java
public int num_features()
```

- **Usage**
  - Returns the number of features stored in this Dataset.
- **Returns** - the number of Geo_features in the collection.

**segment_Elements**

```java
public List segment_Elements()
```
– **Usage**
  * Get all geometries as line segments for use in line intersection and topology computation.

– **Returns** - An ArrayList of all line segments.

– **See Also**
  * ch.unizh.geo.antelope.feature_classes.Geo_feature
  * ch.unizh.geo.antelope.app.Line_poly_flt (in B.3.6, page 119)
  * ch.unizh.geo.antelope.buggy_jfc.fb_line2D_Float
  * ch.unizh.geo.antelope.app.Line_segment (in B.3.12, page 140)
  * java.util.ArrayList

### B.3.5 Class Cloner

This class contains a method to make a deep copy of an arbitrarily complex object using serialization.

This class is taken from the java faq.

#### Declaration

```java
public class Cloner
extends java.lang.Object
implements java.io.Serializable
```

#### Constructors

- **Cloner**
  ```java
  public Cloner( )
  ```

#### Methods

- **cloneObject**
  ```java
  public static Object cloneObject( java.lang.Object o )
  ```

  – **Usage**
    ```java
    try{
      trg = (Src_type)this.cloneObject(src);
    }
    catch(Exception e){
    }
    ```
B.3.6 Class Line_poly_flt

Provides a polyline (line with multiple points) for a topologically based system that has a start node, end node, and middle points that do not intersect other polylines, where coordinates use floating point precision (not Double). Float is used because elements of the line intersection routines part of the JDK were limited to floating point coordinates.

Declaration

```java
public class Line_poly_flt
extends java.lang.Object
implements java.io.Serializable, java.lang.Cloneable
```

Constructors

- `Line_poly_flt`
  - `public Line_poly_flt( )`
    - Usage
      * Constructs an empty polyline with an empty container for the middle points.

Methods

- `add_point`
  - `public void add_point( ch.unizh.geo.antelope.buggy.jfc.fb_Point2D_Float p )`
    - Usage
      * Add a middle point to the end of the middle points container.
    - Parameters
      * p - the 2-D floating precision point to add.

- `get_end`
  - `public Node get_end( )`
    - Usage
      * Get the end node for this polyline.
    - Returns - returns the end node

- `get_id`
  - `public String get_id( )`
    - Usage
      * Get the id for this polyline.
    - Returns - returns a string

- `get_middle_points`
  - `public Vector get_middle_points( )`
    - Usage
- This method returns all the middle points in this polyline in between and exclusive of the start and the end node. Currently, this is used when re-scaling the geometry.
  - **Returns** - returns the vector of middle points

- **get_start**
  ```java
  public Node get_start( )
  ```
  - **Usage**
    * Get the start node for this polyline.
  - **Returns** - returns the start node

- **intersection**
  ```java
  public fb.Point2D.Float intersection( java.awt.geom.Line2D.Float line_1,
  java.awt.geom.Line2D.Float line_2 )
  ```
  - **Usage**
    * Determine to see if two simple line segments intersect. This test may not be very robust.
  - **Parameters**
    * `line_1` - the first 2-D floating precision line.
    * `line_2` - the second 2-D floating precision line.
  - **Returns** - returns the 2-D floating precision point
  - **See Also**
    * `java.awt.geom.Line2D.Float`
    * `ch.unizh.geo.antelope.buggy_jfc.fb.Point2D.Float`

- **print_all**
  ```java
  public void print_all( )
  ```
  - **Usage**
    * A debugging routine, this method prints out the values for the starting position, middle points, and the end position. It does not print out the adjacent face information.

- **set_end**
  ```java
  public void set_end( ch.unizh.geo.antelope.app.Node n )
  ```
  - **Usage**
    * Set the end node for this polyline.
  - **Parameters**
    * `n` - the end node to replace.

- **set_id**
  ```java
  public void set_id( java.lang.String s )
  ```
  - **Usage**
    * Set the id for this polyline.
  - **Parameters**
    * `s` - the string to be used as the id.
• *set_start*

```java
public void set_start( ch.unizh.geo.antelope.app.Node n )
```

- **Usage**
  * Set the start node for this polyline.
- **Parameters**
  * n - the start node to replace.

### B.3.7 Class **Face_list**

**Declaration**

```java
public class Face_list
extends java.lang.Object
implements java.io.Serializable, java.lang.Cloneable
```

**Constructors**

- **Face_list**

```java
public Face_list( )
```

- **Usage**
  * A no argument constructor that creates an object which stores a list faces.

**Methods**

- **calculate_densities**

```java
public void calculate_densities( )
```

- **Usage**
  * Calculates the density of the indexed features for each stored face.

- **create_faces**

```java
public void create_faces( ch.unizh.geo.antelope.app.Line_primitives lp,
                         ch.unizh.geo.antelope.app.Node_list node_list )
```

- **Usage**
  * Using the shared nodes and line primitives, create all of the non-overlapping
topological faces.
- **Parameters**
  * lp - A list of shared line primitives
  * node_list - A list of shared line primitives

- **face_exists**

```java
public boolean face_exists( ch.unizh.geo.antelope.app.Face face )
```

- **Usage**
* Determine whether this face exists already in the list of faces. A return code of true means this face does already exist.

- **Parameters**
  - *face* - a face to determine if it already exists in the list of faces.

- **Returns** - true means the face already exists in the list of faces.

---

**get_faces**

```java
public Vector get_faces() {
    // Implementation
}
```

- **Usage**
  - Retrieve a direct reference to the list (Vector) of faces.

- **Returns** - A direct reference to the list (Vector) of faces.

---

**get_universal_face**

```java
public Face get_universal_face() {
    // Implementation
}
```

- **Usage**
  - Retrieves the universal face which is the parent face of all other faces in the dataset and is guaranteed to be the biggest in extent.

- **Returns** - The universal face that contains all other faces.

---

**index_features**

```java
public void index_features(java.util.ArrayList features_to_index) {
    // Implementation
}
```

- **Usage**
  - In each face, index the geofeatures found in that face.

- **Parameters**
  - *features_to_index* - A list of features to index into containing faces.

---

**print_all**

```java
public void print_all() {
    // Implementation
}
```

- **Usage**
  - Prints the number of faces, which face is the universal face, all faces and corresponding internal faces.

### B.3.8 Class Partition_list

**Declaration**

```java
public class Partition_list extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable
```

**Constructors**

```java
public Partition_list() {
    // Constructor implementation
}
```
Methods

- **create primary**
  ```java
  public void create_primary( java.util.ArrayList line_primitives,
  ch.unizh.geo.antelope.app.Face universal_face )
  ```
  - **Usage**
    - Compute the topological faces that make up the primary partitioning.
  - **Parameters**
    - `line_primitives` - This list of shared line primitives that will make up the primary partitioning.
    - `universal_face` - The Universal face for this topological universe of primary partitioning features.

- **create secondary**
  ```java
  public void create_secondary( java.util.ArrayList line_primitives,
  ch.unizh.geo.antelope.app.Face universal_face )
  ```
  - **Usage**
    - Compute the topological faces that make up the secondary partitioning.
  - **Parameters**
    - `line_primitives` - This list of shared line primitives that will make up the secondary partitioning.
    - `universal_face` - The Universal face for this topological universe of secondary partitioning features.

- **get primary faces**
  ```java
  public Face_list get_primary_faces( )
  ```
  - **Usage**
    - Get the topological faces that make up the primary partitioning.
  - **Returns** - A collection of unique topological faces that form the primary partitioning.

- **get primary line primitives**
  ```java
  public Line_primitives get_primary_line_primitives( )
  ```
  - **Usage**
    - Get the line primitives that make up the primary partitioning.
  - **Returns** - A collection of unique shared line primitives that will be used to compute the primary partitioning of all primary partitioning features.

- **get primary nodes**
  ```java
  public Node_list get_primary_nodes( )
  ```
  - **Usage**
    - Get the nodes that make up the primary partitioning.
  - **Returns** - A collection of unique nodes that will be used to compute the primary partitioning of all primary partitioning features.

- **get secondary faces**
  ```java
  public Face_list get_secondary_faces( )
  ```
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

- **Usage**
  - Get the topological faces that make up the secondary partitioning.
- **Returns**
  - A collection of unique topological faces that form the secondary partitioning.

```java
public Line_primitives get_secondary_line_primitives()
```

- **Usage**
  - Get the line primitives that make up the secondary partitioning.
- **Returns**
  - A collection of unique shared line primitives that will be used to compute the secondary partitioning of all secondary partitioning features.

```java
public Node_list get_secondary_nodes()
```

- **Usage**
  - Get the nodes that make up the secondary partitioning.
- **Returns**
  - A collection of unique nodes that will be used to compute the secondary partitioning of all secondary partitioning features.

```java
public void index_feature_secondary(java.util.ArrayList arr)
```

- **Usage**
  - This method indexes features from the XML named "classname", referenced in features_to_index_{src,trg}, to each secondary partition.
- **Parameters**
  - arr - The list of classes that will be indexed to primary partitions, taken from an XML input configuration file.

```java
public void populate_line_primitives(ch.unizh.geo.antelope.app.Line_primitives lp, java.util.ArrayList line_primitives)
```

- **Usage**
  - Convert the ArrayList of feature geometries into a hash of shared line primitives, modifying lp.
- **Parameters**
  - lp - The hash of shared line primitives.
  - line_primitives - The list of features geometries.

**B.3.9 Class Face**

This concept of a topological face is similar to a GT-polygon. Vector arcs maintains a collection of directed chains that compose this face.

Vector direction specifies if the direction of the chain at the corresponding index of vector arcs is correct or opposite. A value of "true" indicates the direction is correct and no reversing is needed.
Declaration

```java
public class Face
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable
```

Constructors

- `Face`
  ```java
  public Face()
  ```

Methods

- `add_indexed_feature`
  ```java
  public void add_indexed_feature(java.lang.Object obj)
  ```
  - Usage
    - A building or other non-linear feature will be indexed by/to this face.
  - Parameters
    - `obj` - add a feature to the list of indexed features.

- `add_inner_face`
  ```java
  public void add_inner_face(ch.unizh.geo.antelope.app.Face face)
  ```
  - Usage
    - Add a topological face to this face's collection of inner faces.
  - Parameters
    - `face` - add a topological face to the list of inner faces.

- `add_record`
  ```java
  public void add_record(ch.unizh.geo.antelope.app.Line_poly_flt line, java.lang.Boolean correct)
  ```
  - Usage
    - Add a topological arc, "line", to the collection of arcs.
  - Parameters
    - `line` - The topological arc to add to the face.
    - `correct` - true means the start and end nodes do not need to be reversed.

- `area`
  ```java
  public double area()
  ```
  - Usage
    - Computes the area of the outlines of each inner face and subtracts that from the total area.
  - Returns - the area of the face without the inner faces.

- `compute_indexed_density`
  ```java
  public void compute_indexed_density()
  ```
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

- **Usage**
  * Compute the density of features to surface area in this face.

For the time being, this method is a little awkward because it assumes that the indexed features are not inner faces. Therefore, this method returns the area these features consume from this face, NOT counting any inner faces. This could be changed easily.

---

• **get_bounds**
  public Rectangle2D get_bounds()

  - **Usage**
    * Compute the 2-D bounds of the spatial extent of this face.
  - **Returns** - a minimum bounding rectangle.

---

• **get_indexed_density**
  public double get_indexed_density()

  - **Usage**
    * Provide the density of indexed features for this face.
  - **Returns** - the density of indexed features for this face.

---

• **get_indexed_features**
  public ArrayList get_indexed_features()

  - **Usage**
    * Accessor to the class variable index features.
  - **Returns** - The reference to the ArrayList of indexed features.

---

• **get_info**
  public String get_info()

  - **Usage**
    * Return meta-information about this topological face such as bounding rectangle, non-graphic area, number of inner faces, density of the indexed features, and the number of line primitives.

---

• **get_points**
  public Vector get_points()

  - **Usage**
    * Determines all of the points that make up the features that build this topological face.
  - **Returns** - A vector of all the faces points in the face, in the face directed order.

---

• **inner_faces_size**
  public int inner_faces_size()
* Returns the number of topological faces contained in this face.
  
- **Returns** - the number of topological faces contained in this face.

- **interior_contains**
  
  public boolean interior_contains( ch.unizh.geo.antelope.app.Node tmp_node )
  
  - **Usage**
    
    * Determine whether the node passed as a parameter is located inside this face.
  
- **Returns** - true if the node is inside the face.

- **modify_record**
  
  public void modify_record( int index, ch.unizh.geo.antelope.app.Line_poly_flt line, java.lang.Boolean correct )
  
  - **Usage**
    
    * This method should be used when a record was added to the face, but the direction
given may not be correct and was recomputed.
  
  - **Parameters**
    
    * index - the index of the poly line in the list of topological arcs.
    * line - the new poly line.
    * correct - the direction of the topological arc is okay.

- **same_as**
  
  public boolean same_as( ch.unizh.geo.antelope.app.Face tmp_face )
  
  - **Usage**
    
    * Determines if this face is topologically equivalent to the face passed in as a
parameter.
  
  - **Parameters**
    
    * tmp_face - the face to compare with the current face.

- **size**
  
  public int size( )
  
  - **Usage**
    
    * Returns the number of topological arcs that comprise this face.
  
  - **Returns** - the number of topological arcs that comprise this face.

### B.3.10 Class Transform

A class to hold transformation and projection methods

**Declaration**

```java
public class Transform
extends java.lang.Object
implements java.io.Serializable
```

127
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

Constructors

• \textit{Transform}
  
  \texttt{public Transform( ch.unizh.geo.antelope.app.Project \ p, java.lang.String which\_scale )}
  
  \hspace{1em} \textbf{Usage}
  \hspace{2em} \textbullet \hspace{0.5em} The scale\_to\_user parameter is either "map\_scale" or "target\_scale".

Methods

• \textit{compute\_scale\_factor}
  
  \texttt{public void compute\_scale\_factor( )}
  
  \hspace{1em} \textbf{Usage}
  \hspace{2em} \textbullet \hspace{0.5em} Compute the scale factor from original real world coordinates such as UTM to coordinates that fit onto the screen.

• \textit{get\_scale\_factor}
  
  \texttt{public double get\_scale\_factor( )}
  
  \hspace{1em} \textbf{Usage}
  \hspace{2em} \textbullet \hspace{0.5em} Get the scale factor used to scale the geometries.
  \hspace{1em} \textbf{Returns}
  \hspace{2em} \textbullet \hspace{0.5em} The scale factor.

• \textit{print\_all}
  
  \texttt{public void print\_all( )}
  
  \hspace{1em} \textbf{Usage}
  \hspace{2em} \textbullet \hspace{0.5em} Debugging method used to print out meta-information used to compute the projection to standard err.

• \textit{print\_map\_scale}
  
  \texttt{public void print\_map\_scale( )}
  
  \hspace{1em} \textbf{Usage}
  \hspace{2em} \textbullet \hspace{0.5em} Debugging method used to print the map scale to standard error.

• \textit{proj\_to\_viewport}
  
  \texttt{public ArrayList proj\_to\_viewport( java.util.ArrayList geometry )}
  
  \hspace{1em} \textbf{Usage}
  \hspace{2em} \textbullet \hspace{0.5em} "Project" the original geometry to a simple, rescaled coordinate geometry.
  \hspace{1em} \textbf{Parameters}
  \hspace{2em} \textbullet \hspace{0.5em} \textit{geometry} - The Geometries that will be reprojected to the viewport.

• \textit{put\_scale\_factor}
  
  \texttt{public void put\_scale\_factor( )}
  
  \hspace{1em} \textbf{Usage}
* Put the scale factor inherited from the user interface, used to scale the geometries.

• **reproj_to_viewport_vect**

  public Vector reproj_to_viewport_vect( java.util.Vector geometry )

  – **Usage**
    * Reproject a list of geometries represented in a Java Vector to the new viewport.

  – **Parameters**
    * geometry - The list of geometries to reproject.

  – **Returns** - The list of reprojected geometries held in a Java structure.
    Notice: no longer reversing y;

    tmp_y != (1.0 - scaled_y) * (double)scale_factor;

• **reproj_to_viewport**

  public fb.Point2D_Float reproj_to_viewport(
    ch.unizh.geo.antelope.buggy_jfc.fb.Point2D_Float tmp )

  – **Usage**
    * Reproject a node to a new coordinate world.

    Note unfortunately this is not respecting traditional object oriented design.

    Notice: no longer reversing y;

    tmp_y != (1.0 - scaled_y) * (double)scale_factor;

  – **Parameters**
    * The - 2D point to reproject to the new coordinate system.

  – **Returns** - The 2D point at the new coordinate system.

### B.3.11 Class Project

This class holds a Project, which includes a dataset and structures for storing and managing topology and does not refer to the verb meaning "to project" or transform coordinates.

**Declaration**

```java
public class Project
  extends java.lang.Object
  implements java.io.Serializable, java.lang.Cloneable
```
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

Constructors

- Project
  public Project()

Methods

- begin
  public void begin()
  - Usage
    * This method is called when the xml files or serialized feature classes have been specified and starts the geometry scaling and other necessary actions.

- build_partitions
  public void build_partitions()
  - Usage
    * This method builds the source panel topological partitions in a separate topological universe.

- build_topology
  public void build_topology()
  - Usage
    * Build the topology for the current (source) dataset.

- build_trg_partitions
  public void build_trg_partitions()
  - Usage
    * This method builds the target panel topological partitions in a separate topological universe.

- build_trg_topology
  public void build_trg_topology()
  - Usage
    * (re-)build the topology for the target dataset.

- create_def_trg_data
  public void create_def_trg_data()
  - Usage
    * Copy the dataset as is and recreate node lists and shared primitive lists to use in a new co-domain.

- create_gen_view
  public void create_gen_view()
Appendix B

* Create a rendering of the data using any available generalization routines.

---

- **create_line_primitives_from_dataset**
  
  public Line_primitives create_line_primitives_from_dataset(
  ch.unizh.geo.antelope.app.Dataset  d )

  - **Usage**
    * Create a list of shared line primitives.
  
  - **Parameters**
    * **d** - The Dataset that stores all of the features.
  
  - **Returns** - A list of shared line primitives.

  - **See Also**
    * ch.unizh.geo.antelope.app.Dataset (in B.3.4, page 114)
    * ch.unizh.geo.antelope.app.Line_primitives (in B.3.1, page 109)

---

- **create_line_primitives**
  
  public void create_line_primitives( )

  - **Usage**
    * Add all of the line segments to the table of shared primitives.

  - **See Also**
    * ch.unizh.geo.antelope.app.Dataset (in B.3.4, page 114)
    * ch.unizh.geo.antelope.app.Line_primitives (in B.3.1, page 109)

---

- **create_node_list_from_dataset**
  
  public Node_list create_node_list_from_dataset(
  ch.unizh.geo.antelope.app.Dataset  d )

  - **Usage**
    * Create a list of nodes from the list of shared line primitives.

  - **Parameters**
    * **d** - The Dataset that stores all of the features.

  - **Returns** - A list of shared unique nodes.

  - **See Also**
    * ch.unizh.geo.antelope.app.Dataset (in B.3.4, page 114)
    * ch.unizh.geo.antelope.app.Node_list (in B.3.3, page 111)

---

- **create_ungen_view**
  
  public void create_ungen_view( )

  - **Usage**
    * Create a rendering of the data without any generalization performed, that is a straight geometric reduction.

---

- **get_dataset**
  
  public Dataset get_dataset( )
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

- **Usage**
  * Get the dataset used by the source panel in the GUI.
- **Returns** - The dataset used by the source panel in the GUI.
- **See Also**
  * ch.unizh.geo.antelope.app.Dataset (in B.3.4, page 114)

- **get_draw_nodes_src**
  ```java
  public boolean get_draw_nodes_src()
  ```
  - **Usage**
    * Get the boolean value of the source panel draw nodes option.
  - **Returns** - True indicates the nodes should be drawn.

- **get_draw_nodes_trg**
  ```java
  public boolean get_draw_nodes_trg()
  ```
  - **Usage**
    * Get the boolean value of the target panel draw nodes option.
  - **Returns** - True indicates the nodes should be drawn.

- **get_draw_selected_src**
  ```java
  public ArrayList get_draw_selected_src()
  ```
  - **Usage**
    * Get the features that have been chosen for drawing in the source GUI.
  - **Returns** - The features that have been chosen for drawing in the source GUI.

- **get_draw_selected_trg**
  ```java
  public ArrayList get_draw_selected_trg()
  ```
  - **Usage**
    * Get the features that have been chosen for drawing in the target GUI.
  - **Returns** - The features that have been chosen for drawing in the target GUI.

- **get_face_list**
  ```java
  public Face_list get_face_list()
  ```
  - **Usage**
    * Get the list of topological faces for the source panel.
  - **Returns** - The list of topological faces for the source panel.

- **get_features_to_index_src**
  ```java
  public ArrayList get_features_to_index_src()
  ```
  - **Usage**
    * Get the list of features to index into partitions for the source panel.
  - **Returns** - The list of features to index into partitions for the source panel.

- **get_features_to_index_trg**
  ```java
  public ArrayList get_features_to_index_trg()
  ```

132
– **Usage**
  * Get the list of features to index into partitions for the target panel.
– **Returns** - The list of features to index into partitions for the target panel.

- `get_larger_coord_diff`
  public double get_larger_coord_diff( )
  – **Usage**
    * Indicate the spatial extents a scroll panel would need to show all data.
  – **Returns** - the larger X or Y coordinate difference.

- `get_lp`
  public Line_primitives get_lp( )
  – **Usage**
    * Get the list of topological shared line primitives for the source panel.
  – **Returns** - The list of topological shared line primitives for the source panel.

- `get_node_list`
  public Node_list get_node_list( )
  – **Usage**
    * Get the list of unique topological nodes for the source panel.
  – **Returns** - The list of unique topological nodes for the source panel.

- `get_param`
  public String get_param( java.lang.Object key )
  – **Usage**
    * Get the value represented by this key from the internal hash.
  – **Parameters**
    * key -
  – **Returns** - The value represented by this key from the internal hash.

- `get_src_partition_list`
  public Partition_list get_src_partition_list( )
  – **Usage**
    * Get the list of features used to compute the partitions in the source panel.
  – **Returns** - The list of features used to compute the partitions in the source panel.

- `get_src_zoom_factor`
  public double get_src_zoom_factor( )
  – **Usage**
    * Get the zoom factor for the map shown in the source panel in the GUI.
  – **Returns** - The source map zoom factor.

- `get_stk_er`
  public ArrayList get_stk_er( )
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

- **Usage**
  * Get the stacking order for the source panel.
- **Returns** - The stacking order for the source panel.

```java
• get_stk_order
  public ArrayList get_stk_order( )
  - Usage
    * Get the stacking order of feature classes, as drawn in the GUI.
  - Returns - The list of features classed used to set the stacking order.
```

- **Usage**
  * Get the dataset used by the target panel in the GUI.
- **Returns** - The dataset used by the target panel in the GUI.
- **See Also**
  * ch.unizh.geo.antelope.app.Dataset (in B.3.4, page 114)

```java
• get_trg_dataset
  public Dataset get_trg_dataset( )
  - Usage
    * Get the dataset used by the target panel in the GUI.
  - Returns - The dataset used by the target panel.
```

- **Usage**
  * Get the list of topological faces for the target panel.
- **Returns** - The list of topological faces for the target panel.

```java
• get_trg_face_list
  public Face_list get_trg_face_list( )
  - Usage
    * Get the list of topological shared line primitives for the target panel.
  - Returns - The list of topological shared line primitives for the target panel.
```

- **Usage**
  * Get the list of unique topological nodes for the target panel.
- **Returns** - The list of unique topological nodes for the target panel.

```java
• get_trg_node_list
  public Node_list get_trg_node_list( )
  - Usage
    * Get the list of features used to compute the partitions in the target panel.
  - Returns - The list of features used to compute the partitions in the target panel.
```

- **Usage**
  * Get the stacking order for the target panel.
- **Returns** - The stacking order for the target panel.
Appendix B

- **Usage**
  * Get the stacking order for the target panel.
- **Returns** - The stacking order for the target panel.

- **get_trg_zoom_factor**
  ```java
  public double get_trg_zoom_factor() {
  // Implementation
  }
  ```
  - **Usage**
    * Get the zoom factor for the map shown in the target panel in the GUI.
  - **Returns** - The target map zoom factor.

- **init_features_to_index**
  ```java
  public void init_features_to_index(String classname, ch.unizh.geo.antelope.app.Dataset d, String location, java.util.ArrayList arr) {
  // Implementation
  }
  ```
  - **Usage**
    * Find all the features in the Dataset collection that will be indexed to different partitions.
  - **Parameters**
    * **classname** - The classname used during the indexing.
    * **d** - The dataset that holds all relevant features.
    * **location** - source or target domains
    * **arr** - A list of features to index.

  This code looks similar to init_stk_er.

- **init_src_features_to_index**
  ```java
  public void init_src_features_to_index() {
  // Implementation
  }
  ```
  - **Usage**
    * Initialize the features used for indexing into the source partitions.

- **init_src_stk_er**
  ```java
  public void init_src_stk_er() {
  // Implementation
  }
  ```
  - **Usage**
    * Initialize the stacking order for the source panel.

- **init_stk_er**
  ```java
  public void init_stk_er(ch.unizh.geo.antelope.app.Dataset d, String location, java.util.ArrayList draw_selected) {
  // Implementation
  }
  ```
  - **Usage**
    * Creates a structure that renders based on the specified stacking order.
    * `draw_selected` determines if that feature class is turned "on" or not. By default all are turned "on".
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

• init_trg_features_to_index
  public void init_trg_features_to_index( )
  – Usage
    * Initialize the features used for indexing into the target partitions.

• init_trg_stk_er
  public void init_trg_stk_er( )
  – Usage
    * Initialize the stacking order for the target panel.

• partitions_exist
  public boolean partitions_exist( )
  – Usage
    * Determine whether partitioning has been computed for the source panel.
    – Returns - True indicates partitioning has been computed.

• put_param
  public void put_param( java.lang.Object key, java.lang.Object value )
  – Usage
    * Put the object represented by this key into the internal hash.
  – Parameters
    * key - The key used to refer to value.
    * value - The object referred to by key.

• rerender_src
  public void rerender_src( )
  – Usage
    * Recompute any projections for the geometries in the source panel.

• rerender_trg
  public void rerender_trg( )
  – Usage
    * Recompute any projections for the geometries in the target panel.

• set_draw_nodes_src
  public void set_draw_nodes_src( boolean value )
  – Usage
    * Toggle the option whether or not to include nodes in the redrawn image in the
    source panel in the GUI.
  – Parameters
    * value - True indicates to draw the nodes.

• set_draw_nodes_trg
  public void set_draw_nodes_trg( boolean value )
– Usage
  * Toggle the option whether or not to include nodes in the redrawn image in the
target panel in the GUI.

– Parameters
  * value - True indicates to draw the nodes.

• set_larger_coord_diff
  public void set_larger_coord_diff( double num )

  – Usage
    * Transform calls this method ultimately so that we can rescale the scroll plane fromSrcScrollPanel.

  We also use this for the target scale transformation to view coords.

  – Parameters
    * num - the larger coordinate difference between all X coordinates or all Y coordinates.

• set_parameters
  public void set_parameters( java.util.HashMap h )

  – Usage
    * Set the internal structure of parameters to the parameter passed in.

  – Parameters
    * h - The hash of key value pairs used to configure this Project.

• set_part_feature_ids
  public void set_part_feature_ids( java.util.HashMap h )

  – Usage
    * The Partition Specification reader gives a hash of (levels, ids) to this method. This
      method then fans out the levels we’re interested in. Normally, that is to,

      primary

      and secondary (all that are not primary).

• set_primary_feature_ids
  public void set_primary_feature_ids( java.util.ArrayList a )

  – Usage
    * Set the internal list of feature IDs used for the primary partitioning.

  – Parameters
    * a - The list of feature IDs used for the primary partitioning.

• set_secondary_feature_ids
  public void set_secondary_feature_ids( java.util.ArrayList a )

  – Usage
Set the internal list of feature IDs used for the secondary partitioning.

- Parameters
  - a - The list of feature IDs used for the secondary partitioning.

- set_ser_files
  public void set_ser_files( java.util.ArrayList a )

  - Usage
    - Set the list of serialized java object files used by this Project; not used.
  - Parameters
    - a - The list of serialized java object files used by this Project.

- set_src_zoom_factor
  public void set_src_zoom_factor( double value )

  - Usage
    - Set the zoom factor for the map shown in the source panel in the GUI.
  - Parameters
    - value - The source map zoom factor.

- set_stk_order
  public void set stk_order( java.util.ArrayList a )

  - Usage
    - Set the stacking order of feature classes, as drawn in the GUI.
  - Parameters
    - a - The list of feature classed used to set the stacking order.

- set_trg_zoom_factor
  public void set trg_zoom_factor( double value )

  - Usage
    - Set the zoom factor for the map shown in the target panel in the GUI.
  - Parameters
    - value - The target map zoom factor.

- set_xml_files
  public void set_xml_files( java.util.ArrayList a )

  - Usage
    - Set the list of XML files used by this Project.
  - Parameters
    - a - The list of XML files used by this Project.

- shift_coords
  public void shift_coords( )

  - Usage
    - Shift all coords in all features to the viewport.
• **shift_trg_coords**
  
  ```java
  public void shift_trg_coords()
  ```
  
  **Usage**
  
  * Shift all coords in all features to the viewport, which is significant because we have to shift all nodes and shared line primitives.

• **specify_primary_features**
  
  ```java
  public ArrayList specify_primary_features(ch.unizh.geo.antelope.app.Dataset my_dataset, java.util.ArrayList prim_ids)
  ```
  
  **Usage**
  
  * Pass the list of primary partitioning feature IDs to the Dataset.

  **Parameters**
  
  * `my_dataset` - The dataset that will store the geometry information.
  * `prim_ids` - The list of feature IDs used to make the primary partitioning.

• **specify_secondary_features**
  
  ```java
  public ArrayList specify_secondary_features(ch.unizh.geo.antelope.app.Dataset my_dataset, java.util.ArrayList secd_ids)
  ```
  
  **Usage**
  
  * Pass the list of secondary partitioning feature IDs to the Dataset.

  **Parameters**
  
  * `my_dataset` - The dataset that will store the geometry information.
  * `prim_ids` - The list of feature IDs used to make the secondary partitioning.

• **target_exists**
  
  ```java
  public boolean target_exists()
  ```
  
  **Usage**
  
  * Determine whether the target data set has been loaded.

  **Returns** - True indicates the target data set has been loaded.

• **topology_exists**
  
  ```java
  public boolean topology_exists()
  ```
  
  **Usage**
  
  * Determine whether the topology has been computed for the source panel.

  **Returns** - True indicates topology has been computed.

• **trg_partitions_exist**
  
  ```java
  public boolean trg_partitions_exist()
  ```
  
  **Usage**
  
  * Determine whether partitioning has been computed for the target panel.

  **Returns** - True indicates partitioning has been computed.

• **trg_topology_exists**
  
  ```java
  public boolean trg_topology_exists()
  ```
  
  **Usage**
Determine whether the topology has been computed for the source panel.

- **Returns** - True indicates topology has been computed.

```
public void write_line_segments_to_file(java.lang.String filename)
```

- **Usage**
  - This debug method writes all line segments to an output stream and is meant to help test the fast line intersection algorithm.

```
public void write_proj_to_file(java.lang.String filename)
```

- **Usage**
  - Write this entire project and all contents to a GZIP output stream, which can be read in quickly at a later point.

### B.3.12 Class Line_segment

The `Line_segment` class is a simple 2-D floating point precision line used to test intersection during topology construction, and as such is temporary in the system.

#### Declaration

```java
public class Line_segment extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable
```

#### Constructors

- `Line_segment`
  ```java
  public Line_segment()
  ```

- `Line_segment`
  ```java
  public Line_segment(java.lang.String i,
    ch.unizh.geo.antelope.buggy.jfc.fb.Line2D_Float l)
  ```

#### Methods

- `get_id`
  ```java
  public String get_id()
  ```
  - **Usage**
    - Retrieve the hopefully unique ID of this line segment.
  - **Returns** - The hopefully unique ID of this line segment.

- `get_largest_x_point`
  ```java
  public Point2D get_largest_x_point()
  ```
– **Usage**
  * Of the two points that comprise this line segment, return the point with the larger x coordinate value.

– **Returns** - Of the two points that comprise this line segment, return the point with the larger x coordinate value.

---

**get_largest_y_point**

public Point2D get_largest_y_point()

– **Usage**
  * Of the two points that comprise this line segment, return the point with the larger y coordinate value.

– **Returns** - Of the two points that comprise this line segment, return the point with the larger y coordinate value.

---

**get_largest_y**

public double get_largest_y()

– **Usage**
  * Return the largest y coordinate value of the two points.

– **Returns** - The largest y coordinate value as a double.

---

**get_line**

public Line2D get_line()

– **Usage**
  * Get the Line2D object that represents this line segment.

– **Returns** - The Line2D object that represents this line segment.

---

**get_slope**

public double get_slope()

– **Usage**
  * Return the slope of the line.

– **Returns** - The slope of the line as a double value.

---

**get_smallest_x_point**

public Point2D get_smallest_x_point()

– **Usage**
  * Of the two points that comprise this line segment, return the point with the smaller x coordinate value.

– **Returns** - Of the two points that comprise this line segment, return the point with the smaller x coordinate value.

---

**get_smallest_x**

public double get_smallest_x()

– **Usage**
  * Return the smaller x coordinate value of the two points.

– **Returns** - The smaller x coordinate value as a double.
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

- **get_smallest_y_point**
  public Point2D get_smallest_y_point()

  - Usage
    * Of the two points that comprise this line segment, return the point with the smallest y coordinate value.
  - Returns
    * Of the two points that comprise this line segment, return the point with the smaller y coordinate value.

- **intersection**
  public static Point2D intersection( java.awt.geom.Line2D line_1,
                                          java.awt.geom.Line2D line_2 )

  - Usage
    * WARNING: change to fb.Point2D_Float when reintegrating this module
      WARNING: this code is a duplicate of code found in Line_poly_flt: this method should be removed in favor of the other.

- **isLine**
  public boolean isLine( )

  - Usage
    * Test to determine if one line has a differing starting coordinate and ending coordinate, and thus is a line and not a point.
  - Returns
    * true or false

- **setId**
  public void setId( java.lang.String new_id )

  - Usage
    * Set the unique ID for this line segment.

- **toString**
  public String toString( )

- **y_func**
  public double y_func( double some_x )
Appendix C

Prototype XML Data Type Definitions

C.0.13 A sample project file

project.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "project.dtd">
<document>
  <parameters>
    <key>projection</key>
    <value>device</value>
    <key>map_scale</key>
    <value>25000</value>
    <key>target_scale</key>
    <value>50000</value>
    <key>src_features_to_index</key>
    <value>Building</value>
    <key>trg_features_to_index</key>
    <value>Building</value>
    <key>rotation_angle</key>
    <value>0.0</value>
    <key>x_min</key>
    <value>0.0</value>
    <key>y_min</key>
    <value>0.0</value>
    <key>x_max</key>
    <value>500.0</value>
    <key>y_max</key>
    <value>500.0</value>
  </parameters>
</document>
```
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

C.0.14 A sample data file with points and attributes

test_partns.xml

<?xml version="1.0" encoding="UTF-8"?>
<document>
<record>
<class>Border</class>
<kvpairs>
<key>id</key>
[value>bd1</value>
</kvpairs>
<points>
<x>75.0</x>
<y>75.0</y>
<x>425.0</x>
<y>75.0</y>
<x>425.0</x>
<y>75.0</y>
<x>75.0</x>
<y>425.0</y>
</points>
</record>
<record>
<class>River</class>
<kvpairs>
<key>id</key>
[value>r1</value>
</kvpairs>
</record>
</document>
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

```xml
<record>
  <class>Minor_road</class>
  <kvpairs>
    <key>id</key>
    <value>mr1</value>
  </kvpairs>
  <points>
    <x>250.0</x>
    <y>70.0</y>
    <x>250.0</x>
    <y>460.0</y>
  </points>
</record>

<record>
  <class>Highway</class>
  <kvpairs>
    <key>id</key>
    <value>h1</value>
  </kvpairs>
  <points>
    <x>100.0</x>
    <y>50.0</y>
    <x>100.0</x>
    <y>460.0</y>
  </points>
</record>

<record>
  <class>Highway</class>
  <kvpairs>
    <key>id</key>
    <value>h2</value>
  </kvpairs>
  <points>
    <x>400.0</x>
    <y>400.0</y>
    <x>400.0</x>
    <y>50.0</y>
    <x>400.0</x>
    <y>400.0</y>
  </points>
</record>
```
<record>
  <class>Highway</class>
  <kvpairs>
    <key>id</key>
    <value>h3</value>
  </kvpairs>
  <points>
    <x>50.0</x>
    <y>100.0</y>
    <x>450.0</x>
    <y>100.0</y>
  </points>
</record>

<record>
  <class>Highway</class>
  <kvpairs>
    <key>id</key>
    <value>h4</value>
  </kvpairs>
  <points>
    <x>50.0</x>
    <y>400.0</y>
    <x>450.0</x>
    <y>400.0</y>
  </points>
</record>

<record>
  <class>Building</class>
  <kvpairs>
    <key>id</key>
    <value>b1</value>
  </kvpairs>
  <points>
    <x>140.0</x>
    <y>190.0</y>
    <x>190.0</x>
    <y>190.0</y>
    <x>190.0</x>
    <y>220.0</y>
    <x>190.0</x>
    <y>140.0</y>
    <x>140.0</x>
  </points>
</record>
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

```xml
<record>
<class>Building</class>
<kvpairs>
  <key>id</key>
  <value>b2</value>
</kvpairs>
<points>
  <x>300.0</x>
  <y>190.0</y>
</points>
</record>

<record>
<class>Building</class>
<kvpairs>
  <key>id</key>
  <value>b3</value>
</kvpairs>
<points>
  <x>150.0</x>
  <y>350.0</y>
  <x>200.0</x>
  <y>350.0</y>
  <x>150.0</x>
  <y>380.0</y>
  <x>200.0</x>
  <y>380.0</y>
</points>
</record>

<record>
<class>Building</class>
<kvpairs>
  <key>id</key>
  <value>b1</value>
</kvpairs>
<points>
  <x>150.0</x>
  <y>190.0</y>
  <x>200.0</x>
  <y>190.0</y>
  <x>150.0</x>
  <y>130.0</y>
  <x>200.0</x>
  <y>130.0</y>
</points>
</record>
```

148
C.0.15 A sample partition specification file

part_spec.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "part_spec.dtd">
<document>

<record>
  <partition_level>primary</partition_level>
  <singletons>
    <id>mr1</id>
    <id>h1</id>
    <id>h2</id>
    <id>h3</id>
    <id>bd1</id>
  </singletons>
</record>

<record>
  <partition_level>secondary</partition_level>
  <singletons>
    <id>mr1</id>
    <id>h1</id>
    <id>h2</id>
  </singletons>
</record>

</document>
```
<table>
<thead>
<tr>
<th>Line</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td></td>
</tr>
</tbody>
</table>
Index

Aasgaard, 24
AGENT, 29, 31, 35, 43, 62, 77, 87
airphotos, 3, 105
analogue methods, 15
artifical intelligence, 15
artificial intelligence, 15
expert systems, 15
genetic algorithms, 15
neural networks, 15
simulated annealing, 90

Baella, 28
Beard, 19, 42, 43
Bertin, 9
Braendli, 43
Brassel, 15, 17, 19, 30, 33
Brazile, 24, 49, 50, 52
Bruegger, 23
Bundy, 51
Burghard, 91
Buttenfield, 19, 22, 30, 49
cartography, 3
cartographic feature, 105
cartographic procedure, 13, 105
Casasnovas, 23
CHANGE, 27
Christensen, 54, 69
connectivities, 24
constraint, 46
constraints, 15, 19, 33, 42
context-based generalization, 2, 16–18, 86
Cooke, 17
Corbett, 17
Cranston, 72
Cress, 27
Cromley, 9, 17
data quality
fitness for use, 3
Delaunay, 18, 28
Delhi, 19, 81
DeLucia, 19
Dettori, 17
Devogele, 23
Digital Line Graph, 15, 105
DIME, 17
U.S. Department of Commerce, 17
Douglas, 13
Douglas-Peucker algorithm, 13
Doyle, 72
DTM, 28
Dutton, 43
DynaGen, 28
dge, 105
Edwardes, 49
EFDS, 28
Ehrlihoizer, 34
epsilon, 17
ESRI, 27
evaluation, 15, 77
self-evaluation, 2, 86, 107

FDS, 28
Fei, 80
formalization, 13, 105
Fowler, 64
Frank, 23
Freeman, 15
French, 87
Fritsch, 89
FTI, 27

Gamma, 74
Genin, 51
Gold, 18
Gower, 84
Grand, 74
graph, 106
graphicacy, 9, 106
Gruenreich, 27
Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization

Guptill, 34, 43
Hake, 19
Hardy, 29
Harrie, 29, 48, 49, 90, 91
Harvey, 78
Healey, 89
Højholt, 48, 51, 90, 91
International Cartographic Association (ICA), 3
IGN, 23, 29, 30, 43, 67, 75
Ilog, 29
INTERLIS, 72
ISNAP, 28
Jackson, 15
Jansen, 45, 50
Jasinski, 43
Java, 62, 72
Jegenstorf, 4
João, 29
Jones, 18, 29
Kainz, 77
Keller, 15, 72
Kilpeläinen, 23, 29
Kreveld, 18
Kuhn, 23
Lagrange, 42, 77
LAMPS2, 29, 62
Lang, 13
Lang algorithm, 13
Laser-Scan, 43
Laurini, 23
LEDA, 64
Lee, 28, 54
Leitner, 49
Li, 18
line smoothing, 13
Lisp, 29
Lonergan, 9
Mackaness, 15, 45, 49, 51, 79
MAGE, 29
map, 3, 9
scale, 10, 106, 107
topographic, 4
visual variables, 10
McMaster, 3, 15, 19, 33
Mehlhorn, 64
MGE, 28
Minkowski, 36
Molenaar, 28
Monmonier, 15
Morehouse, 17
Morrison, 13, 15
Müller, 22, 30
multi-scale database, 106
multiple representations, 22–30, 106
Nickerson, 15, 19, 79, 90
node, 106
NP-complete, 9, 106
O – notation, 106
OEEPE, 88
Okabe, 18
Olson, 54
OpenGIS, 72
OpenMAP, 72
operators, 16, 19, 29, 106
displacement, 17, 19, 20, 29
exaggeration, 19
simplification, 19, 21
typification, 19, 20
optimization problems, 9, 106
Ordnance Survey, 29
partitioning, 2, 86
patterns, 74
Peng, 18, 24, 28, 49, 50, 59, 60, 80
Perkal, 13
Peucker, 17, 18
Peuquet, 15, 17
Pilouk, 18
Plage, 29
Plazanet, 19, 89, 90
Puppo, 17
Regnauld, 60, 80, 81
RevPG, 27
Rhind, 13
Rieger, 19
Robinson, 15, 19
Ruas, 17–19, 23, 29, 33, 42, 44, 45, 48–52, 77
Saalfeld, 18
Samet, 45, 50
Sarjakoski, 23, 91
Schenkelaars, 23
Schirra, 64
Schlegel, 27
SDS, 29
SDTS, 67
Sedgewick, 64
Swiss Society of Cartography, 3, 10
Shea, 15, 19, 33
Snakes, 91
state diagram, 34
Stratégie, 29
Su, 18
Sweden, 69
Thomas, 54
Thompson, 23
Timpf, 24
TIN, 18
Töpfer, 13
Topology, 50
topology, 17, 35, 42, 50, 64, 107
Turing, 87
URNAGS, 28
USGS, 67
United States Geological Survey (USGS), 27, 37, 67, 73, 105
van Oosterom, 23
Včkovski, 92
Voronoï, 18
Ware, 9, 18, 29, 90
Weibel, 9, 16, 17, 19, 27, 30, 42, 43, 77, 90
Worboys, 60
XML, 62, 67
Zahn, 81
Zoraster, 15