

**Using Discrimination Graphs to  
Represent Visual Knowledge**

by

Jan A. Mulder

Technical Report 85-14  
September 1985

Laboratory for Computational Vision  
Department of Computer Science  
University of British Columbia  
Vancouver, B.C.  
Canada V6T 1W5



by

JAN A. MULDER

B.A.,	University of Leiden (The Netherlands),	1974
M.A.,	University of Leiden (The Netherlands),	1979
M.Sc.,	University of British Columbia,	1979

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF  
THE REQUIREMENTS FOR THE DEGREE OF

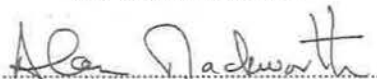
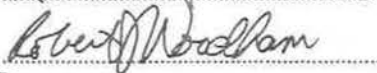
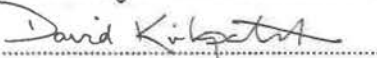
DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming  
to the required standard.

THE UNIVERSITY OF BRITISH COLUMBIA

September, 1985

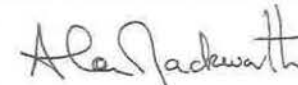
© Jan A. Mulder, 1985

This dissertation is concerned with the representation of visual knowledge. Image features often have many different local interpretations. As a result, visual interpretations are often ambiguous and hypothetical. In many model-based vision systems the problem of representing ambiguous and hypothetical interpretations is not very specifically addressed. Generally, specialization hierarchies are used to suppress a potential explosion in local interpretations. Such a solution has problems, as many local interpretations cannot be represented by a single hierarchy. As well, ambiguous and hypothetical interpretations tend to be represented along more than one knowledge representation dimension limiting modularity in representation and control. In this dissertation a better solution is proposed.

Classes of objects which have local features with similar appearance in the image are represented by discrimination graphs. Such graphs are directed and acyclic. Their leaves represent classes of elementary objects. All other nodes represent abstract (and sometimes unnatural) classes of objects, which intensionally represent the set of elementary object classes that descend from them. Rather than interpreting each image feature as an elementary object, we use the abstract class that represents the complete set of possible (elementary) objects. Following the principle of least commitment, the interpretation of each image feature is repeatedly forced into more restrictive classes as the context for the image feature is expanded, until the image no longer provides subclassification information.

This approach is called discrimination vision, and it has several attractive features. First, hypothetical and ambiguous interpretations can be represented along one knowledge representation dimension. Second, the number of hypotheses represented for a single image feature can be kept small. Third, in an interpretation graph competing hypotheses can be represented in the domain of a single variable. This often eliminates the need for restructuring the graph when a hypothesis is invalidated. Fourth, the problem of resolving ambiguity can be treated as a constraint satisfaction problem which is a well researched problem in Computational Vision.

Our system has been implemented as Mapsee-3, a program for interpreting sketch maps. A hierarchical arc consistency algorithm has been used to deal with the inherently hierarchical discrimination graphs. Experimental data show that, for the domain implemented, this algorithm is more efficient than standard arc consistency algorithms.



## Table of Contents

1. Introduction and Reading Guide .....	1
1.1 What the Research Area is about .....	1
1.2 What this Dissertation is about .....	4
1.3 Reading Guide .....	8
2. Literature Overview .....	9
2.1. Introduction .....	9
2.2. The Representation Problem .....	10
2.2.1. Multiple Levels of Representation in Image and Scene Domain .....	10
2.2.2. Dimensions of Knowledge Representation in the Scene Domain .....	14
2.2.2.1. Composition and Aggregation .....	14
2.2.2.2. Specialization and Generalization .....	15
2.2.2.3. Other Relations .....	19
2.2.3. Composition and Specialization in Computational Vision .....	20
2.3. The Control Problem .....	25
2.3.1. Representational Formats .....	26
2.3.1.1. Semantic Nets .....	26
2.3.1.2. Declarative versus Procedural Representations .....	27
2.3.2. Process Characterizations .....	28
2.3.2.1. The Segmentations and Interpretation Problem .....	28
2.3.2.2. The Cycle of Perception .....	30
2.3.2.3. Constraint Satisfaction .....	33
2.3.2.4. Schema-based Computational Vision .....	37
2.4. Interpreting Sketch Maps .....	44
2.5. The Ambiguity Problem .....	49
2.6. Discussion .....	56
3. Design of Mapsee-3 .....	63
3.1. Introduction .....	63
3.2. Representation .....	68
3.2.1. Schemata .....	68
3.2.2. The Knowledge Representation Dimensions .....	72
3.3 Control .....	78
3.3.1 Composition .....	81

3.3.2. Discrimination .....	89
3.3.2.1. Hierarchical Arc Consistency .....	90
3.4 Setting Up a Knowledge Base for a Particular Domain .....	95
3.4.1. The Basic Composition Hierarchy and Discrimination Graphs .....	97
3.4.2. Constructing an Abstract Discrimination Graph at the Composition Leaf level .....	98
3.4.3. Embedding Abstract Schemata in a Composition Hierarchy and Constructing Abstract Discrimination Graphs at Multi-levels of Composition .....	102
3.4.4. Method Inheritance .....	112
3.5. Discrimination Vision .....	117
4. Description of Mapsee-3 .....	120
4.1. Introduction .....	120
4.2. Input .....	123
4.3. Segmentation .....	123
4.3.1. The Sketch Level .....	124
4.3.2. The Line/Region Level .....	124
4.3.3. The Region Formation Process .....	126
4.4. Image-to-Scene Mapping .....	127
4.5. Interpretation .....	129
4.5.1. The Basic Composition and Discrimination Graphs .....	129
4.5.2. Constructing the Abstract Discrimination Graph at the Composition Leaf Level .....	134
4.5.3. Constructing The Abstract Composition Hierarchy and Discrimination Graphs .....	135
4.5.4. Methods .....	142
4.5.5. Composition and Discrimination .....	147
4.5.6. An Example .....	151
4.6. Summary .....	160
5. Results and Discussion .....	162
5.1. Introduction .....	162
5.2 Results .....	162
5.3. Discussion of Results .....	195
5.3.1. Robustness .....	195
5.3.2. Graceful Degradation .....	196
5.3.3. Domain Independence .....	197
5.3.4. Modularity .....	199



5.3.5 Efficiency .....	202
5.3.5.1 HAC-3 versus AC-3 .....	203
5.3.5.2 Complexity of the Interpretation Graph .....	205
5.3.6. Overall Complexity of the Interpretation Process .....	205
5.4. Generalizing Mapsee-3 .....	206
5.4.1. Constructing an Abstract Composition Hierarchy with Relaxed Restrictions .....	206
5.4.2. Relaxing the Discrimination Constraints .....	208
5.4.3. Method Generalization .....	209
5.5. Discrimination Vision in Context .....	211
5.5.1. Discrimination Vision and Similar Concepts .....	211
5.5.2. Discrimination Vision in a General-purpose Vision System .....	212
5.6 Summary .....	214
6. Summary and Future Directions .....	215
6.1. Summary .....	215
6.2. Future Directions .....	220
6.3 Conclusion .....	223
References .....	224
Appendix A .....	230
Appendix B .....	240
Appendix C .....	250
Appendix D .....	260
Appendix E .....	267

## List of Tables

Table 3.1: A T-junction matrix .....	114
Table 3.2: The T-junction matrix for figure 3.2 .....	115
Table 3.3: The T-junction matrix for two road/rivers .....	115
Table 4.1: Shape attributes of the schemata at the composition leaf level .....	128
Table 4.2: Possible combinations between shape attributes .....	129
Table 4.3: Methods .....	146
Table 4.4: Road-road-tee method .....	146
Table 4.5: Road-over-bridge method .....	147
Table 4.6: Mountain-mountain-tee method .....	147
Table 4.7: Name abbreviations used in the interpretation graph .....	151
Table 5.1: Interpretation color scheme .....	163
Table 5.2: Number of chains vs iteration count .....	194
Table 5.3: Number of instances vs iteration count .....	194
Table 5.4: Number of chains vs method count .....	194
Table 5.5: Number of chains vs number of instances .....	194
Table A1: A truth table representing the constraints between two variables in AC-3 .....	235
Table A2: P-and and P-or truth tables for the hierarchies in figure A1 .....	236

## List of Figures

Figure 2.1: The Mapsee-2 composition and specialization hierarchies .....	46
Figure 2.2: A closed line segment in a sketch map .....	51
Figure 2.3: A simple composition hierarchy .....	52
Figure 2.4: A simple specialization hierarchy .....	52
Figure 2.5: An unnatural specialization hierarchy .....	58
Figure 3.1: A simple discrimination graph .....	75
Figure 3.2: Flow chart of the interpretation process .....	80
Figure 3.3: Two towns connected by a road .....	83
Figure 3.4: A composition hierarchy for Figure 3.3 .....	84
Figure 3.5: A composition hierarchy and discrimination graphs at two levels of representation .....	86
Figure 3.6: A discrimination and superdiscrimination set .....	88
Figure 3.7: A construction example of a discrimination graph .....	102
Figure 3.8: A one-to-one mapping situation .....	103
Figure 3.9: A many-to-one mapping situation .....	104
Figure 3.10: A one-to-many mapping situation .....	106
Figure 3.11: An illustration of the projection algorithm .....	110
Figure 3.12: An example of a T-junction in a sketch map .....	114
Figure 4.1: Lower Mainland of British Columbia .....	121
Figure 4.2: An instance of a line schema .....	124
Figure 4.3: A line hierarchy .....	125
Figure 4.4: The basic composition hierarchy and discrimination graphs .....	131
Figure 4.5: The road-system schema .....	132
Figure 4.6: A relational schema .....	134
Figure 4.7: The discrimination graphs at the composition leaf level .....	135
Figure 4.8: The basic composition hierarchy and discrimination graphs with dummy representations at each level of composition .....	136
Figure 4.9: The composition hierarchy and discrimination graphs for composition levels 1 and 2 .....	138
Figure 4.10: The composition hierarchy and discrimination graphs for composition levels 2 and 3 .....	139
Figure 4.11: The composition hierarchy and discrimination graphs for composition levels 3 and 4 .....	140
Figure 4.12: The composition hierarchy and discrimination graphs for composition levels 4 and 5 .....	141
Figure 4.13: Flow chart of the interpretation process .....	149

Figure 4.14: Illustration of an interpretation graph: stage 0 .....	150
Figure 4.15: Illustration of an interpretation graph: stage 1 .....	153
Figure 4.16: Illustration of an interpretation graph: stage 2 .....	156
Figure 4.17: Illustration of an interpretation graph: stage 3 .....	157
Figure 4.18: Illustration of an interpretation graph: stage 4 .....	158
Figure 4.19: Illustration of an interpretation graph: stage 5 .....	159
Figure 5.1: Ashcroft B.C. ....	164
Figure 5.2: Fraser Valley B.C. ....	165
Figure 5.3: Georgia Strait B.C. ....	166
Figure 5.4: Houston B.C. ....	167
Figure 5.5: Lower Mainland B.C. ....	168
Figure 5.6: Madison, Wisconsin .....	169
Figure 5.7: Okanagan B.C. ....	170
Figure 5.8: Porpoise Island (artificial) .....	171
Figure 5.9: Shuswap Lake B.C. ....	172
Figure 5.10: Spences Bridge B.C. ....	173
Figure 5.11: Segmentation of Fraser Valley .....	174
Figure 5.12: Segmentation of Lower Mainland B.C. ....	174
Figure 5.13: Ashcroft interpretation .....	175
Figure 5.14: Fraser Valley interpretation .....	175
Figure 5.15: Houston interpretation .....	176
Figure 5.16: Georgia Strait interpretation .....	176
Figure 5.17: Lower Mainland interpretation .....	177
Figure 5.18: Madison interpretation .....	177
Figure 5.19: Okanagan interpretation .....	178
Figure 5.20: Porpoise interpretation .....	178
Figure 5.21: Shuswap interpretation .....	179
Figure 5.22: Spences Bridge interpretation .....	179
Figure 5.23: Landmass of Lower Mainland .....	180
Figure 5.24: Howe Sound .....	180
Figure 5.25: Indian Arm .....	181
Figure 5.26: Gambier island .....	181
Figure 5.27: Boundary Bay .....	182
Figure 5.28: Keats island .....	182
Figure 5.29: Lower Mainland road-system .....	183
Figure 5.30: Lower Mainland river-system .....	183
Figure 5.31: Shore of Indian Arm .....	184
Figure 5.32: Coastline of Gambier Island .....	184

Figure 5.33: Shore of Lower Mainland .....	185
Figure 5.34: T-junction of roads .....	185
Figure 5.35: T-junction of rivers .....	186
Figure 5.36: A road crossing over a bridge .....	186
Figure 5.37: A junction between a road and a town .....	187
Figure 5.38: T-junction between a river and a shore .....	187
Figure 5.39: AC-3/HAC-3 iteration count vs number of chains in image .....	188
Figure 5.40: AC-3/HAC-3 iteration count vs number of instances in interpretation graph .....	189
Figure 5.41: Method count vs number of chains .....	190
Figure 5.42: Number of instances vs number of chains .....	191
Figure 5.43: An island with a mountain .....	201
Figure 5.44: Composition hierarchy for the island/mountain example .....	202
Figure A1: The labels of two variables expanded into a hierarchy .....	237
Figure C1: An illustration of the general projection algorithm .....	252
Figure C2: An extreme case of reduction in arity .....	253
Figure C3: The solution to the problem illustrated in figure C2 as provided by the general projection algorithm .....	254
Figure C4: An example of two super-schemata which are discriminations of another .....	255
Figure C5: The solution to the problem illustrated in figure C4 as provided by the general projection algorithm .....	256
Figure C6: The desirable solution to the problem illustrated in figure C4 .....	257

## Acknowledgements

I have acquired many debts in writing this dissertation. First of all, to Alan Mackworth for his advice and supervision of the research reported in this dissertation. I am also grateful to Alan Carter, Bill Havens, Jim Little, and Bob Mercer for discussions and comments on earlier versions of this dissertation.

Thanks are also extended to the other members of my committee: Dave Kirkpatrick, Ray Reiter, Anne Treisman, and Bob Woodham for helpful comments on the work reported. And last but not least, thanks to Gonne and Ellen. Their patience and love were an important factor in the success of this undertaking.



## 1. INTRODUCTION AND READING GUIDE

### 1.1. What the Research Area is about

Vision provides human beings with very powerful mechanisms for perceiving the surrounding world. In just a split second, one can not only recognize and describe objects that appear in the visual field, but also infer motion and distance with apparent ease. Throughout the past decades, vision researchers have sought to explain these mechanisms. While physiological studies of the brain have taught us much about the structure of our visual senses,<sup>1</sup> the higher mental processes cannot be measured directly, because they have no known physical location. Experimental Psychologists derive characteristics of such processes by means of chronometric, recall, and recognition studies.

The development of computers during the last few decades has made possible the design of computer programs that display intelligent behavior. *Artificial Intelligence* is the discipline concerned with the design of these program models. Early research in the field resulted in the development of many computer models for different kinds of intelligent behavior (e.g. Samuel, 1963; Evans, 1963; Newell and Simon, 1963; Quillian, 1969).

---

<sup>1</sup>A good overview of these studies can be found in Hubel and Wiesel (1979).

In the early sixties Artificial Intelligence also became involved with Vision. Computer programs were designed that took the digitized output of a television camera as input and attempted to identify different aspects of the three-dimensional situation that was represented. These studies formed the beginning of a research area that is now known as *computational vision*. Its objectives vary from the study of computational principles underlying vision to the development of high-performance, general-purpose Machine Vision systems.

The problems involved in designing such a system are immense. Take, for instance, the simple task of recognizing a number of children's blocks scattered on a table. For an adult, this kind of task seems trivial, because we carry all the different types of knowledge to perform the task with us (e.g. knowledge about objects, their physical appearance, applications, knowledge of lighting, support, occlusion etc.). The representation and coordination of these knowledge sources in a computer program is a non-trivial task.

For example, edge detection techniques provide us with line segments which mark the areas where intensity changes take place. Such line segments can indicate object and surface boundaries but they may also indicate shadow edges and edges caused by such things as irregularities in surface reflectivity. Thus, image features are highly ambiguous and it is difficult to assign appropriate interpretations to them. A proper interpretation of an image requires a combination of knowledge of the image formation process with knowledge of the objects displayed. The problem of combining and applying these knowledge types to an

image such that the displayed objects can be identified and located is often referred to as the *vision problem*.

Several approaches to solving this problem can be identified. Some researchers have chosen the frontal attack in which the goal is to build systems that take digitized color photographs of outdoor scenes as input and produce a set of meaningful descriptions of the scene as output. Through experience with these systems these researchers hope to acquire an understanding of the kind of knowledge and processes that are necessary and sufficient to build such systems. Others have limited themselves to building systems for a more specific purpose (usually for industrial applications) with the hope that insight will be obtained into the knowledge and processes necessary for success in a particular domain. In contrast, a third group of researchers feel that as a start both the outside world and the vision problem as a whole are too complex to deal with all at once. They feel that one should start by addressing particular aspects of the vision problem in simplified worlds.

Another effect of problems such as the Vision problem is that most of the research in Artificial Intelligence takes place in the form of long term projects. The LNR project at the University of California (Norman and Rumelhart, 1975), the HEARSAY project at CMU (Lesser and Erman, 1977, 1978; Nagao et al, 1978, 1979) and the VISIONS project at Amherst (Hanson and Riseman, 1978) are examples of such an approach.

The research that underlies this dissertation has also been part of such a long term project. The MAPSEE project at UBC is concerned with representational formats and dimensions for representing "high level" visual knowledge. Here the term "high level" applies to knowledge about objects or events that can occur in the scene depicted by an image.

## 1.2. What this Dissertation is about

This dissertation is concerned with the representation of visual interpretations that are ambiguous and hypothetical. The problem directly relates to one of the key issues in the vision problem: the proper mapping of image features to interpretations. Local image features may have many different interpretations. We will therefore refer to this problem as the *ambiguity problem*. Many model-based vision systems use specialization hierarchies as a way of reducing the number of possible local interpretations for each image feature. These hierarchies enable us to replace a set of elementary interpretations by a smaller set of abstract interpretations. In this dissertation we will show that this solution introduces new problems.

This dissertation introduces discrimination graphs as a representation for interpretations that are hypothetical and ambiguous. These graphs represent classes of objects that can have a similar appearance in the image. At the leaves

of such graphs we represent classes of elementary objects which describe the image unambiguously. All other nodes represent abstract classes of such objects. Rather than invoking elementary object classes directly, as is done in most model-based Vision systems, we invoke the class that represents the complete set of possible object classes. Following the principle of least commitment, we then replace this class by one of its subclasses as we expand our focus of attention over the image. This process continues until the image no longer provides information which enables us to do further subclassification.

Discrimination graphs offer many advantages over specialization hierarchies. First, all possible local interpretations for a single image feature can be represented within the boundaries of one discrimination graph. This cannot be achieved with specialization hierarchies, because the possible local interpretations for a single image feature often cannot be captured in a single specialization hierarchy. Second, we can further reduce the number of interpretations that we have to represent explicitly for each image feature. Third, hypothetical interpretations that are competing in the interpretation of a particular image feature can be represented as labels in the domain of a single variable in a constraint graph.<sup>2</sup> In model-based vision systems that use specialization hierarchies at least some competing hypotheses must be represented by different variables. A disadvantage of the latter approach is that every time a hypothesis is invalidated the constraint graph needs to be restructured. With all competing hypotheses

---

<sup>2</sup>We assume that the nodes in such a graph represent abstract objects, the domain their elementary descendants in the discrimination graph, and the links the constraints between the objects.

represented in the domain of a single variable the removal of an inconsistent hypothesis requires only the deletion or replacement of a label in the domain of the variable. As a result, structural changes in the constraint graph are required only in exceptional conditions.

Little is known about the complexity of algorithms which are used to restructure constraint graphs. Propagation of consistency over labels in the domain of a variable, on the other hand, puts us in the domain of constraint satisfaction. Constraint satisfaction algorithms have been extensively studied and used in computational vision.

The idea of using discrimination graphs to represent hypothetical and ambiguous interpretations does not depend on any characteristic of the interpretations themselves and is therefore domain-independent. In this dissertation, however, we concentrate mainly on the use of discrimination graphs for the representation of visual knowledge. In particular, we will describe the design and implementation of Mapsee-3, a sketch map interpretation program that uses discrimination graphs. As a part of the MAPSEE project, Mapsee-3 has inherited an interest in the so-called schema-based object representations. It has also inherited an interest in the knowledge representation dimensions studied: composition/aggregation and specialization/generalization. In Mapsee-3 the latter has been replaced with a discrimination/generalization dimension which is orthogonal to the composition/aggregation dimension.

Mapsee-3 interprets image features which at first are assigned an extremely abstract and ambiguous interpretation. As more and more constraints are discovered in the image, constraint propagation techniques force this interpretation to become more specific and less ambiguous. Discrimination graphs are closely involved in this process. We therefore call this particular approach *discrimination vision*.

A strong emphasis is put on the *conceptual clarity* of the Mapsee-3 design. The conceptual clarity of a system can be evaluated by several criteria:

1. modularity in representation
2. modularity in control
3. uniformity in representation
4. strict separation between domain dependent and domain independent knowledge

It will be shown that Mapsee-3 rates better with respect to these criteria than Mapsee-2, a schema-based program that uses specialization hierarchies instead of discrimination graphs. As well, Mapsee-3 will be shown to be more efficient.

### 1.3. Reading Guide

There are many factors in computational vision that result in ambiguity. The one addressed in this dissertation is the ambiguity that occurs in mapping image features to "high level" interpretations. For this reason the representation of "high level" knowledge is a major concern. In the literature review in Chapter 2 the focus is therefore on the representation and use of this knowledge.

Chapter 3 describes our solution to the ambiguity problem. We discuss the design principles of a system for representing visual knowledge. The design principles are largely domain-independent, at least to the degree that they are applicable to any signal processing domain. Mapsee-3 is an implementation of these principles. This program is described in Chapter 4. The interpretations made by Mapsee-3 from 10 different sketches are discussed in Chapter 5. As well, the ambiguity problem is discussed from a wider perspective. Finally, a summary of this dissertation and possible future directions are provided in Chapter 6.



## 2. LITERATURE REVIEW

### 2.1. Introduction

In this chapter, we review the computational vision literature. It is selective, because of its focus on the "high level" aspect of computational vision. Most of the work in early vision such as edge detection and region formation will be bypassed. This work has been ably reviewed on several occasions (e.g. Barrow and Tenenbaum, 1981; Brady, 1982).

Two passes will be made through the literature. In the first pass, we focus on representation, or, more precisely, the question of the representation of structural descriptions that capture the meaningful organization of an image. During the second pass we are concerned with control, that is, how to characterize the process that constructs and utilizes different structural descriptions. The former problem is also known as the problem of "epistemological adequacy" (McCarthy and Hayes, 1969) or "descriptive adequacy" (Havens and Mackworth, 1983). The latter problem is also referred to as "heuristic adequacy" (McCarthy and Hayes, 1969) or "procedural adequacy" (Havens and Mackworth, 1983).

In the first of the next five sections, we discuss the representation problem, in the second one the control problem. In the third section we discuss some of the work done on the use of sketch maps in computational vision. In the fourth sec-

tion we address the central theme of this dissertation: the representation of interpretations which are ambiguous and hypothetical. In that section we show how most computational vision systems deal with the problem. In the last section the review is summarized, and a different approach for dealing with interpretations that are ambiguous and hypothetical is proposed.

### 2.2. The Representation Problem

#### 2.2.1. Multiple Levels of Representation in Image and Scene Domain

Most of the early computational vision systems were concerned with the blocks world environment. The reason for such a choice is obvious. Blocks are among the simplest three-dimensional objects. These early programs take a two-dimensional line drawing or the digitized output of a TV camera as input and they attempt to identify different aspects of the three-dimensional situation. The two-dimensional drawing is usually referred to as *picture* or *image*; the three-dimensional situation as the *scene*. Some of the lessons learned from these programs are most useful.

Guzman (1968), for example, wrote a program that starts from a specification of the picture lines and vertices. His program groups the regions into "nuclei of bodies", where each body represents an object. The process of linking

regions together is driven by a list of vertex arms specifying the arms of vertex types which can link regions. Thus, the information about what constitutes valid three-dimensional objects is represented in one domain only, the picture domain. Although a few three-dimensional situations were handled correctly, the many failures (e.g., the inability of the program to handle objects with holes) illustrate the necessity to represent the knowledge about three-dimensional objects in more than one domain, as was noted by Clowes (1971).

Clowes represented the information about three-dimensional objects in two domains: a picture domain and a scene domain. For example, in the picture domain one speaks of lines, regions, and vertices, whereas in the scene domain these primitives take the form of edges, surfaces, and corners respectively. Clowes also noted that not all vertex configurations in the picture make sense in the scene domain: only certain configurations are possible<sup>1</sup>. Clowes investigated the possible configurations and interpretations of four vertex types: L, Fork, Arrow, and Tee. For each edge he allowed four different types of interpretations: convex, concave, and two types of occlusion.

Thus, the necessity was shown for distinguishing between at least two domains of representation: a picture domain with descriptions of the two-dimensional aspects of the image and a scene domain with descriptions of the three-dimensional aspects. Subsequent research, however, showed the need for a further refinement of representations in both picture and scene domain.

<sup>1</sup>A similar observation was made by Hoffman (1971).

An obvious shortcoming in the Clowes labeling is, for instance, the inability to deal with shadow lines. Waltz (1972) therefore extended the classification schema. Among other things, picture lines can represent shadow lines. Waltz's line labels also express the illumination status of the surfaces appearing at the edge.

Although Waltz's system meant a further step to a more adequate description of the scene it is still no match for human competence in the domain. Edges do not only express surface relationships, but also have a spatial orientation. Surfaces have orientations and can frequently be labeled with a meaningful name: side-face, or top-face for a polyhedral object, or a door, wall, or roof, if the line drawing depicts a house. Finally, the object depicted by the line drawing as a whole carries a meaningful name: cube, wedge, or house. All such descriptions require a stratification of knowledge in the scene domain. The more recently developed computational vision systems reflect this requirement (e.g., Hanson and Riseman, 1978; Mulder, 1979).

However, the situation becomes even more complex when the domain of interpretation changes from blocks to outdoor scenes. The correct recognition of an object can no longer depend on the availability of one knowledge source only. An object has many different appearances, depending on the observer's position, illumination conditions, and context. Different knowledge sources must be coordinated in order to correctly predict an object's appearance. Several systems have been designed and implemented to interpret outdoor scenes, or aspects of them

(e.g. Bajcsy and Lieberman, 1974; Hanson and Riseman, 1978). Among other things, Bajcsy and Lieberman used four different knowledge sources: knowledge about the world, the observer, the illumination, and the environment.

It is also necessary to maintain different representations in the image domain. Even in unfamiliar or strange situations such as abstract art people can estimate intrinsic characteristics such as color, orientation, shape and illumination. Phenomena such as shape, size, and color constancy are well known. Variations in illumination do not change our perception of surfaces. A black piece of paper may, for instance, reflect more light than a piece of white paper in shadow, but the pieces of paper are still perceived as black and white respectively.

Apart from different object dependent representations in the scene domain, one should therefore also maintain different representations in the image domain: representations that can be computed from the physical information provided by the image (such as color and incident illumination). Marr (1978), for instance, has proposed a "low level" representation that captures the intensity changes and the local three-dimensional geometry of an image. Barrow and Tenenbaum (1978), starting from an intensity image, use the knowledge about viewer and light-source position to compute and make consistent information about illumination, reflection, orientation, and distance of surface. Woodham (1981) showed that by varying the incident illumination under constant viewing direction one can uniquely determine the surface orientation at each image point, using a technique called photometric stereo.

### 2.2.2. Dimensions of Knowledge Representation in the Scene Domain

This dissertation is mostly concerned with the representation of knowledge in the scene domain. If we want to account for human competence in this domain, an increasing stratification of the knowledge itself is required. However, different levels of representation do not stand by themselves. Often we can impose an ordering upon them by using different *relations*.

Two kinds of relations play a very important role in describing visual knowledge: the component relations and the "is-a" relations. Two dimensions of knowledge representation result from these relations: the *composition/aggregation* dimension, and the *specialization/generalization* dimension.

#### 2.2.2.1. Composition and Aggregation

Concepts can be decomposed into parts and they can be aggregated into super-components. The ability to describe objects at different levels of composition contributes to the power of the visual system. We often do not know in advance at which level of detail an object will appear in the image. Thus, the ability to recognize an object by its overall shape, if it is completely visible, or by its components, is an important feature of the system.

Not only are objects often embedded in a network of *composition* relations, two different objects with a common super-component are often related as well. In static imagery the relationship is usually spatial. In motion analysis, on the other hand, composition can be used as an explicit organization for a sequence of events (e.g. Tsotsos, 1984).

Composition comes in two variations: *must-be-part* and *may-be-part*. The former indicates that under all circumstances one concept is a component of another, while the latter does not have this requirement. Outside the computational vision area the composition/aggregation dimension has received little attention. The ability to reason with parts can be found in Raphael's SIR program (Raphael, 1964). More recently, Schubert (1979) has addressed the problems of mechanizing the extraction of composition relationships from tangled hierarchies and the problem of relationship inheritance for parts of objects in a taxonomy. Some knowledge representation languages use composition, which will be discussed further in the section on control.

#### 2.2.2.2. Specialization and Generalization

Another way of structuring the world is to follow the principle of *classification*. Classification means to consider a number of objects, situations, or events as equivalent. Classification is natural, because many real world attributes do not occur independently of each other. Creatures with feathers, for instance,

are more likely to have wings than creatures with fur. A name is generally associated with a class: animal, bird, etc. By means of *class inclusion* different classes are organized into a system. Such a system is called a *taxonomy*.

Specialization or "is-a" hierarchies have been frequently used in the knowledge representation literature, generating a great deal of controversy. In his paper titled "What is-a is and isn't" Brachman (1982) has attempted to make an inventory of different "is-a" interpretations.

First of all, there is the question whether the nodes in the taxonomic hierarchies represent classes or individuals. According to Brachman, an "is-a" link associating a class with an individual has at least four possible interpretations. The most commonly used one is the set membership. It is also the one most commonly found in the computational vision literature where it is called an *instance* link.

For the case in which a taxonomy involves classes only, Brachman found five different partially overlapping interpretations:

- subset/superset:

This interpretation can be found in expressions such as: "A canary is a bird". If  $x$  is in the set of canaries, then it is also in the set of birds. Universal implication appears to be implied in this distinction.

- generalization/specialization:

This interpretation is expressible in the form of a simple conditional, which is not a

well formed first-order formula:

$$\text{canary}(x) \rightarrow \text{bird}(x)$$

This particular interpretation of an "is-a" link comes from Hayes (1979). The absence of a quantifier makes this expression confusing. A universal quantifier in front of this expression makes it indistinguishable from the subset/superset case. On the other hand, if the concepts represent prototypes then we need default rules specifying which properties are inherited in the implication.

- AKO:

To a large extent this interpretation is the same as the subset/superset interpretation. A canary is also "a-kind-of" bird. However, within the subset/superset distinction there is sometimes a need for distinguishing between "kinds" and classes that stand for more arbitrary descriptions such as "a person walking to school" who is a person, but not "a kind of" person. The "a-kind-of" interpretation implies "kinds".

- Value restriction:

"The trunk of an elephant is a cylinder 1.3 meters long". The interpretation is to say that, in order to describe the trunk of an elephant as a cylinder of 1.3 meters, we need a particular context. In Mapsee-3 we will introduce a new kind of "is-a" hierarchy which will be known as a discrimination graph. This graph has the value restriction interpretation.

- Conceptual attainment:

"A triangle is a polygon." Here we mean to express the fact that a triangle is a polygon with three sides, i.e. to demonstrate a case in which one description includes another. This is different from value restriction where some kind of context is needed to progress from one class to the next. However, it is difficult to distinguish this interpretation from the subset/superset interpretation.

Martin (1979) distinguishes as many as nine different forms of specialization. In addition to the individual/class, the specialization/generalization (specialization by species), and the value restriction distinction (specialization by context), Martin distinguishes between specialization by inflection (e.g. dog - dogs), predicate (e.g. fat dogs), appositive (pet dog), stereotype (e.g. lap dog), and slot (e.g. hit ball). Even composition is considered to be a form of specialization. For instance, the "leg of a dog" is seen as specialization by role.

A further complication in distinguishing types of taxonomies is what the class itself represents. Generally, there are two possibilities. The first is that the class is a set of attributes that *all* members of that class have to satisfy. The other is that the class represents the attributes of a typical member. In the latter representation property inheritance need not be universal. A prototypical bird flies, but penguins do not. An example of a system in which both class representations are used is NETL (Fahlman, 1979). Reiter (1980) has developed a form of non-monotonic logic that makes explicit the kind of default reasoning necessary for prototypes.

### 2.2.2.3. Other Relations

#### *Spatial*

Objects can only enter into certain spatial relationships with each other. Objects that are only partially visible can often be recognized as a result of the fact that their components have entered in certain spatial relationships.

#### *Similarity*

This relation was originally proposed by Minsky (1975). Tsotsos *et al* (1980) use this relation as a means of selecting alternative hypotheses. Similarity relations come close to the concept of discrimination graphs which we will propose in Chapter 3.

#### *Depiction*

Depiction is a relation linking concepts in different domains of representation. In vision, depiction is the relation between objects in the image and scene. For instance, a line in the image may depict a corner in the scene. This relation has also been called the relation of representation (Clowes, 1971), and projection (Shibahara *et al* 1983; Tsotsos, 1984).

#### *Causal*

Causal relations are important in knowledge bases in which one event is known to cause another. Rieger and Grinberg (1977) have used such a relation for

representing causality in physical mechanics. More recently, such a relation has been used in medical image interpretation (Shibahara *et al* 1982; Shibahara *et al* 1983).

### 2.2.3. Composition and Specialization in Computational Vision

The summary of possible "is-a" interpretations characterizes the confusion that exists in the field of Knowledge Representation. Fortunately, in computational vision there is less confusion. In most vision systems the "is-a" hierarchies are based on universal implication and are referred to as specialization hierarchies. While an explicit use of the two knowledge representation dimensions is rarely found in the early computational vision literature, more recent work acknowledges the importance of both dimensions.

Roberts's (1965) program recognizes objects in a scene as being instances of a class of three different models: a cube, a rectangular wedge, and a hexagonal prism. These objects are described by their three-dimensional homogeneous coordinates. Compound objects consisting of configurations of the three models can be recognized as well. Guzman's (1968) program determines only the number and location of objects in the scene. Clowes (1971) only determines whether the object(s) in the scene make three-dimensional sense. Falk's (1972) INTERPRET can recognize a set of nine fixed size prototypes, but does not make a distinction between classes of prototypes.

In a few systems the presence of both composition and specialization hierarchies can be demonstrated at more than one level. Winston's (1975) work on structural descriptions is an example of this. Winston wrote a program that derives an abstract (generic) structural description for an arch by providing the program with descriptions of examples and non-examples of that object. The descriptions of the arch and the (non)examples are compared via a similarity network and the arch description is adjusted in case of inconsistencies. Both hierarchies are visible in the network representation (Winston 1975, p. 198). The composition relation comes in two varieties: must-be-part and its negation: must-not-be-part. The class of a node in the network can be determined by following its "a-kind-of" link.

Marr and Nishihara (1976) have proposed a method for representing 3D-shapes, based on a hierarchy of stick figures. Each stick forms the central axis of a generalized cone representation. 3D-model representations are formed by means of a composition hierarchy of stick figures. The central axis of each component of the stick figure is defined relative to the axis of its super-component. In addition, a specialization hierarchy is formed by means of what Marr and Nishihara call a catalogue of 3D-models. Their method further consists of an image-space processor which maps representations from an object-centered frame into a viewer-centered frame and *vice versa*. As well, their method appears to be the first one to use the principle of least commitment in computational vision. Such a principle implies that nothing should be done that may later have to be undone. We will see later that Mapsee-3 operates on the same principle.

Rosenthal and Bajcsy (1978) constructed an inquiry-driven computer vision system. The system is told to look for a particular object in the scene. Among other features, the system uses composition and specialization hierarchies in its search for the object in the image. For instance, when told to look for a car, the system will find that a car is a sub-class for motorvehicle which, in turn, is a component of a thoroughfare. A thoroughfare can, because of its particular shape, be located in the image and the system then proceeds to look for a car inside the thoroughfare strip. Rosenthal and Bajcsy make use of the fact that a *part-of* relation in the scene is often equivalent to an *inside* relationship in the image. This relationship has been made explicit by making the composition hierarchy cross the image/scene boundary. In Mapsee-3 an image-to-scene mapping relation will be used for this purpose.

Composition and specialization hierarchies form an important part of the knowledge representation in the ACRONYM system (Brooks, 1981 and 1983). Models are represented as generalized cones. Their data structures are organized as units with slots and fillers to define their values. Composition hierarchies are recognizable as a subgraph of what Brooks calls the object graph. The nodes in this graph represent models and the arcs are units of class subpart or affixment. Specialization hierarchies appear in the form of a restriction graph in which the nodes represent sets of constraints on the model description and arcs represent subclass inclusion.

In the MAPSEE project an explicit representation of both knowledge representation dimensions has also been pursued by several researchers. Havens (1978) wrote a program that describes line sketches of polyhedral objects at three levels of composition. Havens and Mackworth (1983) use a seven level composition hierarchy in Mapsee-2, a program that interprets line sketches of geographic maps. In a bottom-up direction the relation used is "must-be-part-of". Specialization hierarchies are part of the knowledge representation in this program as well.

A more complex composition hierarchy is used in the MISSEE system (Glicksman, 1982), an offshoot from the MAPSEE project. MISSEE uses multiple information sources for interpreting a digital image. Composition hierarchies form one of the means for combining data about objects from different information sources. Specialization hierarchies are used as well. A short description of the MISSEE system is provided in Glicksman (1983).

Both knowledge representation dimensions can be found in the VISIONS system (Hanson and Riseman, 1978). This system is possibly the most complete general-purpose vision system. Multiple levels of representation in both the image and scene domain are used. As well, at an intermediate level the image is represented by means of surfaces and volumes. Composition hierarchies are present in both the image domain (e.g. region shapes and their components), and the scene domain (e.g. objects and their components). Specialization hierarchies are also present in the scene domain. More recent work in the VISIONS project is described in Weymouth (1981) and Weymouth *et al* (1983).

Composition and specialization are also the prime representational axes in the knowledge representation research done in the Laboratory for Computational Medicine at the University of Toronto. One of the systems developed in this lab is a causal arrhythmia analyzer (CAA) which can recognize repetitive time varying signals such as electrocardiograms (Shibahara *et al* 1982; Shibahara *et al* 1983). The composition and specialization semantics were inherited from a knowledge representation language called PSN (Levesque and Mylopoulos, 1979).

Finally, both hierarchies have been exploited by Browse (1982). Browse's program interprets line sketches of a body-form and determines the two-dimensional position and three-dimensional orientation of its body parts. An interesting aspect of Browse's work is that the line sketch is scanned at different levels of resolution. Browse has suggested that there is a relationship between the levels of detail in the image at which cues are constructed and the level of composition and specialization of the concepts in the scene to which these cues have access. Browse maintains two composition hierarchies. One describes the image at a coarse level, the other at a fine level. Different specialization hierarchies connect the two composition hierarchies.



### 2.3. The Control Problem

So far our main concern has been the *what* and *why* aspect of knowledge representation. In this section we address the *how* question; that is, the question how we represent and how we characterize the processes that use these representations. This section consists of two parts. In the first part we discuss some of the representational formats that have been used in Artificial Intelligence in general, and computational vision in particular. In the second part we discuss different characterizations of the processes that use the knowledge represented.

### 2.3.1. Representational Formats

#### 2.3.1.1. Semantic Nets

Network models of information are based on the concept of associative memory, an idea that Anderson and Bower (1973) have traced back as far as Aristotle. Quillian (1966) is generally attributed with the origin of the use of semantic nets. In Quillian's system the nodes represent word concepts. The links form an indication of the type of inference that can be made from the concept. A distinction between specialization and composition is already apparent in his TLC model (Quillian, 1969). TLC was an investigation of the usefulness of a semantic net as a knowledge base for the reading of text. Among other things this semantic net consists of sets of hierarchies of concepts. A set of properties has been attached to each node which defines the corresponding concept.

Although semantic nets are a nice way of visually illustrating the structure of a knowledge base, it is difficult to infer the formal syntax and semantics of the net from such an illustration. In his famous paper titled "What's in a link" Woods (1975) pointed out that a lot of intuition is often necessary to understand what a semantic net really represents. Questions should be asked about the semantics of the representation itself. The diversity of possible semantics for "is-a" hierarchies is a clear example of this.

Since then, different attempts have been made to be more formal about semantic nets. Schubert (1976), for instance, has provided a clear correspondence between his network notation and predicate calculus. Brachman (1979) has also stressed the need for knowledge-structuring primitives. In the next chapter, when the concepts behind Mapsee-3 are discussed, semantic nets will be used for illustration only. The *schema* will be used as a knowledge-structuring primitive. The concept of *schema* will be discussed in the next section.

### 2.3.1.2. Declarative versus Procedural Representations

The problems of representation and control can be embodied in procedural/declarative tradeoffs (Winograd, 1975). The procedural view assumes that all "knowing" is equivalent to "knowing how" while the declarative view emphasizes "knowing" as "knowing that". The declarative approach provides economy of representation and verifiability. The former is achieved because it is easy to restrict the representation of each kind of information to one particular location. The latter is the case because of the close ties between declarative representations and the mechanisms of First Order Logic. In particular for the latter reason declarative representations are favored in Mapsee-3.

The procedural approach, on the other hand, relies on specific procedures for specific problems. However, an adequate exploitation of the semantics of images requires a mixture of the two approaches. In Mapsee-3, some of the domain-

dependent knowledge is represented procedurally.

### 2.3.2. Process Characterizations

#### 2.3.2.1. The Segmentation and Interpretation Problem

The input to a vision program usually comes in the form of a digitized array of pixels of varying intensity. The goal of the program is to provide different descriptions in terms of models provided by the user. However, pixels do not serve as meaningful units that can be interpreted in terms of models. These models need information about more abstract units which may correspond to such things as object and surface boundaries, shadows, and other illumination effects.

The question of how to abstract this information is usually referred to as the segmentation problem. But even if we have segmented correctly, we are still faced with the question of how to transform descriptions of the image in terms of edges and surfaces into descriptions that capture its meaningful organization. This is usually referred to as the interpretation problem.

The two problems are not totally independent, however. Some computational vision researchers hold that even for segmentation some knowledge about what the image depicts is necessary (e.g., Mackworth, 1977a), whereas others

hold that some knowledge about the intrinsic aspects of objects is sufficient (e.g., Barrow and Tenenbaum, 1978).

One possible reason for this disagreement is that for many early computational vision studies line drawings were used. The usual experience with this domain is that the information which can be abstracted from the image is insufficient to uniquely determine the content of the scene. The only information implicit in a line drawing is about the location and shape of lines. Information about such things as surface texture and reflectivity is absent.

It has been argued (Clocksin, 1978) that as a result of this, computational vision researchers have been forced to rely on control paradigms in which the use of domain-dependent knowledge is the only way to achieve an adequate interpretation of the image. In the next section we introduce such a paradigm.

Several researchers (e.g., Marr, 1978; Barrow and Tenenbaum, 1978) have argued, however, that information about color, texture, and incident illumination can tell us a lot about such things as surface boundaries and shadows. This information can be abstracted by using very general knowledge about objects (e.g., assumptions about surface continuity). However, the degree of adequacy with which this kind of abstraction can be made is a matter of ongoing research.

Although it is clear that in the long term the interpretation problem cannot be solved without at the same time providing a solution for the segmentation problem, many computational vision researchers have not waited for the solution

of the latter problem in order to tackle the former. They rather assume that it is possible to create certain representations in the image domain and have focused on the interpretation problem instead.

It can be argued that line sketches present an impoverished stimulus environment, but this does not refute the fact that humans are perfectly able to recognize them. Thus, we may still hope that at least a limited set of perceptual principles can be studied in line sketch perception. One should be aware, however, that some interpretation problems may be caused by line sketch artifacts.

In Mapsee-3 we avoid the segmentation problem. We will assume it is possible to segment the image into a set of primitives without any knowledge of the scene. Later on, during the discussion in Chapter 5, we will propose a more dynamic solution which does not require a perfect segmentation. Mapsee-3 also uses line sketches.

#### 2.3.2.2. The Cycle of Perception

One of the central paradigms for the control structure of vision programs stems from the first program that could recognize different polyhedral objects. This program was written by Roberts (1965). His program consists of two parts: a program that reduces a gray level picture to a line drawing, and a program that interprets the line drawing. The program can recognize three different types of

polyhedra: cubes, wedges, and prisms.

Mackworth (1977a) noted that the perceptual process in Roberts's program can be characterized as a sequence of four processes: *cue discovery*, *model invocation*, *model testing*, and *model elaboration*. Mackworth (1977a) has shown that this sequence can be found in most vision programs and that the programs can be characterized by the way they treat this sequence. Frequently, the sequence is gone through more than once and is therefore called *the cycle of perception*.

#### *Cue Discovery*

Three concepts are of importance in the interpretation process: primitives, cues, and models. Primitives are the elements in terms of which one seeks to represent the image at different levels of representation. Models serve as interpretations for primitives. Cues serve as mediators between primitives and models. Each cue constrains one or more primitives by suggesting one or more possible interpretations (models).

Cue discovery is the process that constructs primitives and cues from the input data given to the program. During cue discovery, all the image domain representations are constructed. Cue discovery is therefore equivalent to segmentation. The kind of primitives and cues used differs from program to program. In some programs lines are the only primitives (e.g., Clowes, 1971), but more commonly the lines and regions are the primitives (e.g., Guzman, 1968; Falk, 1972). In many vision programs the vertices serve as the cues by means of which the

scene domain models can be accessed.

#### *Model Invocation*

Model invocation is the process of associating the possible interpretations with each of the primitives. The levels of representation that can be accessed differs from program to program. If the cues are not given in the input (such is the case in Guzman (1968) and Clowes (1971)) the segmentation process culminates in their construction (e.g., Falk (1972); Mackworth (1977b)).

Some vision programs have only one level of representation in the scene domain (e.g., Clowes, 1971; Mackworth, 1977b). In such a case the cues have direct access to the models at that level of representation. Some more recently developed vision programs have more sophisticated access mechanisms; the level of representation in the image domain at which the cue is constructed determines the scene domain level that the cue accesses (e.g., Mulder, 1979; Browse, 1982).

#### *Model Testing*

Model testing is the process that tests whether the description of the model proposed for a primitive is consistent with the image description of the primitive. For instance, if a specific river is proposed as an interpretation for a certain line in a sketch map then the curved pattern specified for this river in its model description has to match with the curved pattern of the line.

### Model Elaboration

For an image to make sense as a whole, it is not sufficient that the model descriptions match the primitive descriptions; the possible interpretations for a primitive are usually constrained by more than one cue. Most of the cues also constrain more than one primitive. The problem of simultaneously satisfying the constraints imposed by the cues is called the *constraint satisfaction problem*. There exists a wide variety of algorithms that deal with this problem. As a very particular constraint satisfaction algorithm will also be involved in the solution of the ambiguity problem proposed in this dissertation, we devote a special section to constraint satisfaction.

#### 2.3.2.3. Constraint Satisfaction

A Constraint Satisfaction Problem (CSP) can be defined as follows:

Given  $n$  variables, each with a domain and a set of constraining relations find all possible  $n$ -tuples such that each  $n$ -tuple is an instantiation of the  $n$  variables satisfying the relations (Mackworth and Freuder, 1982). Algorithms that manipulate constraints come in many variations. The domain can be continuous or discrete, and the relations can be unary, binary, or  $n$ -ary. Discrete domains usually consist of a finite set of atomic labels.

In one of the early attempts to solve the constraint satisfaction problem, Huffman (1971) used a depth-first backtrack algorithm. This algorithm is

inefficient and has all problems inherent in depth-first backtrack such as thrashing (Bobrow and Raphael, 1974). Another problem with depth-first backtrack is that this algorithm is exponential in the domain size. In computational vision applications domain sizes can become very large.

Waltz (1972) devised a better solution. Before attempting a depth-first or breadth-first search Waltz applies a junction filtering procedure, whereby the cues (the junctions in his program) are visited in some order. For each of its edges, the junction interpretation list must provide an interpretation that matches at least one of the interpretations allowed for that edge by the interpretation list of the junction at the other end of the edge. Junction interpretations that do not match are deleted. If such a deletion occurs, then all junctions whose interpretations were constrained by the deleted junction interpretations are revisited; their interpretations are filtered to accommodate the new situation.

The advantage of this procedure is that a single pass through the junctions is sufficient. Thus, thrashing can be avoided. Although there is no guarantee that there will only be one interpretation left at each junction (and thus at each primitive) this procedure guarantees the removal of all locally inconsistent interpretations. However, Waltz's junction filtering procedure deals with binary consistency problems only. Mackworth (1977b) extended this algorithm to include  $n$ -ary consistency problems calling it a network consistency algorithm. Network consistency algorithms have the property of eliminating all local inconsistencies that cannot participate in a global solution. However, they do not solve the CSP.

Mackworth (1977c) emphasized the importance of three different forms of network consistency: node, arc, and path consistency. Node, arc, and path consistency algorithms eliminate all local inconsistencies that involve 1, 2, or 3 variables respectively. Among other features, arc consistency has a time complexity that is linear in the number of arcs, and polynomial in the domain size (Mackworth and Freuder, 1982). One particular arc consistency algorithm, AC-3 (Mackworth, 1977c), is cubed in the domain size. Node, Arc, and Path consistency algorithms are useful preprocessors for a Depth-first backtrack or divide-and-conquer algorithm because they have the effect of reducing the domain size (Mackworth, 1977c).

Freuder (1978) proposed a network consistency algorithm called  $k$ -consistency. His algorithm is based on removing all inconsistencies in all subsets of  $k$  out of  $n$  variables. ( $k \leq n$ ). This algorithm solves the CSP if  $k = n$ . Node, arc, and path consistency (Mackworth, 1977c) are actually special cases of this algorithm for  $k = 1, 2,$  and  $3$  respectively.

In some of the applications the labels have certainty values associated with them, in others they do not. However, labels without certainty values can be seen as a special case of labels with certainty values (Hummel and Zucker; 1983). Constraint satisfaction algorithms that use certainty values are often referred to as *Relaxation Labeling Algorithms*.

Rosenfeld *et al* (1976) introduced the idea of assigning weights to labels and relations to the field of computational vision. The weight of the labels is

iteratively updated by means of rules that take into account the compatibility with the labels of the neighboring variables. The global effect on the weight of a single label increases with each iteration.

Relaxation Labeling has been frequently used for line and curve enhancement (e.g. Zucker *et al* 1977). A multi-level approach to relaxation labeling was proposed in Zucker (1978). A survey of this kind of work has been provided in Davis and Rosenfeld (1981). In the scene domain, relaxation labeling has been used for updating hypotheses that are interconnected by composition and specialization hierarchies and a few other relations (Tsotsos, 1984).

Despite all the good features of algorithms that manipulate constraints, these algorithms have problems as well. One problem is, that constraints are only propagated, and actually finding the constraints themselves has to be done elsewhere. Another problem is that labels are treated as atomic elements without an internal structure. As a result, we cannot reason about labels which often represent objects in the interpretation process. These two problems have, among other things, been an important motivation for the development of schema-based vision programs to which we will turn next.

#### 2.3.2.4. Schema-based Computational Vision

The term schema is usually traced back to Bartlett (1932) and to Piaget (e.g., Piaget, 1967). Over the last few years schemata have served as a convergent

notion for knowledge representation research in both Psychology (e.g., Norman and Bobrow, 1976; Neisser, 1976) and Artificial Intelligence (e.g., Minsky, 1975; Kuipers, 1975; Freuder, 1976; Havens, 1978). At present, there is a wide diversity of definitions and characterizations of schemata.

Bobrow and Norman (1975), for example, define schemata as:

"active processing elements that can be activated from higher level purposes and expectations (model-driven), or from input data that must be accounted for" (data-driven).

One characteristic of schemata is that they can represent both declarative and procedural knowledge (Winograd, 1975). From a declarative perspective a schema can be embedded in different types of structural networks such as the specialization and composition hierarchy. A procedural representation, on the other hand, implies that a schema can assume control.

The control structure in most schema-based vision programs is hierarchically organized. A general interpreter takes the input data and constructs primitives and cues. Cues suggest different schemata as interpretations in different parts of the image. The interpreter has a choice between two modes of operation: bottom-up and top-down.

In bottom-up mode, the interpreter collects evidence by observing the cues in the image. Such evidence can be compared against domain specific knowledge

to constrain the number of possible interpretations. In top-down mode, on the other hand, the interpreter hands over control to a schema's procedure. Such a procedure orders the search space heuristically and searches for very specific evidence in specific parts of the image. Most schema-based vision programs can be characterized by the (sometimes very sophisticated) interplay between these two modes of operation.

Kuipers (1975) proposed a schema-based recognition model which is totally top-down. His program can recognize an object in the scene as belonging to one of three classes of objects: parallelepiped with three visible surfaces, wedges with two visible surfaces, and wedges with three visible surfaces. The interpreter starts with assuming the presence of one of the classes and hands over control immediately to the schema for that class.

Kuipers attempts to avoid thrashing behavior by using a complaint department; that is, if a schema fails to reach its objective it consults a similarity network which will recommend a replacement schema. The problem with such an approach is that this similarity network has to provide a replacement candidate for each possible failure situation. An unexpected situation not covered by the similarity network restores all the problems inherent in depth-first search.

Freuder (1976) designed a system that recognizes hammers. The knowledge about hammers is stored in a general knowledge (GK) network, whereas the knowledge specific for the hammer in the image (as it is built up during recognition) is stored in a particular knowledge (PK) network. Features found during a

segmentation process are used as bottom-up evidence for the existence of schemata (called conjectures by Freuder) of which a feature may be a part. The schemata suggested by features are installed in the PK network as hypotheses which may be explored in top-down mode. Such an exploration may result in the creation of other schemata.

Freuder's work is focused on the control structure for coordinating bottom-up and top-down methods. This control structure is based on a priority-queue multiprocessing scheme. When schemata are suggested by features in the image a priority is assigned to them. This priority can be changed during the recognition process. A global scheduler selects and invokes the schema with the highest priority. Successful exploration of a schema results in the hypothesizing of one or more higher order schemata of which the successfully explored schema was a part.

Although Freuder's approach marks an improvement with respect to previous systems his way of using a priority-queue multiprocessing scheme can be subjected to criticism, as was noted by Havens (1978). The problem is that a global priority is assigned to a schema by a local procedure that suggests a schema on basis of one feature found in the image. Havens argued that schema invocations should be pattern-based rather than based on some form of numerical priorities.

Havens (1978) has proposed a complete schema-based recognition model for machine perception. Schemata are characterized as:

"a modular representation of everything known about some concept, object, event, or

situation".

The knowledge associated with schemata can be represented both in declarative and procedural form. Two types of knowledge can be associated with schemata: factual and heuristic knowledge.

Factual knowledge can be represented both in declarative and procedural form. For instance, the embedding of schemata in a network of different composition and specialization hierarchies is a declarative representation of factual knowledge. Heuristic knowledge is represented in procedural form only. By means of its heuristic knowledge (also called methods), a schema guides the search process for the schema's concept.

Havens' recognition model consists of three stages: expectation, matching, and completion. Low level cues can suggest different schemata as a possible explanation. Each of these schemata has expectations associated with it, consisting of possible final instantiations of the schema. For example, a parallelogram can be used as a low-level feature to invoke a schema for a cube. The cube schema has to test whether its description is also made up by two other parallelograms in a particular configuration. As a result the description of the cube schema has to match the description of two other parallelograms. This expectation and matching process can be seen as an iterative recognition cycle.

However, Havens' perceptual model is cyclic in another sense as well. Once all the expectations of a schema are satisfied, the schema will seek completion;



that is, it will act as a cue for other schemata which are further up in the composition hierarchy in which the schema is embedded.

As is the case in Freuder's program, schemata can employ both model-driven and data-driven methods to perform the recognition process. In top-down mode a schema will invoke the methods associated with each of its expectations. These expectations are directly or indirectly verified in the image. In bottom-up mode, on the other hand, a schema is confronted with non-determinism; that is, the schema can be embedded in several composition hierarchies which means that there is more than one possible parent node. As a result, multiple hypotheses have to exist simultaneously. In bottom-up mode methods are therefore realized as concurrent processes.

For this reason Havens also has a multiprocessing scheme. However, invocation of a schema's method is not based on a global numeric priority. The central idea is that a schema's method will remain active until one of its expectations turns out to be difficult to prove. The schema then suspends itself by creating new expectations that describe its unrealized objectives. These expectations are stored as a pattern in a global database. The suspended method can be resumed as soon as another schema provides the kind of evidence the suspended schema is waiting for. Thus we can see that Havens' multiprocessing scheme is based on pattern-based invocation rather than numerical priority-based invocation.

Havens has developed a Lisp-based knowledge representation language, Maya (Havens, 1978), to deal with these issues. Maya provides a data structure

for representing objects and classes, a pattern matching facility, and facilities for parallel processing. Maya is one of the many schema-based languages that have been developed, most of which provide knowledge-structuring primitives and a number of processes that can operate on them. FRL (Roberts and Goldstein, 1977a, 1977b), the UNIT package (Stefik, 1979; Smith and Friedland, 1980), KRL (Bobrow and Winograd, 1977), and PSN (Levesque and Mylopoulos, 1979) are examples of such languages.

As mentioned before PSN was used in Tsotsos's work on motion analysis (e.g. Tsotsos, 1984). PSN formalizes traditional semantic network concepts in a procedural framework. Its primitives are classes and binary relations. Their semantics are defined by means of four basic operations: add, remove, fetch, and test. The knowledge structure is centered around "is-a" and "part-of" hierarchies. Both hierarchies can take part in the inheritance of properties. Another interesting aspect of PSN is the notion of a meta-class, a class of classes, which is used to explain certain features of the representation within itself.

In most schema-based vision systems domain dependency is introduced in the control structure. In particular, the heuristic knowledge associated with a schema is usually domain-dependent. Bajcsy and Joshi (1978) designed and implemented a system that forms an exception to this rule. Their system, which interprets natural outdoor scenes, has been implemented as a production system (Newell, 1973) with three components: a data base, a number of production rules, and an interpreter. The data base consists of *facts* which represent visual pro-

perties of the image (measured or derived). These *facts* are subdivided in two groups: short term *facts*, which are considered to be of immediate importance to the interpreter, and long term *facts*, which are not. The scene knowledge is encoded by means of production rules which are also subdivided in long term and short term rules. The scene objects are ordered by means of different relations such as "kind-of", "physical part-of", size, and distance. The rules can be clustered by means of the first two relations, the *facts* by the latter two. By using the concept of partial ordering of objects in combination with a production system methodology the system achieves its design goal: a systematic (domain-independent) control structure which can deal in an efficient manner with a visually rich domain. A general control structure is a well known feature of production systems (Davis and King, 1975). The clustering of rules and *facts* enable the system to apply relevant rules only.

#### 2.4. Interpreting Sketch Maps

Sketch maps have often been used as an aid in interpreting aerial or satellite photographs. They can be seen as a simplified representation of the photograph. Often we can recognize a particular area just by looking at the sketch map. Sketch maps are therefore useful in two respects. First, they carry many characteristics of the original image and can be used as an aid in interpreting the original image. Second, a significant number of the perceptual problems associated

with interpreting the original image are present in the interpretation of sketches. The automatic interpretation of sketches is therefore by itself a worthwhile goal.

The HAWKEYE system (Tenenbaum *et al* 1978; Bolles *et al* 1979) is an example of a system that uses a map database as an aid in interpreting aerial photographs. After establishing geometric correspondence between an aerial image and a symbolic map, information from the map is used to guide interpretation of the former. This technique has been successfully applied to such tasks as monitoring the volume of water in a reservoir and monitoring the number of box-cars in a railyard. Glicksman's (1982) work in this area has already been mentioned in section 2.2.3.

The MAPSEE project at U.B.C. is one of the projects which has adopted the problem of sketch map interpretation. Mapsee-1 was developed by Mackworth (1977b). Line segments and regions are the primitives in this program. The vertices are the cues. All of these are created by a segmentation process. The cues are all ambiguous. For instance, both the bar and stem of a T-vertex can be a river and all surrounding regions can be land. Another possibility is that the bar is a shore and the stem is a river. The regions adjacent to the stem now have to be land, but the third region adjacent to the bar is water. A network consistency algorithm is used to reduce the possible label set.

The scene domain knowledge in Mapsee-1 was represented at one level only. As a result it was impossible to speak about such concepts as road-systems and geo-systems. Furthermore, the labels were represented as labels only, lacking any

modularly organized internal structure by means of which one could reason about them. The former problem was resolved in a system with a stratified knowledge base that could interpret line sketches of houses (Mulder, 1979). The latter problem was resolved by representing models as schemata (Havens, 1978). A schema-based approach formed the foundation of Mapsee-2 (Havens and Mackworth, 1983; Havens, Mackworth, and Mulder, 1985).

The objects in Mapsee-2 are described by means of their attributes. Thus, each object "knows" what its components are and what it is part of. The objects are embedded in both a composition and a specialization hierarchy. Figure 2.1 A and B show these hierarchies. Each of the primitives (line-segments and regions) gets represented at each level of composition and specialization by means of two processes: composition and specialization.

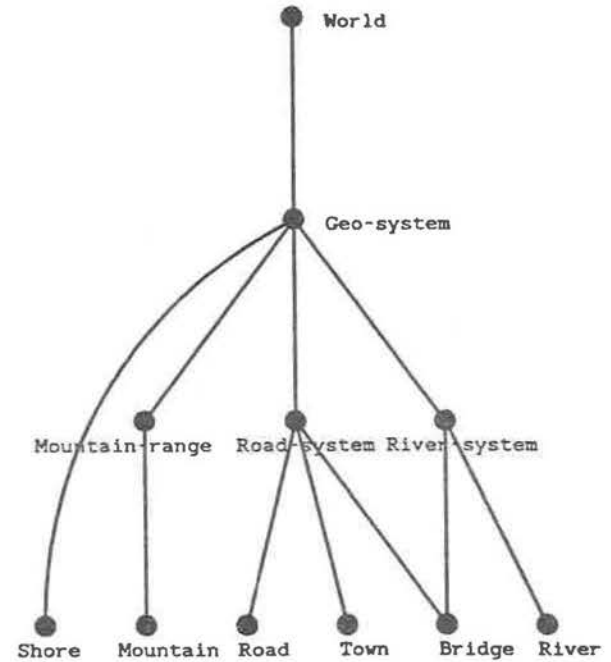


Figure 2.1A: The Mapsee-2 composition hierarchy

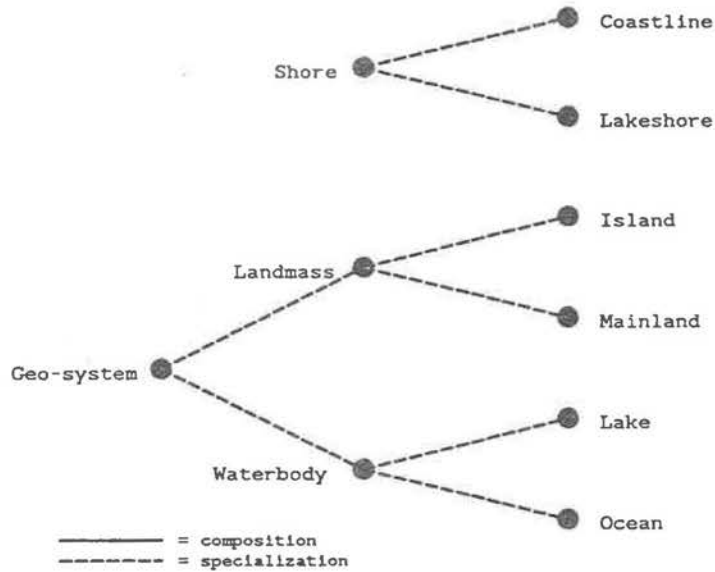


Figure 2.1B: The Mapsee-2 specialization hierarchies

Composition is achieved by a bottom-up, depth-first approach. Image cues suggest one or more interpretations at the leaves of the composition hierarchy. If there is more than one interpretation for a primitive then the interpretation is considered hypothetical. By means of composition, each hypothesis becomes a component of an instance of a schema one or more levels up in the composition hierarchy. For instance, each river instance has to become a component of a

river-system instance which in turn becomes a component of a geo-system instance etc. Composition is a very complex process, as it simultaneously has to deal with both the proper matching of instances at different levels of composition and the maintenance of mutual consistency between different hypotheses. This process is described in detail in Havens *et al* (1985).

In composition, objects are treated as schemata. The composition process has to include looking at the schema's attributes in order to find its super-components. Specialization, on the other hand, is dealt with as a Constraint Satisfaction problem in which objects are treated as labels. There is a specialization hierarchy for each schema, whereby each schema-instance has a label as one of its attributes. This label can be any node in the specialization hierarchy in which the schema is embedded. For instance, a geo-system instance can have the label island, which implies that the instance has been specialized to be an island (Figure 2.1B).

Specialization in Mapsee-2 is a top-down process. At the start each schema instance has the name of its schema as label. This label forms the root of a specialization hierarchy. Specialization takes place incrementally. Each time a new component is added during composition the validity of its label is tested. For instance, the label of a geo-system with a mountain-range as a component is specialized to be a landmass. If it then turns out that the geo-system is surrounded by a shore, the label is specialized to island.

In Mapsee-2, specialization is a special form of Arc Consistency, called *Hierarchical Arc Consistency*. In the constraint graph, the variables are the instances and their domain the specialization labels which are hierarchically organized. Hierarchical Arc Consistency is implemented in a procedural way. As a similar algorithm has been used in Mapsee-3, we will postpone the discussion of this algorithm until Chapter 3. The Mapsee-2 program interpreted sketch maps for the MISSEE program (Glicksman, 1982).

### 2.5. The Ambiguity Problem

Image cues tend to be highly ambiguous; that is, one cue suggests many possible interpretations for the primitive(s) constrained by the cue. Most of those interpretations tend to be only locally consistent. Once the consistency requirements over a wider area of the image are considered, the number of possible interpretations for each primitive will usually be reduced to one. The problem of how to reduce the large set of interpretations that are locally consistent to a smaller set that are globally consistent is an important aspect of the interpretation problem as defined in section 2.3.2.1. A careful consideration of the problem shows that it can be described in two different ways: one way is to view it as a problem of representing interpretations that are *hypothetical*; the other is to look at it as a problem of representing interpretations that are *ambiguous*.

### *The hypothetical point of view*

Figure 2.2 shows a closed line-segment that could serve as a cue for two geographic objects: a coastline and a lakeshore. We can assume that these objects are embedded in a Mapsee-2 like composition hierarchy such as the one in Figure 2.3. This figure demonstrates the *hypothetical* point of view, which states that every locally consistent interpretation forms a hypothesis. As we start to consider larger areas in the image, we have to explicitly maintain some data structure that tells us which hypothetical interpretations go together with those in adjacent areas and in what way. By means of the process of constraint propagation we will be able to gradually eliminate certain hypotheses and the data structures in which they are embedded.

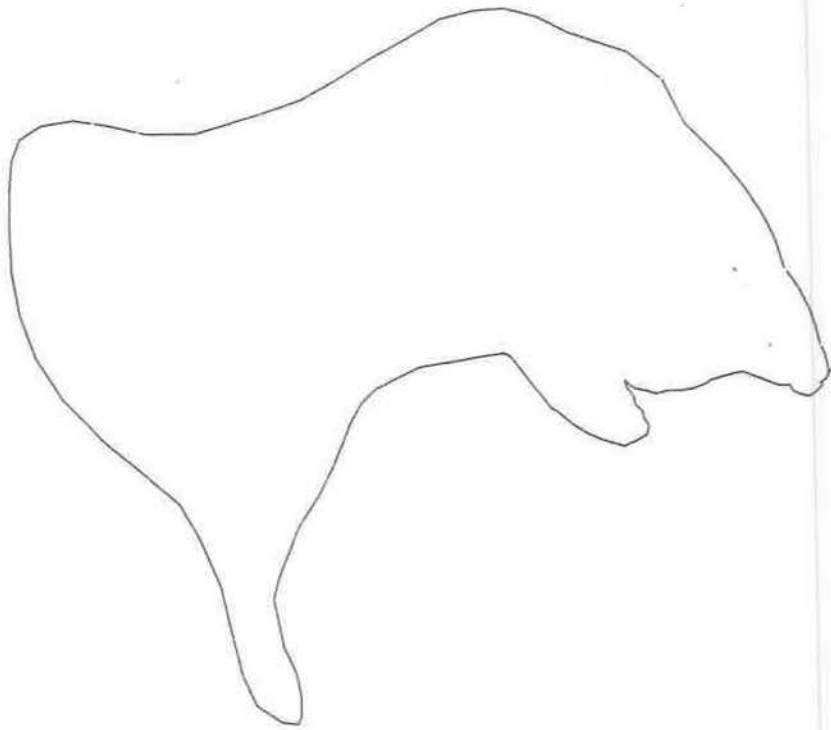


Figure 2.2: A closed line segment depicting a shore

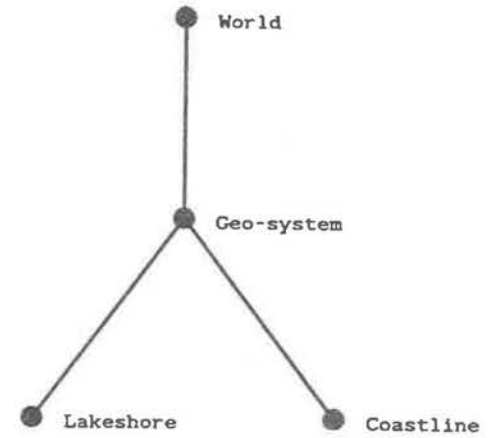


Figure 2.3: A simple composition hierarchy

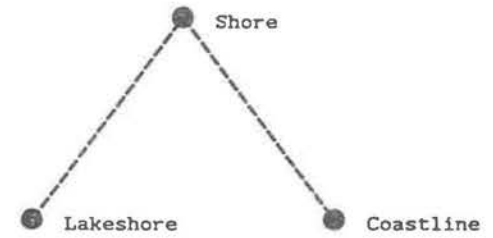


Figure 2.4: A simple specialization hierarchy



*The ambiguity point of view*

In Figure 2.4 we have introduced a specialization hierarchy. This enables us to use the *ambiguity* point of view stating that a coastline and a lakeshore are both a specialization of a more generic interpretation, a shore. Regardless of whether the primitive will turn out to be a coastline or lakeshore, we always know it must be a shore. Thus, in the ambiguity approach the image cue will constrain the primitive to be a shore and since this is the only possible interpretation, it is not hypothetical. Only after constraining evidence has come from adjacent areas in the image is the interpretation specialized into a lakeshore or coastline.

The ambiguity point of view is strongly linked with the *principle of least commitment* which was introduced in computational vision by Marr and Nishihara (1976). It means that we stick to the most abstract possible interpretation until evidence from the image forces us to move towards a more specific interpretation. This principle is also reflected in human visual perception. Humans do not generally interpret an image in terms more specific than the circumstances require.

The example shows that ambiguity can be represented in the form of a *specialization* hierarchy. The example was concerned with a two level hierarchy only. A representation in terms of the top node in the hierarchy (shore) is ambiguous but non-hypothetical, whereas a representation in terms of the leaf nodes (coastline and lakeshore) is unambiguous but hypothetical. With multiple level

specialization hierarchies, we can create many different mixtures of the two approaches. The *hypothetical* point of view and *ambiguity* point of view can therefore be seen as the extremes in a specialization/generalization dimension, and we can use this dimension as a criterion for comparing different vision systems.

Vision programs that have no specialization hierarchies automatically end up at the hypothetical end of the scale (e.g. Freuder, 1976). Most programs with specialization hierarchies, on the other hand, are neither extremely hypothetical or ambiguous, but somewhere in between.

An object in a specialization hierarchy, when suggested by a cue as a possible interpretation, is non-hypothetical only if no other objects are suggested by the same cue. The cues in ACRONYM are all ambiguous and the interpretations suggested by them are all hypothetical. Mapsee-2 (Havens *et al* 1985) is somewhat different in this respect, because some cues are ambiguous, although others are not.

The vision programs with specialization hierarchies differ strongly in the way they use this hierarchy in the interpretation process. In Mapsee-2 there are some cases in which a model at the top of such a hierarchy is suggested as a non-hypothetical interpretation for a primitive in the image. As soon as more of the primitive's context becomes known, a gradual specialization of the interpretation takes place toward one of the leaf nodes in the hierarchy. The majority of



interpretations, however, are hypothetical.

Browse (1982) uses cues from different levels of resolution in the image to access different levels of specializations. All interpretations are hypothetical. Interpretations suggested at a coarse level of detail have to agree with those suggested at a fine level of detail. Browse uses the specialization hierarchy to bring the two sets into agreement.

Tsotsos (1984) uses specialization hierarchies in the CAA program (Shibahara *et al* 1982; Shibahara *et al* 1983), but they are not used to reduce or eliminate hypothetical interpretations. All class instantiations in his system are hypothetical. Alternative hypotheses can be tested through similarity links (Tsotsos *et al* 1980). In a case of incorrect prediction, the system moves upward along the specialization hierarchy in order to remove the constraints from the failing hypothesis.

In ACRONYM a class of objects is suggested as a possible interpretation for parts of the image. One of the uses of the specialization hierarchy in ACRONYM is to determine whether one of the sub-classes of this class can also serve as a possible interpretation. If this is not the case, the hypothesis is considered to be false. Although no one will argue that specialization hierarchies can be used for representing ambiguity, one may wonder whether the same hierarchy should be used to determine the correctness of hypotheses. The level of specialization at which an object can be described depends entirely on what and how much of the object is visible in the image. If not enough information is available to verify the

correctness of a sub-class then ACRONYM will reject a possibly correct hypothesis.

## 2.6. Discussion

The "high level" vision literature has been looked at from several points of view. With an emphasis on knowledge representation, the following aspects were highlighted:

1. Computational vision systems require a flexible knowledge base that can deal with an image at different levels of detail and specificity. A multi-level representation along two dimensions: a composition/aggregation dimension which enables the system to interpret an image at different levels of detail, and a specialization/generalization dimension which enables the system to interpret an image at different levels of specificity, provide the system with such flexibility.
2. There is a need to describe objects in at least two different ways:
  - a. as labels which form the domain in a CSP.
  - b. by their internal structure.

The schema-based systems provide the ability to do both.

3. Is-a hierarchies vary widely in their interpretation. The discussion focused on the fact that specialization hierarchies often appear to be playing a role in the representation and resolution of ambiguity.

4. Hypotheticality and ambiguity can be seen as the extremes in a specialization/generalization dimension. There appears to be no "high-level" vision system at the ambiguity end of this dimension.

With the representation of ambiguous and hypothetical interpretations as the central topic of this dissertation the last observation is very interesting, and several possible reasons suggest themselves as an explanation of why this is the case.

One possible reason is that model-based research in computational vision has been concentrating on the representation of objects and their interrelationships. Hypothetical interpretations, although recognized as a problem, have been dealt with more as a side issue. As a result, solutions to the problem are usually implemented in a procedural way. The data structures that actually represent hypothetical interpretations are part of the *temporary* database created during construction of an image interpretation. Yet, if we want to be able to reason about possible ambiguities in interpretations then we need to represent the knowledge about these ambiguities in a declarative form and in the *permanent* database of the system.

Figure 2.5 suggests a second possible reason. This figure shows a situation in which a closed line-segment has three possible interpretations: a coastline, a lakeshore, and a road. A coastline and lakeshore can be generalized to a shore, but for a shore and a road no natural categorization exists. All Vision systems discussed use natural categorization only. However, image features often suggest interpretations which do not fit together in natural categories. For instance, a collection of green pixels is extremely ambiguous. They can depict a golf course, farm land, a park, forest, or even the astroturf in a stadium. In order to take an *ambiguity* approach one will often have to exceed the boundaries of natural categorization. In Figure 2.5, for instance, we have created the unnatural concept road/shore.

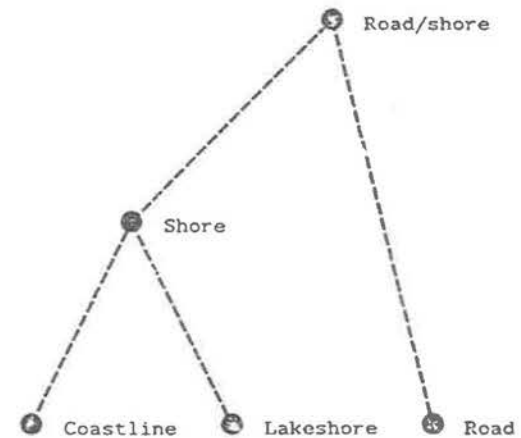


Figure 2.5: An unnatural specialization hierarchy

A third possible reason is that as a result of noisy data the image features may be unreliable. In such a case we can consider them to be hypothetical. Consequently, all model invocations would necessarily be hypothetical as well. This aspect, however, stands apart from the previous two arguments. Even in an ideal segmentation that leads to "correct" features only the *ambiguity* problem still exists.

The last reason forms a good justification for using a hypothetical approach, although we can criticize this approach for containing some undesirable features.

1. Systems that use specialization hierarchies for representing ambiguity but which do not fall at the ambiguity end of the scale violate the conceptual clarity criterion of modularity in representation. Part of the possible interpretations for an image feature are now represented along the composition/aggregation dimension as hypothetical interpretations, whereas another part is represented along the specialization/generalization dimension as ambiguous interpretations.

2. Modularity in control is also affected. In a system with modular control, the processes operate on one particular knowledge representation dimension only. As a result of the fact that ambiguity is spread over more than one knowledge representation dimension, the process that deals with ambiguity has to access more than one knowledge representation dimension as well. In section 5.3.4 we will discuss a Mapsee-2 example and compare it with Mapsee-3.

3. Each possible interpretation for a primitive has to be explicitly represented and

pursued as hypothesis. The number of possible interpretations for a single primitive is often very large.

4. Each set of interpretations that cannot be "joined" into one (abstract) interpretation by means of a specialization hierarchy needs to be maintained in an interpretation graph by different variables. As already mentioned in Chapter 1, competing hypotheses are thus represented by different variables. Each time a hypothesis is invalidated, we have to restructure the interpretation graph. This can be a complex and cumbersome operation. In particular, Mapsee-2 suffered from that problem.

All these problems can be alleviated if we construct a knowledge representation dimension whose sole purpose is the representation of all possible ambiguities in interpretation. This knowledge representation dimension can take the form of a discrimination graph such as the one illustrated in Figure 2.5. Depending on how we construct this graph we can build a vision system with a hypothetical approach, an ambiguity approach, or anything in between. Additionally, discrimination graphs offer the following advantages:

1. Different hypotheses and their mutual consistencies can now be represented in the form of an explicit, declarative data structure, which is part of the permanent knowledge base. Such a data structure can form a knowledge representation dimension by itself, thus achieving modularity in representation.

2. Modularity in control is also achieved because we now need one process only to operate on each knowledge representation dimension. A composition process will

operate along the composition/aggregation dimension. This process is considerably simplified as a result of the removal of the hypothetical element. The discrimination process operates along a discrimination/generalization dimension. This process deals with ambiguity. Moreover, it can now be completely formalized as a network consistency algorithm. We will discuss this in the next chapter.

3. Discrimination graphs are by nature hierarchical. This enables us to represent a set of elementary interpretations by means of one abstract interpretation. We can thus reduce the number of interpretations that need to be represented for a single primitive.

4. Discrimination graphs allow us to provide a unique (abstract) interpretation for each set of interpretations that an image primitive can depict. Hence, competing hypotheses can be represented in the domain of a single variable in the interpretation graph which represents the current state of interpretation of the image. Invalidation of a particular hypothesis now only requires a deletion or replacement of a label in the domain of a variable. This can be achieved without changing the structure of the interpretation graph.

In the next chapter we describe the design of Mapsee-3, a system that uses discrimination graphs. The design is in the spirit of the Mapsee-2 system. Object classes and relations are represented as schemata: modular units of declarative and procedural knowledge. Schemata are embedded in both composition and

discrimination graphs. Attached procedures have the ability to search for and, if found, create spatial relationships between their components. Composition will be seen as the process that constructs a network of constraints. Discrimination will then be viewed as a network consistency process that maintains consistency in the network.

### 3. DESIGN OF MAPSEE-3

#### 3.1. Introduction

In this chapter we describe the design of Mapsee-3, a schema-based sketch map interpretation program. Although the program was designed for the interpretation of sketch maps, many of its design principles are of a more general nature. In this chapter, we focus on the design principles of the system. The actual implementation of the system is discussed in Chapter 4.

The most important features of the program are:

1. A declarative and domain-independent data structure for representing ambiguities that can arise when we map image primitives to scene interpretations. This data structure takes the form of a discrimination graph. A motivation for the desirability of such a structure was given in the previous chapter.
2. Two knowledge representation dimensions along which knowledge about scenes can be represented. A composition hierarchy of object classes forms the core of a composition/aggregation dimension. At each level of composition one or more discrimination graphs form a discrimination/generalization dimension which is orthogonal to the composition/aggregation dimension.
3. Algorithms that automatically construct parts of the scene's knowledge base.

4. A composition process that operates in a data-driven manner on the composition hierarchy. At first, image primitives are mapped into interpretations that form the leaves of a composition hierarchy in the scene domain. Gradually, these interpretations are transformed into more aggregated interpretations which are represented at intermediate levels in this hierarchy. This process of aggregating interpretations continues until the top of the hierarchy has been reached.

5. A network consistency algorithm that operates on discrimination graphs. In the previous chapter we observed that algorithms of this type have been well researched and they can be more efficient than depth-first backtrack.

In the design, great emphasis has been put on conceptual clarity, including:

1. Modularity in representation. Objects and relations are represented as schemata. The knowledge representation dimensions used are orthogonal.
2. Modularity in control. There is only one process that operates in each knowledge representation dimension, a process which can alter only data structures in the dimension in which it operates. It cannot directly invoke a process operating along another dimension. Each Mapsee-3 process owns a queue from which it takes its input. Different processes intercommunicate by putting items on each other's queue.
3. Uniformity. Each schema type has a fixed representational format. A grammar is provided for this representation.

4. Strict separation between domain-dependent and domain-independent knowledge. The schema, the knowledge representation dimensions and the processes operating along them form a domain-independent format for representing knowledge about different scenes.

The design also focuses on efficiency. Many computational vision systems (including Mapsee-2) create interpretation graphs in which the variables (nodes) represent hypothetical interpretations. Such graphs require continuous restructuring when hypotheses are invalidated. In Mapsee-3, different hypotheses are maintained in the domain of a *single* variable. The nodes and links represent non-hypothetical interpretations and constraints only. As a result of the algorithms used, invalidation of a hypothesis requires only deletion or replacement of a label in the domain of a variable. The structure of the interpretation graph remains unaffected by this operation. Other efficiency measures result from the use of hierarchical constraint propagation in the system.

The Mapsee-3 knowledge base has been designed to operate ideally in a signal processing environment. The input to such an environment consists of one or more signals whose intensity varies over time or space. The objective of the system is to segment the signal(s) into primitives, segmenting in such a way that each primitive can be interpreted meaningfully. A wide variety of domains can be looked at as signal processing environments, including areas such as the interpretation of sound waves, and spectral analysis, in addition to computational vision.

In its current implementation, however, Mapsee-3 is somewhat limited because of the assumption that the domain can be ideally segmented; that is, we assume no noisy data. As a result, we know what the primitives are, and we know that they are correct. So we do not have to deal with one of the causes of ambiguity noted in the previous chapter. It was observed that ambiguity is caused by at least two different factors: a segmentation process that has to deal with noisy data, and image primitives which are underconstrained when it comes to interpretation. In this dissertation, we are concerned only with underconstrained image primitives.

In many respects Mapsee-3 is a sequel to Mapsee-2, the schema-based sketch map interpretation program which was summarized in the previous chapter. For example, the Mapsee-3 composition hierarchy is similar to the one used in Mapsee-2 (see Figure 2.1A). The main point of departure from Mapsee-2 is the use of discrimination graphs which form a knowledge representation dimension by themselves. This enables Mapsee-3 to be more modular in representation and control than Mapsee-2.

The Mapsee-3 interpreter operates in a data-driven manner. After a segmentation process that constructs images, an image-to-scene mapping process maps primitives into object classes which are the leaves of a composition hierarchy in the scene domain. Thereafter, interpretation is guided by two modular processes: composition and discrimination. Composition ensures that each primitive is subsequently represented at each level of composition, starting at the leaves of the

hierarchy and gradually working to the top. Discrimination graphs, on which the discrimination process operates, express possible refinements of interpretations at each level of composition. Using the principle of least commitment, this process uses the spatial relations found by the composition process as a way of refining the interpretation of each primitive.

Composition can also be considered as the process that searches for spatial constraints between primitives. The constraints found are represented in the form of an interpretation graph. Discrimination is then the process that propagates these constraints over this graph until a consistent situation is arrived at. In Mapsee-3, discrimination has been implemented by means of a network consistency algorithm which uses the principle of least commitment.

In section 3.2 we explain the representation of the system. We discuss the unit of knowledge representation, and the knowledge representation dimensions. In section 3.3 we discuss the three stages of the Mapsee-3 control: segmentation, image-to-scene mapping, and interpretation. With the focus on interpretation, we discuss the two components of this process: composition and discrimination. The Mapsee-3 knowledge base can be subdivided into a natural and an unnatural constituent. While the former must be provided by the user, the latter can be constructed automatically from the former. The construction algorithms are discussed in section 3.4. A summary of the design is given in section 3.5.

## 3.2. Representation

### 3.2.1. Schemata

The unit of representation in Mapsee-3 is the schema. A schema in Mapsee-3 is a list of attribute-value pairs which describe the schema's internal properties and its constraints on other schemata. Two categories of schemata are distinguished: image schemata and scene schemata. The first category can be subdivided in six (somewhat domain-dependent) classes: points, links, lines, chains, patches, and regions. The second category is subdivided in two (domain-independent) classes: object classes and relations.

In most schema-based vision systems, only object classes are represented as schemata. Relations are generally represented by the schema's attributes. The reasons for representing relations as schemata are twofold:

- 1) It leaves the user-implementer with the choice of creating an object- or relation-based system. This is particularly important with respect to procedural attachment. In Mapsee-3, the user can decide to which kind of schemata to attach part of the control.
- 2) Relations can be of any arity, while the constraints with other schemata remain binary. In object-based systems, it is more difficult to represent higher order relations.

As this dissertation focuses primarily on the representation of scene knowledge, we describe only the scene schema and its attributes. The reader is referred to Appendix D for a complete description of all classes of schemata and their syntax. The following attributes can be found in every scene schema  $X$ :

- 1) A schema-label: uniquely identifies an object class or relation to the system (e.g. \*S99).
- 2) type: indicates whether the schema represents an object class or relation.
- 3) composition level: each scene schema is embedded in a composition hierarchy.
- 4) discrimination level: each schema is also embedded in a discrimination graph, the concept of which is explained in section 3.2.2.
- 5) links-in: the list of schemata which have pointers directed at  $X$ .
- 6) links-out: the list of schemata to which  $X$  has pointers.
- 7) mandatory components: the list of schemata that enter in a "must-be-parts" relation with  $X$ .
- 8) other components: the list of schemata that enter in a "may-be-parts" relation with  $X$ .
- 9) mandatory super-components: the list of schemata that enter in a "must-be-part-of" relation with  $X$ .
- 10) other super-components: the list of schemata that enter in a "may-be-part-of" relation with  $X$ .
- 11) discriminations:  $X$ 's successors in the discrimination graph.
- 12) generalizations:  $X$ 's parents in the discrimination graph.

13) methods: a schema can represent both declarative and procedural knowledge. The schema's methods are procedures which are "owned" by the schema. Each Mapsee-3 method consists of a function that takes one or two arguments. A more detailed explanation of the operation of methods is provided in section 3.3.1.

14) instances: during interpretation each schema can be instantiated zero or more times. Each instantiation is represented as a uniquely identifiable unit. A scene schema instance  $Y$  has the following attributes:

- a) instance label: uniquely identifies  $Y$  to the system.
- b) iinverse: Each relation in Mapsee-3 has an inverse. If  $Y$  is an instantiation of a relation, then it must have an inverse which is an instantiation of the inverse of  $Y$ 's parent schema.
- c) parent: the schema  $Y$  is an instance of.
- d) ilinks-out: the instance equivalent of "links-out".
- e) ilinks-in: the instance equivalent of "links-in".
- f) icomponents: the established components of  $Y$ .
- g) isuper-components: the established super-components of  $Y$ .
- h) labels: the list of current interpretation(s) of  $Y$ . At the time of creation  $Y$  inherits the label of its parent  $X$ . This label can be replaced by any of  $X$ 's successors in the discrimination graph, if the situation requires it.
- i) idepicted-by: each schema instance is depicted by one or more image primitives.

15) inverse: see iinverse.



16) depicted-by: see idepicted-by

In Mapsee-3 schemata never point directly at other schemata of the same type. The only exception to this rule is for the discrimination and generalization attribute. These attributes are taken to be internal properties of the schema. A concrete example from the sketch map world illustrates the constraints (see Figure 2.1A and B). A geo-system, for instance, is a mandatory component of the world; that is, the geo-system schema points at a "must-be-part-of" schema which, in turn, points at the world schema. The "must-be-part-of" schema has an inverse which represents a "must-be-parts" relation. The latter serves as an intermediary between the world and the geo-system schema. The geo-system's discriminations, on the other hand, are the landmass and waterbody schemata, both of which are object classes.

Mapsee-3 differs from Mapsee-2 in its representation of the scene knowledge. In Mapsee-2 some schemata are treated as labels, whereas others are treated as object classes with an internal structure. In Mapsee-3 each schema can be treated either way. The representation of image schemata, on the other hand, is very similar to the one used in Mapsee-2 (see Appendix D for a syntactic description).

### 3.2.2. The Knowledge Representation Dimensions

The knowledge base of which the schemata are a part is organized along three dimensions: composition/aggregation, discrimination/generalization, and a dimension in which the relations are represented that map image primitives into scene interpretations and *vice versa*.

#### - Composition/Aggregation

Schemata are embedded in a Composition hierarchy by means of a "parts" and "part-of" relation. The former points the schema to its components, the latter at its super-schema(ta) in the hierarchy. Composition relations can be either of the "must-be" or "may-be" kind provided the distinction is explicit. The composition hierarchies in Mapsee-3 occur either in the image or in the scene domain, but they never cross the image/scene boundary. The Mapsee-3 composition hierarchy in the scene domain is based on spatial or time relationships. Two schemata in such a relationship must have a common super-schema further up in the hierarchy.

We can also look at a composition hierarchy from a different point of view. Together with the relations on which it is based, it forms a graph. Each node in this graph is a schema, and each link is a predicate. The graph forms a closed world in the sense that it contains all and only those predicates that are true. Any predicate which is not in the hierarchy is false in this world. Figure 2.1A, for

instance, shows an example from the sketch map world. *Part-of (shore, geo-system)* is true, but *Part-of (mountain, road-system)* is false. The graph can therefore be seen as a model constraint graph. The schemata are the variables, their label the domain.

An interpreted image with schema instances at different levels of composition consists of an interpretation graph in which the nodes are schema instances and the links represent constraints between these instances. Each image primitive is represented by one or more instances at different levels of composition. For instance, in Figure 2.1A an image primitive interpreted as a mountain will also depict an instance of a mountain-range at the next level up, and an instance of a geo-system at the level beyond that. These instances are all part of an interpretation graph. This graph is constructed by a composition process that operates along the composition/aggregation dimension. We will discuss this process in the control section of this chapter.

#### - Discrimination/Generalization

Image primitives may be ambiguous when it comes to interpreting them. Suppose a particular image primitive  $p$  can be interpreted by each one of two different schemata  $A$  and  $B$ . In the hypothetical approach, we would have to instantiate each of these schemata as a possible hypothesis. The ambiguity approach, on the other hand, requires at least one additional schema that can represent  $p$  in a non-hypothetical way. This schema,  $A/B$ , represents intension-

ally the set consisting of  $A$  and  $B$ .

Figure 3.1 shows a simple OR graph consisting of  $A/B$  and its elementary schemata. We call this graph a *discrimination graph*. Each link can be thought of as the constraint necessary to specialize  $A/B$  into one of the schemata it intentionally represents. If we follow the principle of least commitment, then  $p$  will be represented at first as an instance of  $A/B$ . If a constraint represented by one of the arcs emanating from  $A/B$  is found then  $A/B$  will specialize into the schema at the tail of the arc.

The structure of the discrimination graph determines whether the system can use the ambiguity approach at all times during the interpretation process. For instance, by means of the discrimination graph in Figure 3.1 the ambiguity approach can be used to interpret  $p$ . This is the case, because both the possible interpretations for  $p$  and all their possible combinations are explicitly represented in the graph. Figure 2.5 is an example of a discrimination graph for which this is *not* the case. In this example,  $p$  is a closed line segment which depicts either a road, coastline, or lakeshore. Initially  $p$  can be interpreted as an instance of a road/shore schema. However, if at a certain point in the interpretation process, the constraints are such that lakeshore and road are the only possible interpretations then we are in trouble. There is no schema which

uniquely represents the combination road/lakeshore. As a result we have to represent  $p$  as a road/shore with two possible (hypothetical) interpretations.

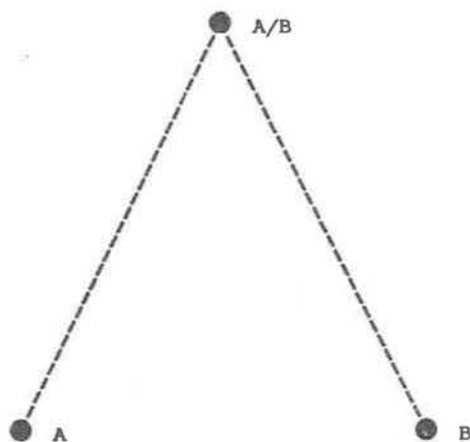


Figure 3.1: A simple discrimination graph

The structure of the discrimination graph thus determines what the approach to the ambiguity problem must be. If  $p$  has  $n$  possible interpretations and we construct a two-level discrimination graph with a source node which intensionally represents the  $n$  interpretations then we opt for an approach which is mainly hypothetical. If, on the other hand, we construct a discrimination graph which not only represents the  $n$  interpretations but all possible combinations as well, then we opt for the ambiguity approach. As well, we can construct a graph which constitutes a compromise between the two approaches. In Chapter 4 we will see that Mapsee-3 does the latter.

The idea of discrimination graphs is not new. They are identified in early AI systems such as EPAM (Feigenbaum, 1963; Simon and Feigenbaum, 1964). In Mapsee-3, their use is therefore not novel in concept, but in application: the representation of visual interpretations that are hypothetical and ambiguous. The term discrimination was chosen because of its application. In Mapsee-3, the graph is used to visually discriminate between different schemata. Because context is used as a discriminating factor, one way of looking at the discrimination graph is to see it as a similarity graph. The links in the graph represent similarity in appearance between the schemata represented. If we replace  $A/B$  in Figure 3.1 by the schema *shore*,  $A$  by *coastline*, and  $B$  by *lakeshore* then we have created the discrimination graph shown in Figure 2.4. In sketch maps a shore, lakeshore, and coastline are all depicted by a closed line segment. Only by taking the context into consideration can we discriminate between a coastline and lakeshore. If the shore surrounds a landmass it becomes the former, if it surrounds a waterbody it becomes the latter. Thus, discrimination comes close to what Brachman (1982) calls "value restriction". However, discrimination graphs distinguish themselves from specialization hierarchies in that there is no universal implication. An object class may appear in many different ways in the image. In the sketch map world, for instance, a road may be depicted by many differently shaped line segments. The line segment may be closed, or it may run off the picture frame on one or both sides.

The Mapsee-3 interpreter expects each discrimination graph to be orthogonal to the composition hierarchy; that is, each discrimination graph is located at a

particular level of composition. This is done to keep the system as modular as possible. Orthogonality can be easily achieved because of the additional requirement that the leaf nodes of the composition hierarchy are equidistant from the top node. For composition hierarchies that do not satisfy the latter requirement, we can create dummy schemata at the missing levels. Although these dummy schemata are essentially undefined interpretations, the system will treat these schemata as being redundant, and it can disregard their presence.

As mentioned before, a schema instance's interpretation is expressed by one of the instance's attributes, called its label. The main advantage of such a representation is that refinement of interpretation of a particular primitive does not result in a restructuring of the interpretation graph. For instance, in Figure 2.1B, a primitive depicting a shore is represented in the interpretation graph as an instance of a shore and is labeled *shore* as well. If the interpretation needs to be refined to *coastline* then all we need to do is replace the label *shore* by the label *coastline*. This leaves the value of the composition attributes of the instance unchanged. Thus, the composition and discrimination process operate independently from each other. This results in modularity in control, which the Mapsee-2 system did not have.

A last point with respect to discrimination graphs is that many nodes do not represent natural object classes. The road/shore object class in Figure 2.5 is such an example. The graph therefore consists of a natural and an unnatural constituent. The subgraph containing the shore, coastline, and lakeshore forms the

natural constituent in Figure 2.5. Natural constituents have to be provided by the user. However, in order to take the burden away from the user, we can provide algorithms by means of which we can automatically construct unnatural constituents. In section 3.4, we will discuss the algorithms for constructing the unnatural constituents in the discrimination graphs of the Mapsee-3 system.

#### - *Image-to-scene mapping*

As mentioned before, composition hierarchies do not cross the image/scene boundaries. The connection between image and scene is provided by the relations "depicts" and "depicted-by". In general, image primitives constructed at different levels of detail in the image can depict schemata at different levels of composition in the scene, as in the system created by Browse (1982). In Mapsee-3, image primitives depict schemata only at the composition leaf level in the scene. At this level each schema instance is depicted by one image primitive.

### 3.3. Control

In its most rudimentary form, Mapsee-3 is a sequence of three processes: *segmentation*, *image-to-scene mapping*, and *interpretation*.

Segmentation is the process by which primitives are created from input data. These primitives serve as a basis for interpretation and they are presumed

correct. We thus avoid a number of low level vision issues which, as mentioned before, this dissertation does not address.

Discrimination graphs are based on a categorization of image primitives with respect to a particular characteristic (e.g. shape). Image-to-scene mapping puts each image primitive in such a category and it creates an instance of an abstract schema at the composition leaf level which uniquely represents the particular category to which the primitive belongs.

This dissertation focuses on the interpretation process. It ensures that each image primitive is represented at all levels of composition and with an appropriate interpretation. An interpreted image consists of an interpretation graph in which each node represents a schema instance and each link a constraint between two different instances. Because a schema can be an object class or a relation of any arity, the graph is actually a super-graph.

Interpretation consists of two processes: *composition* and *discrimination*. Composition represents each image primitive at all levels of composition, while discrimination ensures that each image primitive is represented at an appropriate level of discrimination at all times. As mentioned before, another way of looking at composition and discrimination is to see composition as the process that constructs the interpretation graph and discrimination as the process that propagates consistency over this graph.

Both composition and discrimination are modular processes with composition being the only process, that operates in the composition/aggregation dimension. The same holds for the discrimination process in the discrimination/generalization dimension. The modularity of both processes is further enhanced by the fact that they cannot directly invoke each other. Two different queues serve as a buffer between the processes.

Figure 3.2 shows a flow chart of the control structure of the interpreter. Interpretation takes place in cycles. During each cycle, one schema instance is matched and linked with a schema instance at the next higher level of composition. Composition is subdivided in two stages: completion and assembly. Discrimination, a constraint propagation process, is implemented by means of a hierarchical arc consistency algorithm which is invoked twice during an interpretation cycle.

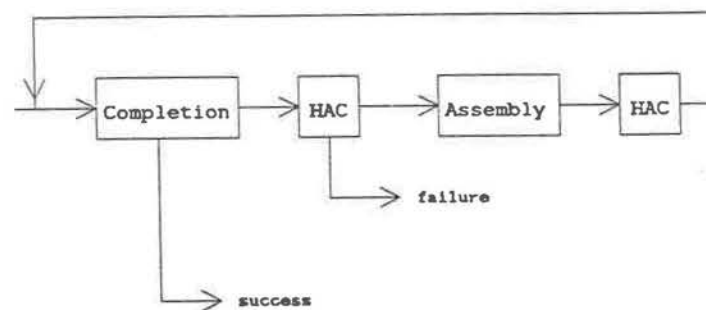


Figure 3.2: Flow chart of the interpretation process

In this chapter we look at composition and discrimination from a conceptual point of view. The actual implementation is discussed in Chapter 4. A formal description of the interpreter is provided in Appendix E.

### 3.3.1. Composition

Image-to-scene mapping results in a number of schema instantiations at the composition leaf level, one for each image primitive. The objective of composition is to represent these leaf level instances at each of the other levels of composition, thereby establishing different relationships between the instances. Composition is a data-driven process. During each cycle of interpretation one schema instance is represented at the next higher level of composition.

Composition is subdivided in two stages: completion and assembly. The reason for this subdivision is that for one instance to be a component of a super-instance both instances must match in two different ways. First of all, the lower instance must establish a relationship with another instance at the same level of composition, which has already been established as a component of the super-instance. Secondly, the labels of the lower instance must be compatible with those of the super-instance. The first match is achieved by the completion process, the second match by the discrimination process. Once both types of match have succeeded, the lower and super-instance can be linked in an operation taken care of by the assembly process.

In order to obtain a match between an instance and a super-instance the completion process invokes the latter's *methods*. These methods are functions which are stored as the value of the "method" attribute of the super-instance's parent. Once invoked, these functions search for a relationship between the completing component and an instance which is already established as a component of the super-instance. These functions are specific for the domain in which they operate and together form a domain-dependent aspect of the representation which the user must provide when creating the knowledge base.

The list of schema attributes described in section 3.2.1 show that each schema instance has a parent and that each schema is embedded in a composition hierarchy. This hierarchy can be followed by tracing the schema's component attributes.

The first step in the completion of a schema instance  $S_i$  is to fetch the super-component(s) of its parent  $S$ . Let us assume for the moment that there is only one super-component, which we call  $SS$ . The objective of the completion process is to find an instance  $SS_i$  of  $SS$  of which  $S_i$  can become a component.  $SS_i$  is a valid super-component for  $S_i$  if an existing component  $C_i$  of  $SS_i$  can be found that can enter in a relationship with  $S_i$ . Schema  $SS$  has methods with the power to search for and establish such relations. The completion process can invoke these methods when it attempts to complete  $S_i$  to  $SS_i$ .

In the case that  $SS$  does not yet have a single instance then the completion process has the power to create a new instance  $SS_i$  which becomes a completion

candidate for  $S_i$ . The same is done when  $S_i$  fails to match with any of the existing instances of  $SS$ .

An example will be helpful to further clarify composition. Suppose, we want to interpret a simple sketch consisting of two towns connected by a road (Figure 3.3). We will be using the composition hierarchy in Figure 3.4. In this example we bypass the fact that there exists a relational level in between the two object class levels. The segmentation process has constructed three primitives: two blobs and a line. We assume that the blobs serve as a cue for town and line for road. As a result, two town instances (*town-1* and *town-2*) are created and one road (*road-1*). The details of the composition process will be somewhat dependent on the sequence in which we complete these instances; the final result, however, should always be the same.

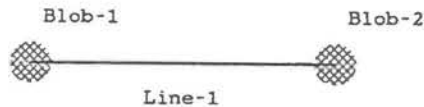


Figure 3.3: Two towns connected by a road

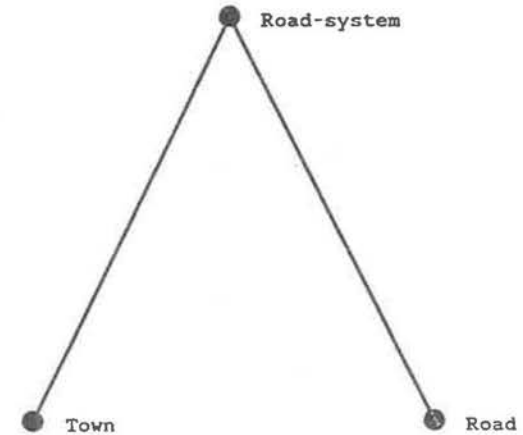


Figure 3.4: A composition hierarchy for Figure 3.3

If we complete *town-1* first, then *town-1* becomes a component of road-system. No road-system thus far exists, so we create one: *road-system-1* of which *town-1* becomes a component. The completion of *road-1* results in the invocation of road-system's methods. Road-system owns several methods, in particular one that establishes whether one of *road-1*'s ends is close to *town-1*'s location. If this is the case, then a spatial relation "road-townp" is established between *road-1* and *town-1* and *road-1* becomes a component of *road-system-1*.

If, however, we complete *town-2* before *road-1*, then the composition process follows a different course. If we complete *town-2* to road-system then

the road-system will find that it has no method to establish a relationship between *town-1* and *town-2*. As a result, it creates a new instance *road-system-2* of which *town-2* becomes a component. If we now complete *road-1* then the road-system finds that *road-1* can be a component of both its instances. Since *road-1* can be a component of one road-system instance only, road-system will merge *road-system-2* and *road-system-1* into one new instance *road-system-1* of which *road-1* becomes a component.

Composition becomes more complex when discrimination graphs get involved. Figure 3.5 illustrates such a situation. We will use this composition/discrimination graph to reinterpret Figure 3.3 with the difference that we now take line-1 to be an ambiguous feature serving as a cue for a road and a river. Using the principle of least commitment, line-1 causes an instantiation of road/river: *road/river-1*. If we complete *road/river-1* it will complete to road/river-system. With not a single instance created at the second composition level, *road/river-1* becomes a component of *road/river-system-1*. *Town-1*, however, wants to complete to road-system. Now it is not sufficient to say that because road-system has no instances, we must create a new one. *Town-1* can become a component of road-system, as well as of any generalization or discrimination (if there are any) of road-system.

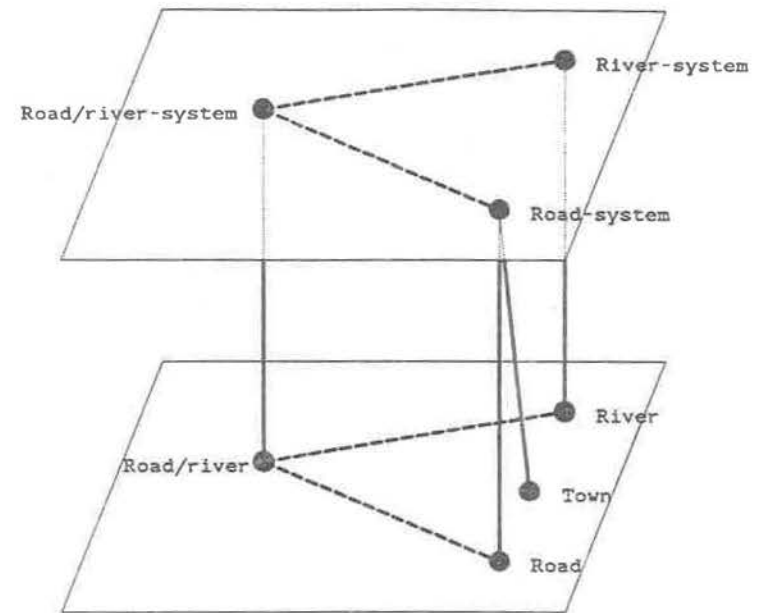


Figure 3.5: A composition hierarchy and discrimination graphs at two levels of representation

Abstract schemata such as *road/river-system* can inherit the methods of their descendents in the discrimination graph under certain circumstances. We will discuss these inheritance rules in section 3.4.4. It is sufficient for the moment to assume that *road/river-system* inherits *road-system*'s "road-townp" method. Completion of *town-1* to *road/river-system* enables the latter to use



road-system's "road-townp" method to establish the T-junction. As a result *town-1* becomes a component of *road/river-system-1*.

From this example we can infer that an instance does not necessarily complete to an instance of its super-schema. Given an instance  $S_i$  of a schema  $S$ , what is the set of schemata to which  $S_i$  can potentially complete? Let  $SS$  be the super-schema of  $S$  in the composition hierarchy. Let the *discrimination set* of  $SS$  be all the possible discriminations and generalizations of  $S$  in the discrimination graph in which  $S$  is embedded. We include  $SS$  in this set and call the set  $D$ . Let the *superdiscrimination set* of  $SS$  be the set of all schemata which have an element of  $D$  as a possible discrimination. We call this set  $SD$ . The completion set for  $S_i$  are the instances in  $D \cup SD$ . If no matching instance in this set can be found then  $S_i$  becomes a component of a newly created instance of  $SS$ .

As an example consider the discrimination graph in Figure 3.6.  $SS$ 's discrimination set is given by the dotted line, the superdiscrimination set by the dashed line. The arrows show the direction of discrimination in this graph.

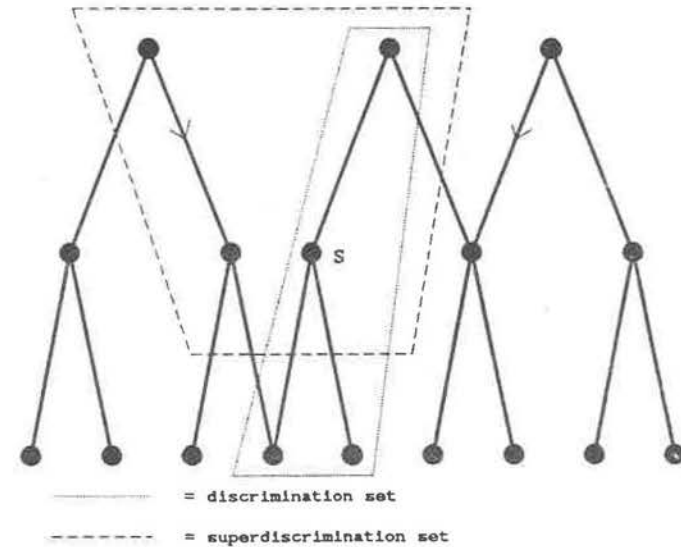


Figure 3.6: A discrimination and superdiscrimination set

We can also infer from the example in Figure 3.5 that completion proceeds in a non-hypothetical manner. Thanks to the existence of abstract schemata such as *road/river* and *road/river-system* there is no need to create two hypothetical instances *road-1* and *river-1* and to pursue them as such. Thus, no hypotheses can enter in the composition/aggregation dimension.

Completion does not actually link  $S_i$  and  $SS_i$ , but this is done during assembly which is postponed until a compatibility check has been made

between the label(s) of  $S_i$  and  $SS_i$ . Such a compatibility check involves operations in the discrimination/generalization dimension and belongs therefore to the domain of the discrimination process to which we turn next.

### 3.3.2. Discrimination

In section 3.2.2, we looked at discrimination graphs as a representation for "look alike" schemata that could be discriminated only when placed in a proper context. Upon instantiation a new instance inherits the schema label of its parent as label. Thus, in Figure 3.5 *road/river-1* will start out with *road/river* as label and *town-1* with *town* as label. The discovery of a relation "road-townp" causes both instances to become a component of the same super-component: *road/river-system-1*. In the model constraint graph represented by the composition hierarchy in Figure 3.5, *Part-of (town, road/river-system)* is false. Compatibility of labels is achieved by replacing the labels of *road/river-system-1* and *road/river-1* by one of their successors in the discrimination graph: *road-system* and *road*.

We will formally describe and treat discrimination as a constraint satisfaction problem. In the following section we will therefore describe a general algorithm that deals with constraints organized in hierarchical form, as they are in the discrimination graph.

#### 3.3.2.1. Hierarchical Arc Consistency

In section 2.3.2.3 the Constraint Satisfaction Problem (CSP) was defined as a situation with  $n$  variables each with a domain and a set of constraining relations. A solution to this problem consists of all possible  $n$ -tuples such that each  $n$ -tuple is an instantiation of the  $n$  variables satisfying the relations. Several kinds of algorithms were discussed. Some algorithms, such as depth-first backtrack always solve the problem, whereas others such as network consistency algorithms, do not provide that guarantee.

The problem with algorithms such as depth-first backtrack is that they are exponential in the domain size which is particularly damaging in computational vision problems where domain sizes are generally very large to begin with. Network consistency algorithms such as arc consistency are considered to be useful because they are generally polynomial in the domain size.

Additionally, arc consistency is attractive, because:

1. In the best possible case, arc consistency solves the CSP. If there is only one  $n$ -tuple satisfying the relations for all  $n$  variables and if the  $k$ -ary ( $k \leq n$ ) constraints are sufficient to propagate this solution, the CSP is solved.
2. In the worst possible case, arc consistency does not solve the CSP, but can serve as a useful preprocessor for depth-first backtrack.
3. Arc consistency algorithms are simpler than depth-first backtrack. They have

been well researched, and their complexity is known (Mackworth and Freuder, 1982).

4. The discrimination process can be implemented entirely by means of an arc consistency algorithm. This process is the only one to deal with the discrimination/generalization dimension. Hence, there is modularity in both representation and control.

The first two arguments are efficiency arguments. The last two arguments, however, strongly support the conceptual clarity of the system. For this reason, Mapsee-3 uses an arc consistency algorithm to do constraint propagation. However, in regular arc consistency algorithms the domain consists of label sets. This would not suffice for the discrimination graphs in which labels are organized in a hierarchical form. Mapsee-3 therefore uses hierarchical arc consistency, a hierarchical version of arc consistency.

Hierarchical arc consistency is a network consistency algorithm for hierarchically organized domains. The objective of a hierarchical representation is that for each variable instantiation we no longer have to explicitly represent all possible labels. This set is now represented implicitly by one or more abstract labels. The particular algorithm we will be discussing in this section is called HAC-3, a derivative of AC-3 described in Mackworth (1977c). Hierarchical arc consistency (HAC) is formally described in a concurrent paper (Mackworth, Mulder, and Havens, 1985). As HAC-3 is used in Mapsee-3, we will discuss this topic in some

detail here as well. In this section AC-3 and HAC-3 will be described informally. A formal description is provided in Appendix A.

### AC-3

We represent the CSP as a graph  $G$  with variables  $V$ , each with a domain  $D$ .  $G$  is consistent if all nodes and arcs are consistent. A node  $V_i$  is consistent if the predicates applicable to  $V_i$  are true for all labels in  $D_i$ . The arcs are consistent if for each arc  $(i, j)$  each of the labels in  $D_i$  is consistent with at least one label in  $D_j$ .

AC-3 consists of two steps. In the first step, the consistency of each node and arc is tested. All inconsistent labels are deleted. In the second step all arcs pointing at a variable in the domain of which labels were deleted are revisited. Step 1 is repeated for each of those arcs. This process continues until all nodes and arcs are consistent or until the domain of each variable is empty.

AC-3 is a very interesting algorithm for applications in computational vision. It does not build an explicit data structure for administering the compatibility between different labels of adjacent variables in the constraint graph, but instead, AC-3 searches for a compatible label in the domain of an adjacent variable, terminating the search as soon as such a label is found. No data structure is created for administering the compatibility. The worst-case complexity of AC-3 is linear in the number of constraints and quadratic in the domain size (Mackworth and Freuder, 1982).

In the best possible case for AC-3, all but one label of each variable will eventually be deleted. In the general case, AC-3 is not guaranteed to solve the constraint satisfaction problem. However, as discussed in section 2.3.2.3 AC-3 is a useful preprocessor by means of which we can reduce the domain size before applying depth-first backtrack.

### HAC-3

In AC-3 the domain of a variable is organized as a set of labels. Each instantiation of a variable has this set or a subset of these labels in its domain. In HAC-3 the domain  $D$  is organized in a hierarchical form. Each node in this hierarchy stands for a label that is unique in  $D_i$ . The labels at the leaves of the hierarchy are the same (basic) labels that were represented in the variable domain in AC-3. The source node of the hierarchy intensionally represents the complete set of labels at the leaves of the hierarchy. Each intermediate node represents a subset of this set.

HAC-3 is also a two-step algorithm. In the first step the hierarchical arc consistency of each node and arc is tested. An arc  $(i, j)$  is hierarchically arc consistent, if each label in  $D_i$  is hierarchically arc consistent with at least one label in  $D_j$ . A label pair  $(m, n)$  is hierarchically arc consistent if for all descendants  $d$  of  $m$  in the hierarchy  $P(d, n)$  is true. For each label pair  $(m, n)$  that is not hierarchically arc consistent we replace  $m$  by those descendants  $d$  in the hierarchy such that  $(d, n)$  is hierarchically arc consistent. The second step in HAC-3 is

similar to the one in AC-3.

In both HAC-3 and AC-3 the final label of a variable is determined entirely by the constraints present in the image. In HAC-3, however, the final label need not be a leaf label in the discrimination graph but can be any intermediate label as well. This appears to be a natural phenomenon. An example illustrates this. Under ideal observation conditions a human observer can specialize a picture of a car up to its make and year. Such a recognition, however, would not be possible for the same car covered with snow.

This is a major advantage of HAC-3 over AC-3 and depth-first backtrack where a unique final label is not possible unless it is a leaf label. In natural image understanding one cannot always discriminate down to the leaves of the graph as the snow covered car demonstrates. HAC-3 mimics this behavior. Another advantage of HAC-3 over AC-3 is one of efficiency. Compared to AC-3 the domain size of HAC-3 is generally smaller. As a result, there can be an increase in time efficiency. In Chapter 5 we will discuss some experimental data obtained from Mapsee-3 which actually show an improvement in time efficiency of HAC-3 over AC-3.

### 3.4. Setting up a Knowledge Base for a Particular Domain

We have now discussed the Mapsee-3 representation and control. However, we have not yet shown how to actually construct a knowledge base for a particular domain. One part of the knowledge base has to be provided by the user, whereas the other part is automatically constructed by the system.

The very strict modularity requirements of the Mapsee-3 system impose a number of restrictions on the ways in which we can construct the different knowledge representation dimensions. Most important of all, the discrimination/generalization dimension must represent the knowledge about possible ambiguities in interpretations. In order to preserve modularity in representation and control we must prevent hypothetical interpretations from being introduced along any of the other dimensions.

In the image-to-scene dimension we have to ensure that for every image primitive there exists an (abstract) schema in the scene which intensionally represents all possible interpretations of this primitive. If we map the primitive into this schema, we obtain an interpretation that is ambiguous but is not hypothetical.

In the composition/aggregation dimension we have to ensure that the mapping of an interpretation at one particular level of composition never leads to an instantiation of a schema at an adjacent level that is hypothetical. This can be prevented if we ensure that there are no composition relations of the "may-be"

kind. In the sketch map world, for instance, a road is a mandatory component of a road-system. A primitive interpreted as a road must therefore be part of a road-system as well. If the road interpretation is non-hypothetical then the road-system interpretation cannot be hypothetical either. Because the composition process in Mapsee-3 is entirely data-driven, all "part-of" relations in the Mapsee-3 implementation are of the "must-be" kind.

Given all these constraints on the structure of the knowledge base, its construction is not a trivial matter. In particular, the presence of schemata which do not represent natural concepts complicate matters. Both natural and unnatural concepts have to be embedded in a composition hierarchy. In order to relieve the user from the burden of having to construct the complete knowledge base, we have designed a number of algorithms that enable us to automatically construct the unnatural constituent of the knowledge base.

In the next three sections, we discuss the construction of the Mapsee-3 knowledge base. In the first section we discuss the information that has to be provided by the user and the constraints which must be satisfied. In the second section we discuss the construction of the discrimination graphs located at the leaf level of the Mapsee-3 composition hierarchy. In the third section we discuss the algorithm by means of which the remainder of the knowledge base is constructed.

### 3.4.1. The Basic Composition Hierarchy and Discrimination Graphs

Mapsee-3 requires the presence of a segmentation process that cumulates in the formation of a number of (image) primitives. For any particular domain the user must categorize the primitives with respect to one or more characteristics that are of interest (e.g. shape, texture). Each primitive category must map into a particular set of interpretations. This mapping scheme must also be provided by the user.

Each interpretation is represented as a schema. The user must provide the information for all natural schemata as follows:

1. An internal structure for each schema. This structure consists of a list of attribute-value pairs which conforms in format with the syntactic rules for schemata provided in Appendix D. The user is also responsible for providing the schema's procedural knowledge, its methods.
2. The composition hierarchy and discrimination graph(s) in which these schemata are embedded. In an aggregation direction only "must-be-part-of" links are allowed. Discrimination graphs must be orthogonal to the composition hierarchy.
3. The set of image primitive categories depicted by each schema and *vice versa*.

The reader in need of a concrete idea of what the Mapsee-3 composition hierarchy and discrimination graphs look like at this stage is invited to look

ahead to Figure 4.4. As mentioned before, the image primitives in Mapsee-3 map into schemata at the composition leaf level only. This is not a general constraint on the knowledge base but it simplifies its construction.

From here on we will refer to a natural schema as a *basic* schema. The composition hierarchy containing basic schemata is referred to as the *basic* composition hierarchy. All discrimination graphs embedding basic schemata are *basic* discrimination graphs. Unnatural schemata will be referred to as *abstract* schemata.

### 3.4.2. Constructing an Abstract Discrimination Graph at the Composition Leaf Level

While describing the concept of discrimination graphs, we pointed out that they can represent any approach, be it a hypothetical approach, an ambiguity approach, or any position in between. The Mapsee-3 system takes an intermediate position that comes very close to an ambiguity point of view, closer than any of the vision systems reviewed in Chapter 2. All the different sets of basic interpretations that can arise for any image primitive category are represented by a single schema. At the start of the interpretation process, each primitive can therefore be represented by a single (abstract) schema. However, not all combinations of basic interpretations are explicitly represented. In intermediate situations in the interpretation process, a primitive sometimes depicts more than one

interpretation.

The Mapsee-3 discrimination graphs are OR graphs which are binary in a discrimination direction. Neither condition is a general constraint on the approach. The only reason for imposing these constraints is to obtain an improvement in the time efficiency of the network consistency algorithm operating on these graphs. The leaves of the graph represent basic schemata, whereas most of the other nodes in the graph represent abstract schemata which intensionally represent the basic schemata that descend from them. The network consistency algorithm operating on the graph follows a principle of least commitment. For this reason the set of basic schemata intensionally represented by each node must be a subset of the set represented by its ancestor(s).

The first discrimination graphs to be constructed are the discrimination graphs at the composition leaf level. These graphs come first, because the composition leaf level is the point of entry in the scene domain for the image-to-scene mapping process. At this point, only the basic discrimination graphs are given as they were constructed by the user. In Figure 4.4, level 1 is the composition leaf level and contains only one basic discrimination graph that consists of more than one node. The abstract discrimination graphs will become an extension of these graphs.

No abstract schemata have yet been created. As a first step, we take each primitive category and create an abstract schema which intensionally represents the set of basic schemata depicted by the category. In the example in Figure 3.7,

for instance, we have assumed the existence of four different primitive categories: one depicts a town, a second one depicts a road or shore, a third one a road, river, or shore, and a fourth one a road or river. As the town is a basic schema which already exists, we only have to create three abstract schemata: a road/shore, a road/river/shore, and a road/river.

Next, we create abstract discrimination graphs which are binary OR graphs. These graphs are constructed such that the source nodes of the basic discrimination graphs become the leaves and all abstract schemata are contained in the graphs. As well, the graphs are constructed such that the descendants of each abstract schema represent an exclusive subset of the set of basic schemata represented by their ancestors. The latter has also been motivated by an efficient operation of the network consistency algorithm operating on the graph.

The algorithm that constructs the abstract discrimination graph at the composition leaf level consists of three steps:

Step 1. Order the abstract schemata by the size of the set of basic schemata they intensionally represent. Call this ordered list  $AS$ .

For each element  $s$  in  $AS$  do:

Find the schema that represents the largest subset of  $s$ . Call this schema  $s_1$ . If the setsize of  $s_1$  is less than half the size of  $s$   
 then execute step 2,  
 else execute step 3.

Step 2. Split the set represented by  $s$  into two disjoint subsets of approximately equal size, and create two new abstract schemata  $s_3$  and  $s_4$ . Each of these schemata represents one of the subsets. Create a discrimination link between  $s$  and its two siblings and insert the siblings into  $AS$  at a location that corresponds to their setsize.

Step 3. Find the schema  $s_2$  representing the exclusion of the sets represented by  $s$  and  $s_1$ . If this schema does not exist

then create it and insert it into  $AS$  at the proper location.

Create a discrimination link between  $s$  and its siblings  $s_1$  and  $s_2$ .

The graph that results from this construction is not necessarily unique. One may find more than one  $s_1$ . In this case one can arbitrarily take one of these schemata to be  $s_1$ . Depending on the  $s_1$  chosen the result will be different.

Figure 3.7 is a simple example of the construction of discrimination graphs. The basic schemata in this figure are: *town*, *road*, *river*, and *shore*. The abstract schemata are (ordered by setsize): *road/river/shore*, *road/shore*, and *road/river*. Two discrimination graphs result from this construction. One consists of a single schema: *town*. The other contains all other schemata. Note the ambiguity in the construction. *Road/river/shore* could also have been linked with *road/river*

and *shore* instead.

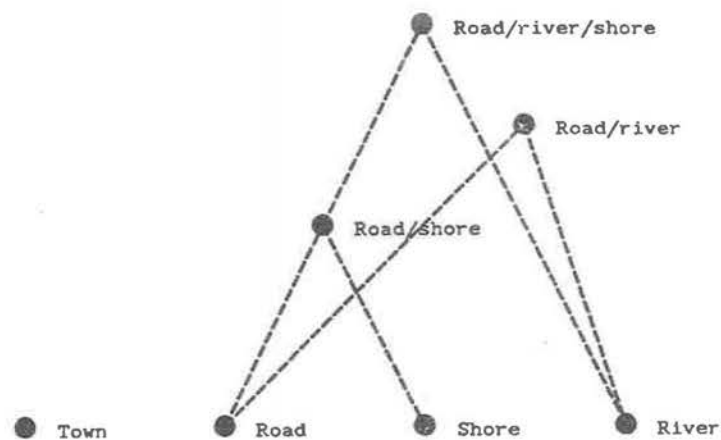


Figure 3.7: A construction example of a discrimination graph

### 3.4.3. Embedding Abstract Schemata in a Composition Hierarchy and Constructing Abstract Discrimination Graphs at Multi-levels of Composition

Thus far we have created abstract discrimination graphs at the composition leaf level only. For modularity reasons alone, it is necessary to embed each of the abstract schemata in these graphs in a composition hierarchy as well. If this were not the case, then we would face the problem that interpretations could never be developed along the composition/aggregation dimension without becoming hypothetical. Figure 3.8 can serve as an illustration of this problem.



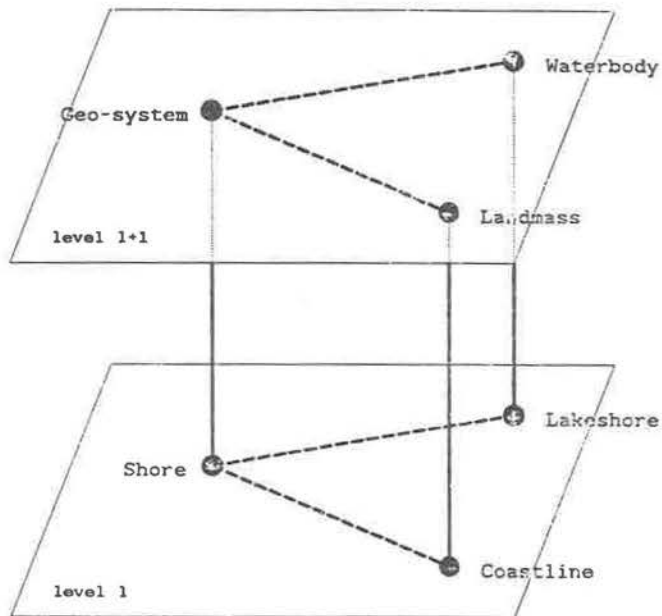


Figure 3.8: A one-to-one mapping situation

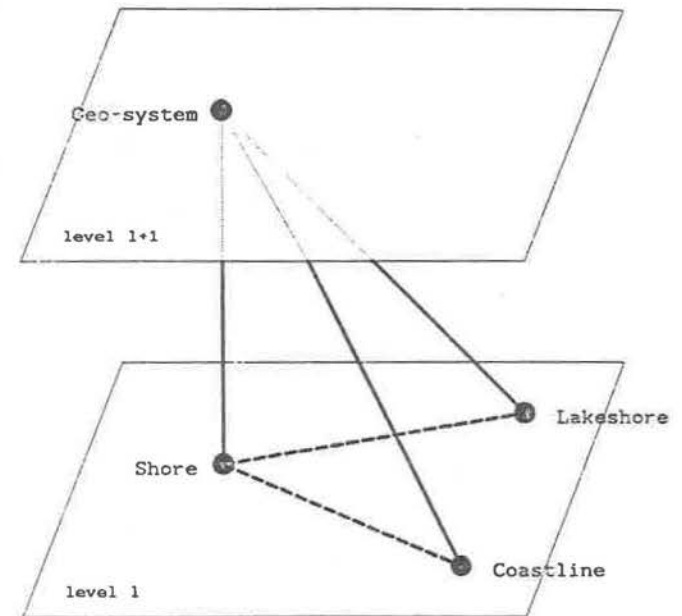


Figure 3.9: A many-to-one mapping situation

Suppose that *coastline* and *lakeshore* are basic schemata embedded in a basic composition hierarchy. *Coastline* is a mandatory component of *landmass* and likewise *lakeshore* is a component of *waterbody*. Also suppose that *shore* is an abstract schema which intensionally represents *coastline* and *lakeshore* and we have just constructed the abstract discrimination graph containing *shore*, *coastline*, and *lakeshore*. *Shore* is not yet in a composition hierarchy. Without a

super-schema there is only one way to represent *shore* at the next higher level. We replace *shore* by its two (hypothetical) descendants both of which have a super-schema. However, this means that *shore* becomes a component of two hypothetical super-schemata: *landmass* and *waterbody*, only one of which can be the correct one. In other words, without a super-schema for *shore* we can only represent *shore* at a higher level by introducing a hypothetical schema. The only way to avoid this problem is to find or create a super-schema at level  $l+1$  of which *shore* is a mandatory component.

In the composition/aggregation dimension basic schemata map from one level of composition to the next one up in one of three ways: one-to-one, many-to-one, or one-to-many. Figure 3.8 illustrates a one-to-one mapping situation. Each of the shore's descendants maps into a different super-schema. *Shore* cannot map into either *landmass* or *waterbody* without introducing a hypothetical schema. We must therefore create a new (abstract) schema at level  $l+1$  (*geo-system*) which intensionally represents *landmass* and *waterbody*. *Shore* becomes a mandatory component of *geo-system*. Using the algorithm for creating abstract discrimination graphs we create a new abstract discrimination graph at level  $l+1$  which contains *geo-system*, *landmass*, and *waterbody*. We thus effectively project the discrimination graph at level  $l$  onto level  $l+1$ .

Figure 3.9 illustrates a many-to-one mapping situation. Both of *shore*'s descendants map into one super-schema, *geo-system*. In this situation there is no need to create a new abstract super-schema for *shore*. All of *shore*'s hypotheti-

cal interpretations are a component of *geo-system*. *Shore* must therefore become a mandatory component of *geo-system* as well.

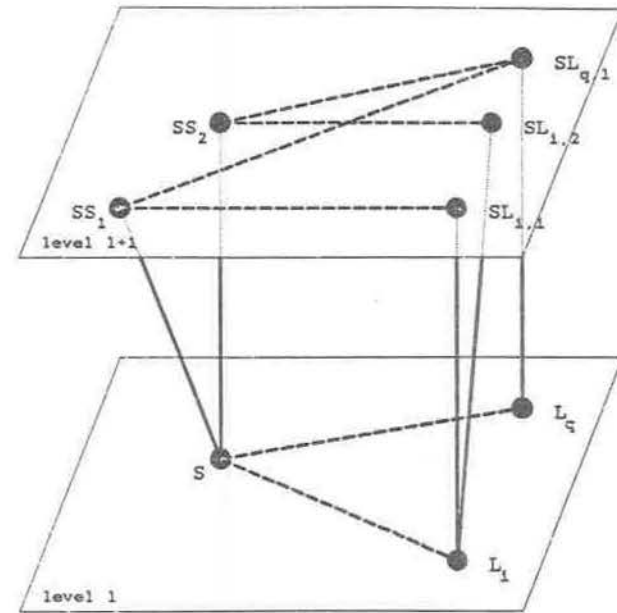


Figure 3.10: A one-to-many mapping situation

Figure 3.10 illustrates the most complicated mapping, a one-to-many mapping. One of *S*'s descendants,  $L_i$ , is a mandatory component of two super-schemata:  $SL_{i,1}$  and  $SL_{i,2}$ . *S* cannot become a component of any of the  $SL$  schemata without introducing hypothetical schemata. However, the creation of just one super-schema for *S* at level  $l+1$  is an inadequate solution as well.

Once again, the reason is modularity.

As mentioned before, an interpretation is represented in Mapsee-3 as a schema-instance. Each instance is embedded in an interpretation graph which reflects the structure of the composition hierarchy in which the instance's parent schema is embedded. For example, an instance of schema  $L_q$  in Figure 3.10 is embedded in an interpretation graph which, among other things, contains an instance of schema  $SL_{q,1}$ . An instance of schema  $L_i$ , on the other hand, will find itself in an interpretation graph with an instance of schema  $SL_{i,1}$  and an instance of schema  $SL_{i,2}$ . As the interpretation graph reflects the composition hierarchy, the composition process is responsible for constructing and altering the structure of this graph.

The discrimination process, on the other hand, can only operate in the discrimination/generalization dimension and it cannot change the structure of the interpretation graph. In order to preserve modularity in control, we have to prevent that an operation in the discrimination/generalization dimension necessitates a change in the structure of the interpretation graph. This would happen if  $S$  had only one super-schema  $SS$ . If during interpretation a schema-instance with label  $S$  refines its label to  $L_i$ , then  $S$ 's super-component  $SS$  would be required to split into two instances, one with label  $SL_{i,1}$ , and another with label  $SL_{i,2}$ . This split would require a change in the structure of the interpretation graph.  $S$  must therefore have at least as many super-schemata as any of its descendants in the discrimination graph.

The same modularity requirement is reflected in the construction of the discrimination graphs at level  $l+1$ . The only way to avoid changes in the structure of the interpretation graph is to ensure that all super-schemata of descendants from  $S$  generalize to at least one super-schema of  $S$ . In the reverse direction each super-schema of  $S$  must map into one super-schema of each of  $S$ 's descendants. In this way it is guaranteed that a label refinement in an interpretation graph which contains instances of  $S$ ,  $SS_1$ , and  $SS_2$  always enables us to find new labels for each of these instances without having to change the structure of the graph.

We will now discuss the algorithm by which the projection of an abstract schema onto level  $l+1$  and the subsequent construction of a discrimination graph at that level can be done. The reader can verify that for the basic mapping situations illustrated in the Figures 3.8 and 3.10 the application of this algorithm results in the situation illustrated in these figures. The situation in Figure 3.9 is dealt with correctly as well. However, a redundancy is created because *shore* becomes a component of a newly created (abstract) generalization of *geo-system* rather than *geo-system* itself.

## The Projection Algorithm

*- Subdividing the discrimination graphs into subtrees*

The first step in projecting a discrimination graph from level  $l$  onto level  $l+1$  is a subdivision of each discrimination graph into two-level subtrees. Each node in the graph and its direct descendants in a discrimination direction form a subtree. The next step is to assign a level number to each node in the discrimination graph. The leaves of the graph become level 1, their parents level 2 etc. In case of conflict the highest level number prevails. The projection takes place subtree by subtree, beginning with all trees with source node at level 2, because all leaves at level 1 are already contained in a basic composition hierarchy. Next the trees at level 3 etc.

*- Projecting subtrees from composition level  $l$  to level  $l+1$* 

Each subtree  $ST$  consists of a source schema  $S$  and a set of leaves  $L$ .  $L$  has  $q$  elements:  $L_1, \dots, L_q$ . A discrimination link  $d_{i,l}$  connects  $S$  with  $L_i$  ( $1 \leq i \leq q$ ).  $n_i$  "must-be-part-of" links emanate from each  $L_i$  ( $1 \leq i \leq q$ ). The set of super-schemata in the composition hierarchy of  $L_i$  we call  $SL_i$ .  $SL_i$  has  $r$  elements:  $SL_{i,1}, \dots, SL_{i,r}$  ( $r = n_i$ ). Figure 3.11 shows the situation.

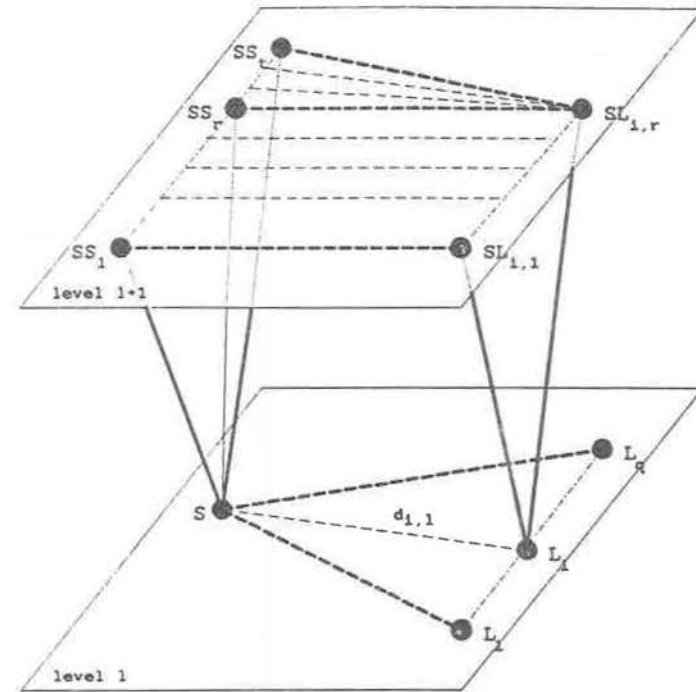


Figure 3.11: An illustration of the projection algorithm

The projection algorithm consists of two steps:

1. Create a new set of super-schemata  $SS$  and create a "must-be-part-of" link between  $S$  and each element of  $SS$ .  $SS$  has  $r$  elements:  $SS_1, \dots, SS_r$  ( $r = \max n_i \{1 \leq i \leq q\}$ ).
2. For each set  $SL_i$  ( $1 \leq i \leq q$ ) do:

Create new discrimination links connecting  $SL_i$  with  $SS$ . Each of these links represents  $d_{i,l}$  at level  $l+1$ . The elements in both sets are connected in the following way:

$SL_{i,1}$  connects with  $SS_1$ ,  $SL_{i,2}$  connects with  $SS_2, \dots, SL_{i,r}$  connects with  $SS_r$ .

If  $r < l$

then create additional links that connect  $SL_{i,r}$  with  $SS_k$  for each possible value of  $k$  ( $r < k \leq l$ ).

Figure 3.11 shows there is a one-to-one mapping between  $SS$  and  $SL_i$  for the elements 1 -  $(r-1)$ , and a one-to-many mapping from  $SL_{i,r}$  to the elements  $SS_r - SS_l$ . This algorithm guarantees that there is a unique mapping between  $SS$  and  $SL_i$  ( $1 \leq i \leq g$ ) for each  $d_{i,l}$ .

By means of this algorithm one automatically constructs abstract composition hierarchies and abstract discrimination graphs at multiple levels of composition. The only complication with respect to Mapsee-3 is that the "must-be-part-of" links themselves are represented as schemata. In Mapsee-3, object class levels alternate with composition relation levels. The projection algorithm works well in that situation, but does not provide economy of representation. Figure 3.9 is an example of such a situation. In the projection algorithm that was actually used to construct the Mapsee-3 knowledge base, some efficiency measures were taken. These measures were implemented as preprocessors of the general projection algo-

rithm and they are not essential to the general understanding of the projection algorithm. The reader is therefore referred to Appendix C for a description of these measures.

#### 3.4.4. Method Inheritance

The discussion of how to automatically construct abstract schemata at different levels of composition has now been completed. Most of the attributes of abstract schemata are easy to generate as they depend directly on the location of the schema in the composition hierarchy or discrimination graphs. However, this is not the case for the schema's methods.

Many schemata own methods. As discussed before, they are used to search for and establish relationships between components of the schema which owns the method. Methods are central to the interpretation process because the relationships they create form the constraints on the operation of both the composition and discrimination process. In Mapsee-3 only basic schemata own methods because they have to be provided by the user. Methods apply to one or more schemata at one particular level of composition only.

Because of their central importance to the composition process, abstract schemata need methods as well. Fortunately, it is possible to provide procedures by means of which we can automatically determine whether or not a method of a particular schema can be inherited by any of its generalizations in the

discrimination graph. The goal of methods is to create relationships between instances at a lower composition level. Two or more instances can enter in a relationship if the primitives depicted by the relationship satisfy particular constraints, and if the labels of the respective instances are not neutral with respect to each other. An example will illustrate how to determine inheritance.

In Figure 3.12 we are faced with the question whether or not a T-junction can exist between the line segments shown. The spatial configuration of the line segments allows for the formation of such a junction. Most spatial relationships in the Mapsee-3 implementation consist of T-junctions of one form or another. The stem of the T-junction has three possible interpretations: *road*, *river*, or *mountain*. For the bar, there exist only two possible interpretations: *road*, or *river*. For any pair of interpretations, the Mapsee model constraint graph determines whether particular pairs of interpretations can coexist under the spatial situation shown and whether a T-junction can be formed. Coexistence and T-junction formation rules can be expressed by means of a matrix, an example of which is shown in Table 3.1. The possible interpretations for the stem of the Tee are the rows of the matrix, the bar interpretations form the columns. Each cell in the matrix can assume one of three values:

+ if the interpretation in the corresponding row can form a T-junction with the interpretation in the corresponding column.

- if the interpretation in the corresponding row cannot coexist with the interpretation in the corresponding column under the spatial situation shown.

\* if the interpretation in the corresponding row cannot form a T-junction, but can coexist with the interpretation in the corresponding column.

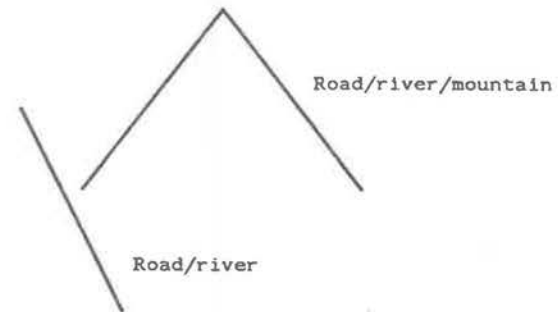


Figure 3.12: An example of a T-junction in a sketch map

	Road	River	Mountain	Shore	Bridge-side	Town
Road	+	-	-	*	-	+
River	-	+	+	+	+	-
Mountain	*	*	+	*	*	*
Shore	-	-	-	-	-	-
Bridge-side	*	*	*	*	*	*
Town	*	*	*	*	*	*

Table 3.1: A T-junction matrix

	Road	River
Road	+	-
River	-	+
Mountain	*	*

Table 3.2: A T-junction matrix for Figure 3.12

	Road	River
Road	+	-
River	-	+

Table 3.3: The T-junction matrix for two road/river

A T-junction can only be formed in Figure 3.12 if each of the possible interpretations for the stem can form a T-junction with at least one of the possible interpretations for the bar and is not indifferent with respect to any of these interpretations (i.e. no \*). Table 3.2 shows the situation in Figure 3.12. No T-junction can thus be formed because *mountain* is indifferent to both *road* and *river*.

This example shows the essence of determining whether a particular method can be inherited by an abstract schema. Sets of basic schemata form an abstract schema. The possible interpretations for the stem of the Tee define the abstract schema *road/river/mountain*; the bar interpretations define the abstract schema *road/river*. A *road/river/mountain* and a *road/river* cannot form a T-junction. Table 3.3, on the other hand, shows that two *road/river*s can. T-junction

methods are owned by schemata such as *road-system* and *river-system*. We can set up a simple procedure for determining whether this method can be inherited by any of *road-system*'s or *river-system*'s generalizations in the discrimination graph.

Each method  $M$  can create a relation  $R$  at composition level  $l$ . A method  $M$  owned by a schema  $S$  can potentially be inherited by any of the generalizations  $G$  of  $S$  in the discrimination graph containing  $S$  and  $G$ . With the matrix representation it is easy to determine whether  $G$  inherits  $M$  from  $S$ .

1. Take the set of components of  $G$  at level  $l$ . Call this set  $GC$ .
2. Create the set  $BD$  consisting of the union of all basic descendants of  $GC$  in their respective discrimination graphs.
3. Create a matrix  $MTR$ . Each row and column entry is formed by an element of  $BD$ .
4. Fill in the values of  $MTR$  (+, -, or \*) such that each row entry represents the source of  $R$  and each column entry the object.
5. If  $MTR$  contains no \* then  $M$  can be inherited by  $G$ , otherwise it cannot.

In this example we have used a binary relation. The same procedure can be used for  $n$ -ary relations. The matrix becomes  $n$ -dimensional in such a case.

### 3.5. Discrimination Vision

We have now discussed the main features of the Mapsee-3 design. The schema syntax provides a general format for representing knowledge about a particular domain. Composition hierarchies and discrimination graphs are useful for many different domains. Once the user has provided these hierarchies for a particular domain, the processes of composition and discrimination can operate on them. The only domain-dependent aspect of the representation are the schema's methods, which are invoked by the composition process. However, the user has to provide (write) them. As well, if a particular domain requires additional domain-dependent schema attributes, it is up to the user to provide such attributes and the retrieval functions operating on them. The user, however, only needs to provide the natural constituent of the knowledge base. The projection algorithms take care of the unnatural constituent.

The objectives of the system design are conceptual clarity and efficiency. The former objective is reached because of the efforts to keep representation and control of the system modular and uniform. In particular, hypothetical and ambiguous interpretations are represented in one knowledge representation dimension: the discrimination graphs. Efficiency results partially from the representation chosen and partially from the algorithms used. In the interpretation graph, competing hypotheses are all represented in the domain of a single variable. Invalidation of a hypothesis results in only deletion or replacement of a label and not in a major restructuring of the interpretation graph. The latter would be the case in

an interpretation graph where competing hypotheses are represented by different variables. Mapsee-2 was an example of such an approach.

The hierarchical arc consistency algorithm provides efficiency as well, and does so in two different ways. First of all, the principle of least commitment helps to keep the number of labels in the variable domains as small as possible. Furthermore, hierarchical arc consistency does not maintain an explicit representation of all possible combinations of hypotheses, most of which will be eliminated during the interpretation process. If competing hypotheses are spread over different variables, then at the very least we have to partially represent the possible combinations.

The model that underlies the design described in this chapter interprets image primitives by means of schema instances which at first have an extreme generic and unspecified interpretation. As more and more constraints are discovered during composition, this interpretation becomes more and more specific until a final characterization of the scene is obtained. Because of this continuing process of interpretation refinement along a discrimination graph we call this particular approach to model-based vision: *discrimination vision*.

Summary of Mapsee-3's main features:

1. Image primitives are interpreted in terms of schemata which can be treated either as:



- a. object classes or relations with an internal structure.
- b. atomic labels.

2. Three knowledge representation dimensions are distinguished: composition/aggregation, discrimination/generalization, and image-to-scene mapping. The discrimination/generalization dimension provides a means of representing all possible forms of ambiguity as they arise from the image in an explicit declarative form in the permanent knowledge base of the system. This knowledge base consists of a natural and unnatural constituent. The latter can be constructed automatically from the former.

3. An interpretation process that consists of two modular components: composition and discrimination. Both processes are simple and modular, because ambiguous and hypothetical interpretations are represented in one dimension only. A one-dimensional representation also enables us to entirely describe discrimination by means of a network consistency algorithm.

The implementation of Mapsee-3 is described in the next chapter.

## 4. DESCRIPTION OF MAPSEE-3

### 4.1. Introduction

Like its predecessors, Mapsee-3 is a program for interpreting sketch maps. Sketch maps are a useful domain because of their simple semantics, which makes an attractive testbed for particular representational formats. A mixture of natural and conventional knowledge is necessary for interpreting sketch maps such as the one in Figure 4.1. Objects such as roads, rivers, and shores could have been taken from an aerial photograph by tracing their course. On the other hand, for the drawing of objects such as mountains and bridges, conventions exist which are only indirectly related to their natural appearance.

Mapsee-3 is an implementation of the design described in the previous chapter. The program is roughly divided in three parts: *segmentation*, *image-to-scene mapping*, and *interpretation*. Segmentation is a process that constructs the primitives and the cues for the interpretation process. All cues are based on the shape of the primitives. During image-to-scene mapping, all cues computed are systematically checked, and are then used to create generic instances of schemata at the leaf level of the composition hierarchy. One instance is created for each primitive. Interpretation is the process of composition and discrimination as described in Chapter 3.

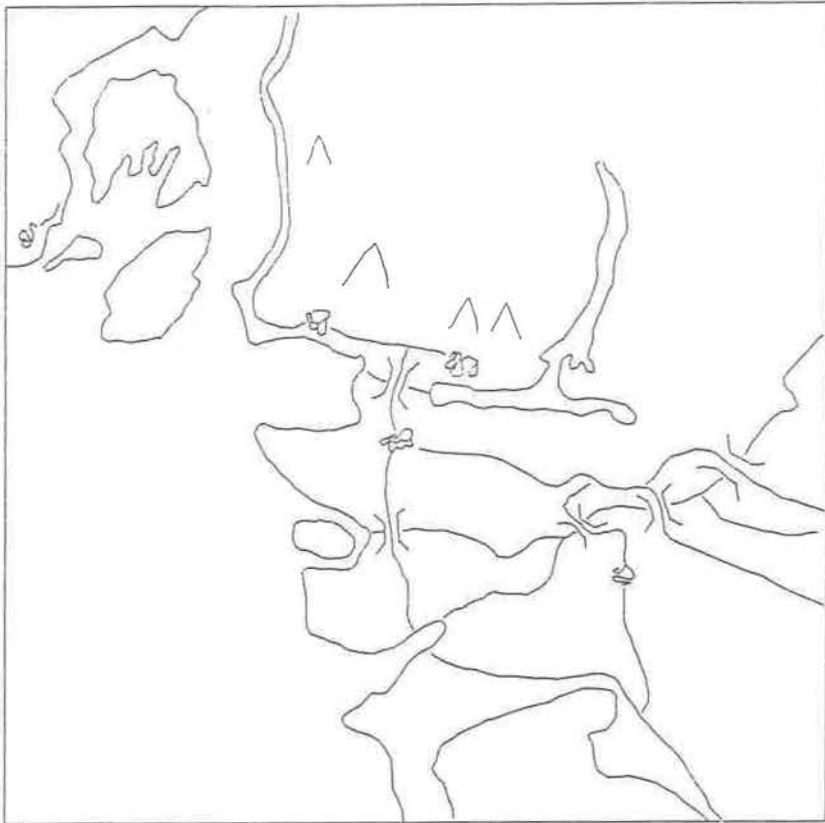


Figure 4.1: Lower Mainland of British Columbia

In sketch maps the primitives are the line segments and regions. In the scene domain, a line segment can be interpreted as one of the following object classes:

road, river, lakeshore, coastline, bridge-side, mountain, or town. Each of these is a component of one or more super-classes such as road-system and geo-system. A region, on the other hand, can be interpreted as mainland, island, lake, or ocean. Each class forms a node in a discrimination graph. For instance, the classes shore, lakeshore, and coastline are embedded in one graph.

Mapsee-3 recognizes two kinds of relationships: composition and spatial relationships. Composition relations are "part-of" and "parts" relations. The former are all of the "must-be" kind. Most spatial relationships in Mapsee-3 are depicted by T-junctions, such as "road-road-tee", a junction between two roads, and "river-shore-tee", a junction between a river and a shore.

All Mapsee-3 primitives are categorized with respect to shape. Thus, the discrimination graphs are all based on similarities in shape characteristics of the different object classes. Towns, for instance, are characterized by blobs, whereas bridge-sides are depicted by three connected straight line segments, the two outer lines of which are symmetric with respect to the middle line. Most shape categories do not uniquely determine a particular object class. The final interpretation of a particular line segment is therefore partially the result of the shape category to which it belongs, and partially the result of the spatial relationships in which it enters with other line segments.

Mapsee-3 was implemented in Franz Lisp on a Vax 11/780 running Unix 4.2BSD. Maya data structures (Havens, 1978) were used for representing schemata.

## 4.2. Input

The input to Mapsee is a plot program. Each plot consists of a sequence of plotter commands  $\{Draw-to(x,y)$  and  $Move-to(x,y)$  from the current position}. Each sequence of  $Draw-to$  commands is called a *chain*.

## 4.3. Segmentation

Six different types of schemata are maintained in the image: points, links, lines, chains, patches, and regions. Each of these types has a very simple structure.<sup>1</sup> Figure 4.2 shows an example of a line instance. In the Mapsee-3 implementation, segmentation is a semi context-free process at best. Both representation and control are tuned to the sketch map world. The types of image schemata used illustrate this phenomenon. The main purpose of segmentation is to build a reasonable shape description for each chain and to find the regions adjacent to each chain. Two different levels of representation can be distinguished in the image: The *Sketch* level and the *Line/region* level.

<sup>1</sup>The reader is referred to Appendix D for a description of the syntactic structure of each type of image schema.

```
( *instance* *line
  deviance 19.6902
  lnparam ((0.0920 . 0.9957) 9.2944)
  length 0.0769
  components (*line-2 *line-179)
  deviant *point-88
  chn *chain-2
  ends (*point-6 *point-177))
```

Figure 4.2: An instance of a line schema

### 4.3.1. The Sketch Level

Two different types of representation are maintained at the Sketch level. First of all, the sketch is represented as a set of interconnected points (chains) as given in the input. The points are also represented in a 32 x 32 array which covers the whole image. Each cell in this array is a pointer to the list of points in the area covered by the cell. By means of this representation, questions about proximity of other chains to a certain point can be quickly answered.

### 4.3.2. The Line/region Level

For each of the chains a line hierarchy is built. Its construction is illustrated in Figure 4.3. A chain of (interconnected) points is drawn from *A* to *B*. The top line of the hierarchy (line 1) connects the end points of the chain.

Each of its two successors (line 2 and 3) connects one of the end points of its predecessor with the point of maximal distance in the curve approximated by the predecessor. This distance is called the *deviance* of the line. Line 2 has the lines 4 and 5 as successors, and line 3 is succeeded by the lines 6 and 7. This binary tree is continued until all the points in the chain are covered. In this way each chain can be described at any desired level of detail.

This representation has several disadvantages, however. For one thing, a small change in the shape of a line segment can bring about large changes in the line hierarchy. A better technique for representing the shape of line segments has been recently proposed by Mackworth and Mokhtarian (1984).

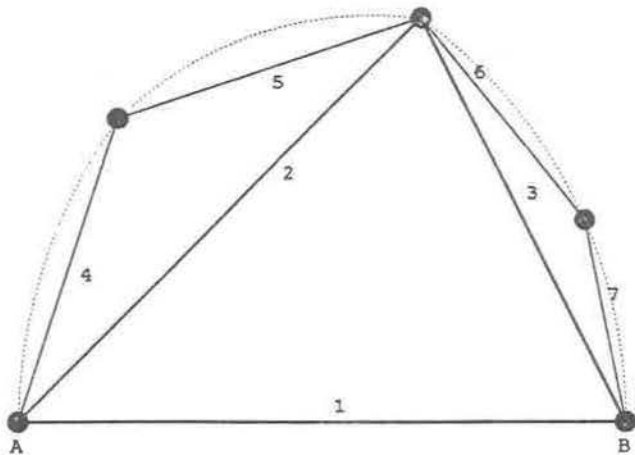


Figure 4.3: A line hierarchy

### 4.3.3. The Region Formation Process

The region formation process starts with a query as to whether the picture as a whole is empty. If the answer is negative, then the picture is subdivided into four square subpatches and the query is repeated for each of the subpatches. If a patch is empty, then no further subdivision is made. However, this subdivision of patches does not continue *ad infinitum*. Region refinement stops when a patch size of  $1/R^2$  of the total picture area is reached. For all examples shown in Chapter 5, an  $R$  value of 8 was chosen. Each set of four-connected empty patches defines a region.

The region formation process is conservatively biased. The reason for this bias is that in free hand sketches lines which are supposed to join may leave a small gap. We want to prevent a region "leakage" through these gaps. For example, leakage through a shore line that is not properly closed would cause a landmass to be interconnected with a waterbody. This would have disastrous effects on the interpretation process. The decision at which patch size to stop segmenting is a heuristic decision. The image data structures and procedures were inherited virtually unchanged from Mapsee-1 (Mackworth, 1977b).

#### 4.4. Image-to-scene Mapping

As mentioned before, the segmentation process culminates in the computation of a number of shape attributes for each chain. By means of these attributes, the image appearance of all the objects at the leaf level of the composition hierarchy in the scene domain can be described. The shape of a mountain, for instance, is constrained by means of six attributes. All these attributes are constructed on the level at which the chain is described by two lines only (see Figure 4.3). The six attributes are specified as follows:

1. The angle between the two lines has to remain within certain bounds.
2. The deviance of each line has to be very small.
3. The line length of both lines has to be approximately equal.
4. The angle of each line with the vertical has to be small.
5. The angles in 4 have to be approximately equal as well.
6. The y-coordinate of the intersection point of the two lines has to be greater than the y-coordinate of one of its end points (We do not want mountains upside down).

Each schema at the composition leaf level is described in terms of the attributes just mentioned. Table 4.1 shows this description. In this figure “+” means

must-be-there, “-” means must-not-be-there, and “\*” means may-be-there. Most of the attributes shown in the figure represent a set of attributes. The mountain shape attribute, for instance, stands for the logical *AND* of the six attributes mentioned above. The potential closure attribute represents the fact that a chain which is only partially visible in the image may actually be closed. This is always the case when it runs off the edge on both sides. Towns are uniquely described by the blob attribute.

	Potential closure	Visible closure	Mountain shape	Bridge-side shape	blob
Town	-	+	-	-	+
Road	*	*	*	*	-
River	*	-	*	*	-
Mountain	*	-	+	-	-
Bridge-side	*	-	-	+	-
Shore	*	*	*	*	-

Table 4.1: Shape attributes of the schemata at the composition leaf level

Most attributes are mutually exclusive, or can go together in certain ways only. The mountain-, bridge-, and blob-shape are mutually exclusive. The same holds for potential and visible closure. Table 4.2 shows all the possible combinations of attributes. The set of basic schemata that can satisfy these attributes can be found by replacing the “\*”’s in Table 4.1 by a “+” or “-” in all possible ways. An abstract schema is created for each set of basic schemata in Table 4.2.

For each chain, the image-to-scene process checks the value of each of the attributes in Table 4.2, and creates a (non-hypothetical) instance of the corresponding abstract schema (class).

Sets of basic schemata	Potential closure	Visible closure	Mountain shape	Bridge-side shape	blob
Town	-	+	-	-	+
Road/river	-	-	-	-	-
Road/shore	-	+	-	-	-
Road/river/shore	+	-	-	-	-
Road/river/bridge-side	-	-	-	+	-
Road/river/mountain	-	-	+	-	-
Road/river/mountain/shore	+	-	+	-	-
Road/river/bridge-side/shore	+	-	-	+	-

Table 4.2: Possible combinations between shape attributes

#### 4.5. Interpretation

##### 4.5.1. The Basic Composition and Discrimination Graphs

In Mapsee-3 object classes and relations are represented by schemata. Figure 4.4 shows the basic composition and discrimination graphs for the domain. These hierarchies have to be provided by the user. Any schema embedded in these hierarchies is called a *basic* schema. Five different levels of composition can be distinguished:

1. Object level
2. Super-object level
3. System level
4. Geo-system level
5. World level

Relational levels exist between all levels of composition. Each arc in Figure 4.4 stands for two composition relations: a *part-of* and a *parts* relation. The former is of the "must-be-part-of" type.

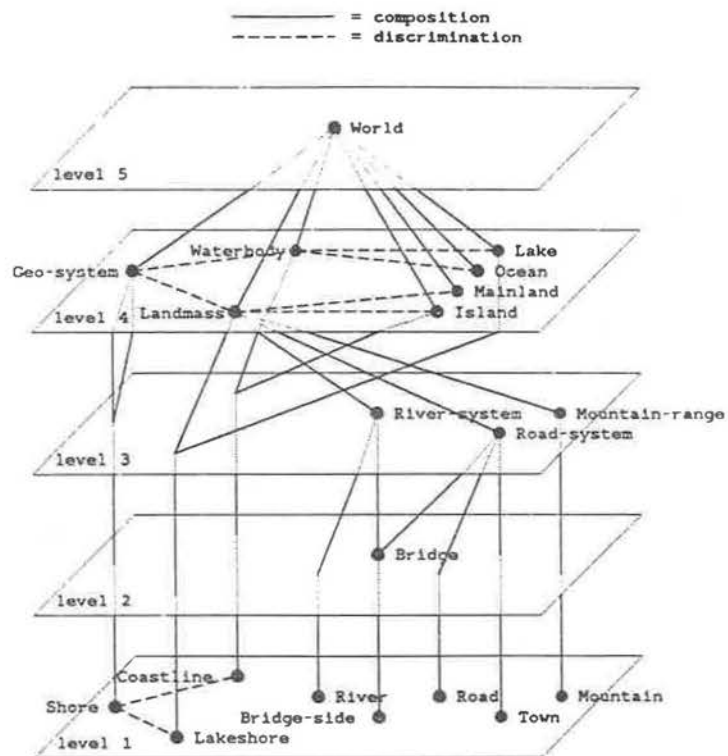


Figure 4.4: The basic composition hierarchy and discrimination graphs

```

(object '*S92 'schema
  schema-label '*S92
  type 'object
  domain 'scene
  name 'road-system
  dec-level 3
  spec-level 1
  labels '(*S92)
  links-in '(*S40 *S75 *S77 *S111i *S335 *S335i)
  links-out '(*S40i *S75i *S77i *S111 *S335 *S335i)
  parts '(*S40i *S75i *S77i)
  part-of '(*S111)
  specializations 'nil
  generalizations '(*S86 *S88)
  internal-methods 'nil
  methods '(M300 M301 M302 M313)
  disambiguating-methods '(M313)
  semi-disamb-methods '(M302 M313)
  completed-system-methods '(M335)
  M300 '(road-road-tee (*S54) (*S54))
  M301 '(road-town-tee (*S54) (*S20))
  M302 '(road-over-bridgep (*S54) (*S52))
  M313 '(mergep (*S54) (*S54))
  M335 '(incomplete-roadsystemp (*S92))
)

```

Figure 4.5: The road-system schema

Figure 4.5 shows the internal structure of the road-system schema. Each schema is identified by a unique schema label (e.g. \*S92). In correspondence with the scene schema syntax, each scene schema has the following attributes:

1. a composition level (dec-level)
2. a discrimination level (spec-level)
3. The list of schemata it points at (links-out)

4. The list of adjacent schemata pointing at the schema (links-in)
5. The potential components of the schema (parts)
6. The super-schema(ta) it must be a part of (part-of)
7. its specializations (in the discrimination graph)
8. its generalizations (in the discrimination graph)
9. Different types of methods (to be discussed in section 4.5.4)

A schema always has itself as a label. When an instance is created, it will initially be assigned the label of its parent schema. Thus, a *geo-system* instance will have the label *geo-system* at the start. When the interpretation process proceeds, however, the discrimination process may refine the label. Discrimination does not affect the description of an instance, except for its label. A *geo-system* instance can obtain the label(s) of any of its discriminations (see Figure 4.4).

Figure 4.6 below shows the internal structure of the "part-of" schema \*S77. It connects a road with a road-system. Its inverse (\*S77i) connects a road-system with a road. Like the object schemata, the "parts" and "part-of" schemata are embedded in a discrimination graph.

```
(object '*S77 'schema
  schema-label '*S77
  type 'relation
  domain 'scene
  name 'part-of
  dec-level 2.5
  spec-level 1
  labels '(*S77)
  links-in '(*S54)
  links-out '(*S92)
  parts 'nil
  part-of '(*S92)
  specializations 'nil
  generalizations '(*S69 *S71)
  inverse '*S77i
)
```

Figure 4.6: A relational schema

#### 4.5.2. Constructing the Abstract Discrimination Graph at the Composition Leaf level

Figure 4.7 shows the abstract discrimination graph at the composition leaf level. The abstract schemata are given in Table 4.2. The graphs are the result of applying the algorithm given in section 3.2.4, as the reader can verify. The only ambiguity in the graph results from the fact that *road/river/shore* could also have been subdivided as *road/river* and *shore*. The construction results in two discrimination graphs. One graph consists of a single node: *town*. The other contains the remainder of basic and abstract schemata.



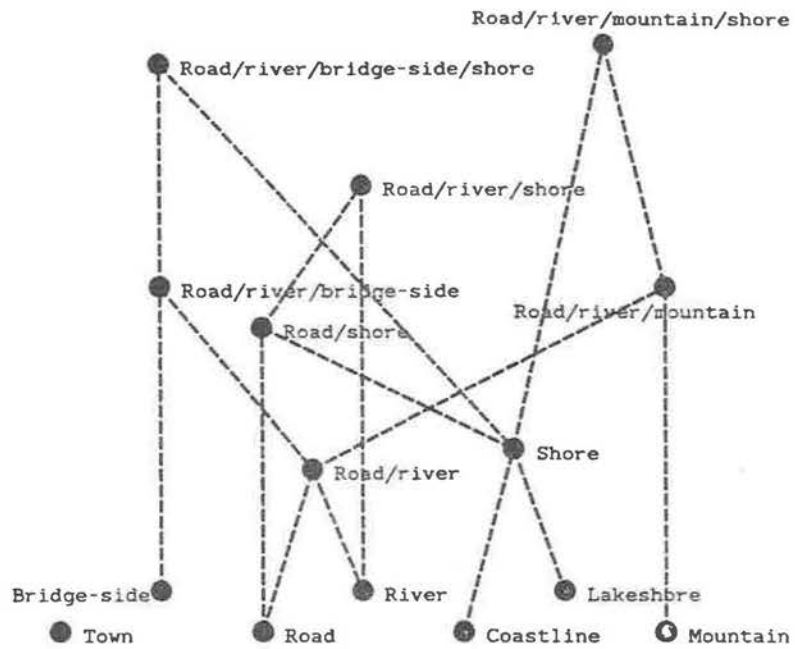


Figure 4.7: The discrimination graphs at the composition leaf level

4.5.3. Constructing the Abstract Composition Hierarchy and Discrimination Graphs

In the basic Composition hierarchy (Figure 4.4), not all schemata are represented at all levels of composition. For instance, *shore* is not represented at levels 2 and 3, whereas *road*, *river*, *mountain*, and *town* are not represented

at level 2. Our first move is therefore to create dummy representations for each of these objects at each of the missing levels. In this way the hierarchy obtains uniform depth. Figure 4.8 is the result. The dummy representations carry the same name as their component, except that a \* has been added. Thus, *river* becomes a component of *river\**.

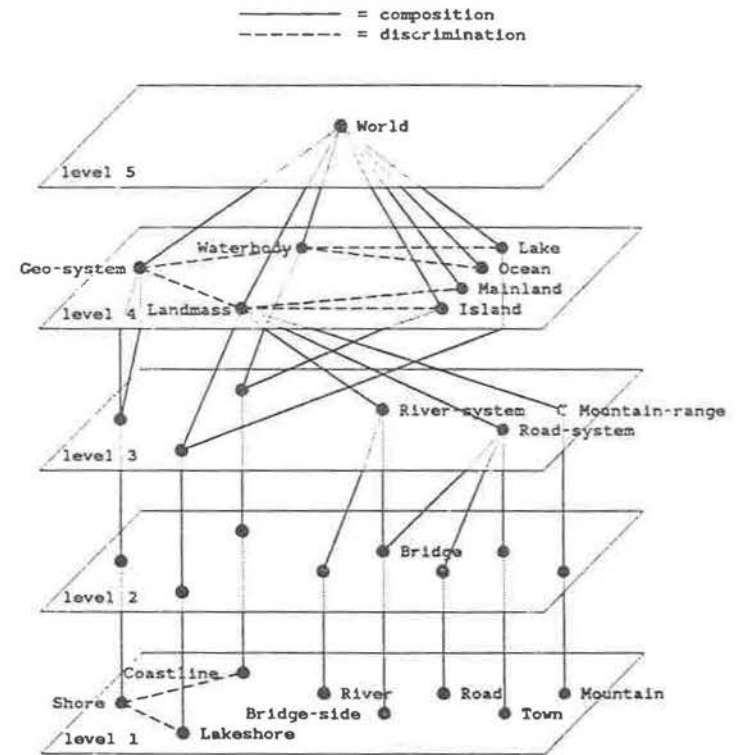


Figure 4.8: The basic composition hierarchy and discrimination graphs with dummy representations at each level of composition

The creation of dummy representations does not affect the orthogonality of the dimensions, because the meaning of a dummy representation at a particular level of composition is that it is undefined. If a particular instance obtains a dummy interpretation, then we have essentially created an undefined interpretation in the interpretation graph. One can either accommodate such an interpretation (which is what Mapsee-3 does), or remove it. Suppose  $A$  is a component of  $B$  and  $B$  is a component of  $C$ . If  $B$  becomes undefined then we can remove  $B$  by making  $A$  a component of  $C$  directly.

Applying the projection algorithm given in Appendix C, we create the discrimination graphs for level 2. This must be done in two steps because there is a relational level in between level 1 and 2. Discrimination graphs are also created at this relational level. Figure 4.9 shows the result. The intermediate relational level is not shown in this figure. Similarly, we project level 2 onto level 3 (Figure 4.10).

Note that each instance of *superroad/river/bridge* becomes a component of two different instances of *road/river-system*. This is the consequence of the modularity criterion. *Superroad/river/bridge*'s discrimination *bridge* has two super-schemata. Any generalization of *bridge* must therefore have at least two super-schemata as well.

The projection rules do not require the creation of any abstract schemata at level 4 (Figure 4.11). As a result no projection from level 4 onto level 5 is required, because all links are part of the basic composition hierarchy (Figure

4.12).

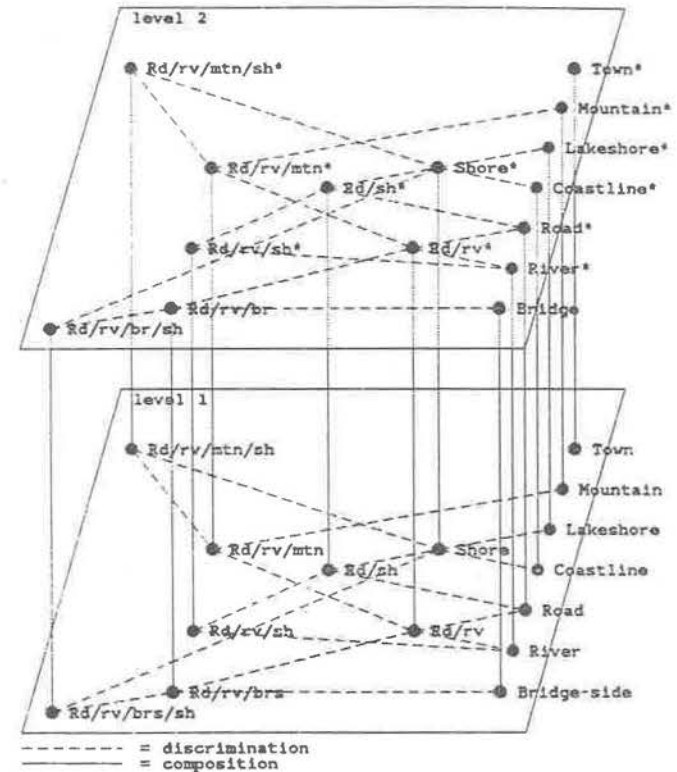


Figure 4.9: The composition hierarchy and discrimination graphs for levels 1 and 2

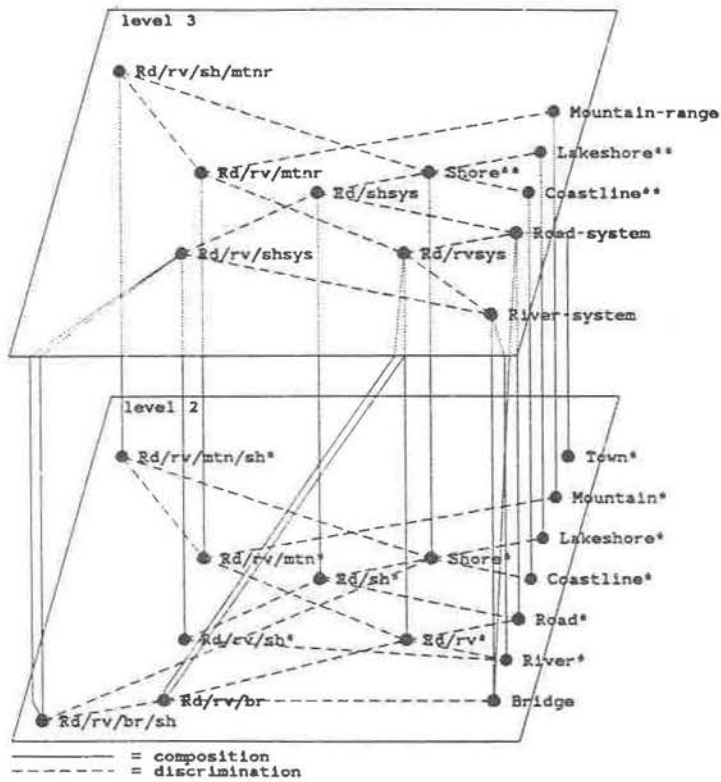


Figure 4.10: The composition hierarchy and discrimination graphs for levels 2 and 3

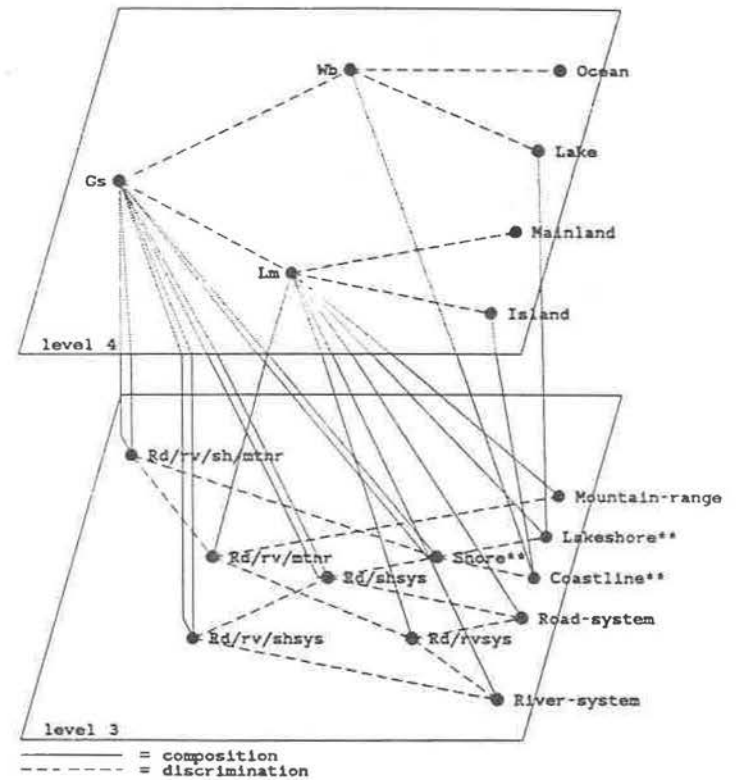


Figure 4.11: The composition hierarchy and discrimination graphs for levels 3 and 4

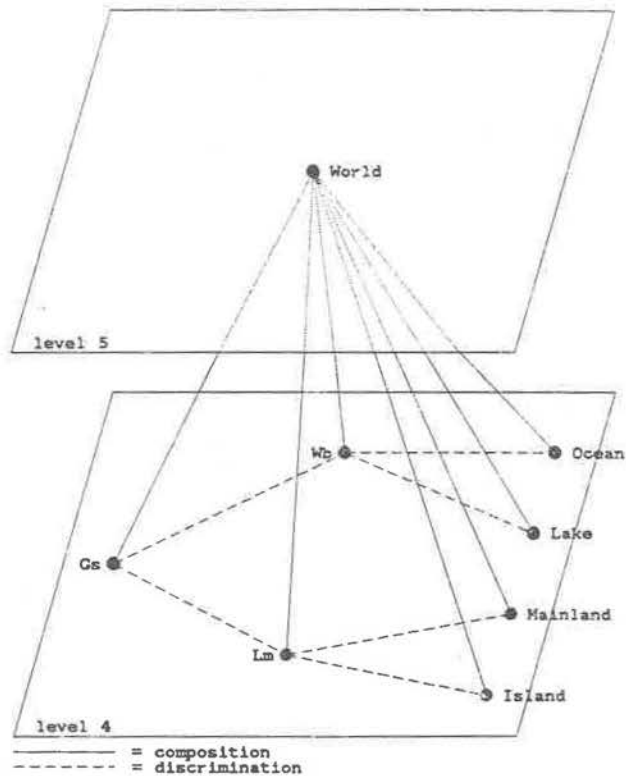


Figure 4.12: The composition hierarchy and discrimination graphs  
for levels 4 and 5

#### 4.5.4. Methods

In a Mapsee-3 schema, all attributes beginning with M followed by a number are methods. The value of such an attribute is a monadic or dyadic function which, upon successful application, creates an instance of a unary or binary (spatial) relation. Examples of methods can be found in Figure 4.5. For each of the arguments of the function, the owner schema provides a list of schemata, whose instances can fill these arguments. In Figure 4.5, for instance, the method M300 (road-road-tee) takes two arguments. Each of these arguments must be an instance of \*S54 (road).

A successful application of a method results in the instantiation of a schema representing a relation. A successful application of road-system's "road-road-tee" method, for instance, results in an instantiation of \*S300 (road-road-tee). This instance will link two road instances.

The inheritance scheme for methods discussed in section 3.4.4 has not been implemented as such. When represented in matrix format, the Mapsee-3 relations show very particular patterns. A number of method types have therefore been created, each one of which corresponds to a particular matrix pattern. Each type has its own inheritance rules and the application of these rules has the same result as the general inheritance scheme.

In correspondence with the syntax, a distinction is made between internal and external methods. The former kind represents monadic, the latter dyadic

functions. All other types correspond to particular matrix patterns. Table 4.3 lists all methods and their types.

Regular methods are characterized by a matrix containing one cell with a "+" value and "\*"s in the other cells. Table 4.4 shows an example of such a relation (road-road-tee). For semi-disambiguating methods, the matrix contains one or more rows or columns with one "+" and "-"s otherwise. The "road-over-bridge" method (Table 4.5) owned by the road-system schema is an example of this. Disambiguating methods are characterized by a matrix in which one cell has a "+" value, when the remainder of the matrix cells have "-"s. The "mountain-mountain-tee" method (Table 4.6) exemplifies this case. For all three method types it is assumed that they are able to establish a positive relationship. If the relationship is negative then all the "+"s in the matrix must be replaced by "-"s and *vice versa*.

Disambiguating methods can be inherited by all generalizations of the schemata to which the method applies. There is a restriction, however, for semi-disambiguating methods. One of the instances serving as an argument in the method's function must have a unique basic label. In the method's matrix, this label must represent a column or row which contains no "\*"s. If this condition is satisfied, then the method can be inherited by a generalization of the owner schema. In the example in Table 4.5, the second argument of the function "road-over-bridge" must have *bridge* as label. If this is the case, then any generalization of the road-system schema can inherit this method. Finally, regular methods

cannot be inherited by any abstract schemata.

Completed-system methods form a category by themselves. They are disambiguating methods with an additional constraint: can only take effect after image interpretation has been completed. Mapsee-3 is strongly data-driven. A schema's method cannot search or investigate chains that have not already been involved in the composition process. For instance, it can happen that a road has the shape of a bridge-side. Originally such a chain will be interpreted as a road/river/bridge-side. With one cycle of the interpretation process completed, this road/river/bridge-side will not be refined to a bridge, because there is no matching bridge-side. Completed-system methods have the ability to determine such things as: "single bridge-sides cannot be bridge-sides". As a result, the road/river/bridge-side will be forced to specialize to a road/river. However, such measures can be taken only *after* we have visited all chains. For this reason interpretation goes through two cycles with completed-system methods being applied in the second cycle.

The methods listed in Table 4.3 perform the following operations. All methods ending with "-tee" form a T-junction between two components. For instance, "road-road-tee" forms a relation between two roads. The "road-over-bridge" method imposes the road interpretation on any chain crossing a pair of chains forming a bridge. The "bridge-side/bridge-side" method imposes a relation with the same name on any pair of matching bridge-side-shaped chains. The disambiguating nature of this relation causes the interpretation of both chains to

be refined to bridge-side. The "surface-overlap" method ensures that geo-system components with overlapping regions become a component of one and the same geo-system instance. This method does not create a relation. The "single-world" method ensures that all geo-system instances become a component of one world instance, while not creating a relation either.

The "merge" method merges schema instances that have become redundant. No relation is created. "Island-inside-waterbody" imposes the island label on a geo-system which is surrounded by a waterbody and *vice versa*. "Lake-inside-landmass" follows the same principle. If a shore is cut off by the picture frame on both sides and one of the geo-systems adjacent to the shore is a landmass then the geo-system on the other side must be a waterbody. This interpretation is imposed by the "landmass-beside-waterbody" method. The last three methods create a relation. The "not-roadp" method deletes the road label from any chain whose end points are adjacent to a chain interpreted as a river. The "not-riverp" method does the same for a river label when it is adjacent to a road. The completed-system methods were explained before in this section.

	Method type	Method name	Owner schema
External	Regular	Road-road-tee Road-town-tee River-river-tee River-under-bridge River-shore-tee River-mountain-tee	Road-system Road-system River-system River-system Geo-system Geo-system
	Semi-disambiguating	Road-over-bridge	Road-system
	Disambiguating	Mountain-mountain-tee Bridge-side/bridge-side Surface-overlap Single-world Merge	Mountain-range Bridge Geo-system World all
Internal	Regular	Island-inside-waterbody Lake-inside-landmass Landmass-beside-waterbody	World World World
	Disambiguating	Not-roadp Not-riverp	Geo-system Geo-system
	Completed-system	Incomplete road-system Incomplete river-system Incomplete bridge	World World World

Table 4.3: Methods

	Road	River	Bridge-side	Shore	Mountain	Town
Road	+	*	*	*	*	*
River	*	*	*	*	*	*
Bridge-side	*	*	*	*	*	*
Shore	*	*	*	*	*	*
Mountain	*	*	*	*	*	*
Town	*	*	*	*	*	*

Table 4.4: Road-road-tee method

	Road	River	Bridge	Shore	Mountain	Town
Road	*	*	+	*	*	*
River	*	*	-	*	*	*
Bridge	*	*	-	*	*	*
Shore	*	*	-	*	*	*
Mountain	*	*	-	*	*	*
Town	*	*	-	*	*	*

Table 4.5: Road-over-bridge method

	Road	River	Bridge	Shore	Mountain	Town
Road	-	-	-	-	-	-
River	-	-	-	-	-	-
Bridge	-	-	-	-	-	-
Shore	-	-	-	-	-	-
Mountain	-	-	-	-	+	-
Town	-	-	-	-	-	-

Table 4.6: Mountain-mountain-tee method

#### 4.5.5. Composition and Discrimination

The control flow of the Mapsee-3 interpreter has been discussed in Chapter 3 and is shown once more in Figure 4.13. The interpreter is formally described in Appendix E. Control alternates between the interpreter's two main constituents: composition and discrimination. The latter is shown as HAC (hierarchical arc consistency) in Figure 4.13. Both processes take their input from and return their output to two different queues. Composition owns a completion queue; discrimination owns a consistency queue. These queues form the means of communication

between the two processes.

The image-to-scene process has pushed a number of instances on the consistency queue, one for each primitive. All of them are instances of schemata at the composition leaf level. These instances also form the beginning of the interpretation graph. The interpretation process operates in cycles. With an interpretation for each primitive at each level of composition as the goal, each cycle consists of taking a schema instance from the completion queue and returning a new instance which represents the previous instance at the next higher level of composition. This also has the result of extending the interpretation graph with one or more instances and relations.

Each instance starts out with the label of its parent. As each instance gets represented further up the composition hierarchy, more and more spatial relations are embedded in the interpretation graph. If the current label of an instance becomes incompatible with all of the labels of one of its neighbors then hierarchical arc consistency, using the principle of least commitment, will replace the label by one of its successors in the discrimination graph. For example, a road/shore instance (see Figure 4.9) can obtain the label coastline.

Apart from being data-driven, the interpreter operates in a breadth-first manner. At first, all instances at the object level are represented at the super-object level. Next, all super-object level instances are represented at the system level. This process continues until the world level is reached.

The line sketch at the image level in Figure 4.14 shows a river flowing under a bridge. This is the only interpretation allowed by the Mapsee semantics. No objects other than rivers are permitted under bridges. However, each of the chains is ambiguous at the start of the interpretation process. The bridge-sides can be interpreted as a road, river, or bridge-side, whereas the other two chains can be either a road, or a river. Abstract schemata exist for each of these interpretation combinations. The image-to-scene process has therefore created the instances:  $Rd/rv/brs-1$  &  $2$ , and  $Rd/rv-1$  &  $2$ . In order to provide the reader with a somewhat more detailed description of the interpretation process, we will follow the completion of the first two instances to the next level of composition up. For a complete description of the interpretation of Figure 4.14 the reader is referred to Appendix B.

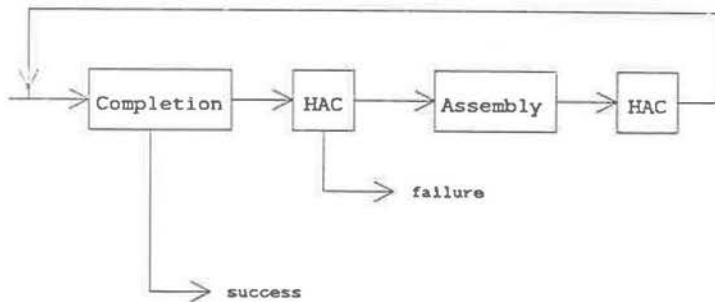


Figure 4.13: Flow chart of the interpretation process

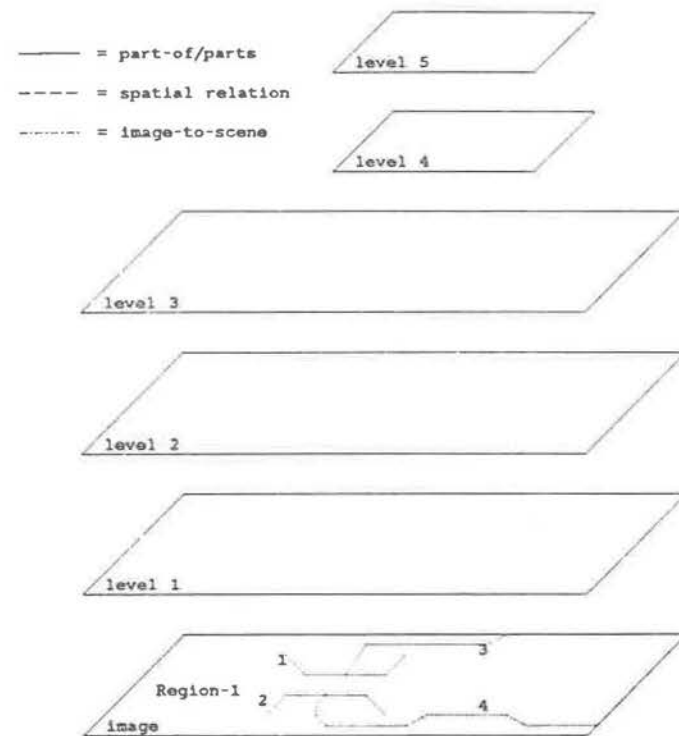


Figure 4.14: Illustration of an interpretation graph, stage 0



Name abbreviations	
Name	Abbreviation
Road/river/bridge-side	Rd/rv/brs
Road/river/bridge	Rd/rv/br
Road/river	Rd/rv
Road/river*	Rd/rv*
Road/river-system	Rd/rvsys
River-system	Rvsys
Road-system	Rdsys
Landmass	Lm
World	Wrld
River	Rv
River*	Rv*
Bridge-side	Brs
Bridge-side/bridge-side	Brs/brs
River-under-bridge	Rv/u/br

Table 4.7: Name abbreviations in the interpretation graph

#### 4.5.6. An Example

Table 4.7 shows the name abbreviations used for the object classes and relations needed for Figure 4.14. In this example we follow the interpretation process only for the two chains representing the bridge-sides. We exclusively concern ourselves with the events taking place at composition levels 1 and 2. We disregard the existence of a relational level in between. Segmentation of Figure 4.14 results in the creation of one region and four chains. Table 4.2 shows that bridge-side-shaped chains without potential closure can depict three different scene objects: a road, river, or bridge-side. The image-to-scene mapping process therefore creates two instances of the road/river/bridge-side schema at composition level 1,

$rd/rv/brs-1$  and  $rd/rv/brs-2$  (see Figure 4.15). Upon its creation each instance inherits the labels of its parent:  $rd/rv/brs$  (shown in brackets). Both instances are pushed onto the completion queue by the image-to-scene mapping process. This triggers the start of the interpretation process.

$Rd/rv/brs-1$  is picked first. The completion process uses the model constraint graph (see Figure 4.9) in order to find the super-schema of  $Rd/rv/brs$  (which is  $Rd/Rv/Br$ ). Next, it searches for an instance in  $Rd/Rv/Br$ 's super-discrimination set for instances to which a method applies enabling  $Rd/Rv/Br-1$  to become a component of this instance. The criterion for a successful application is that the super-instance has a component that can form a spatial relation with  $Rd/Rv/Br-1$ .  $Rd/Rv/Br-1$  must become a component of the super-instance whose methods succeed in establishing such a spatial relation.

In this example not a single instance has yet been established at level 2. The default rule therefore prevails. A new instance of  $Rd/Rv/Br$  is created:  $Rd/Rv/Br-1$  with the label of its parent. No links between  $Rd/Rv/Br-1$  and  $Rd/Rv/Br-1$  are yet established. First, we must ensure that the label of  $Rd/Rv/Br-1$  is compatible with the label of  $Rd/Rv/Br-1$ . Hence, the completion process pushes the link to be established on the consistency queue. At this point the completion stage is over and HAC is invoked for the first time (see Figure 4.13). HAC tests and makes consistent the label of  $Rd/Rv/Br-1$

with the label of  $Rd/Rv/Br-1$ .

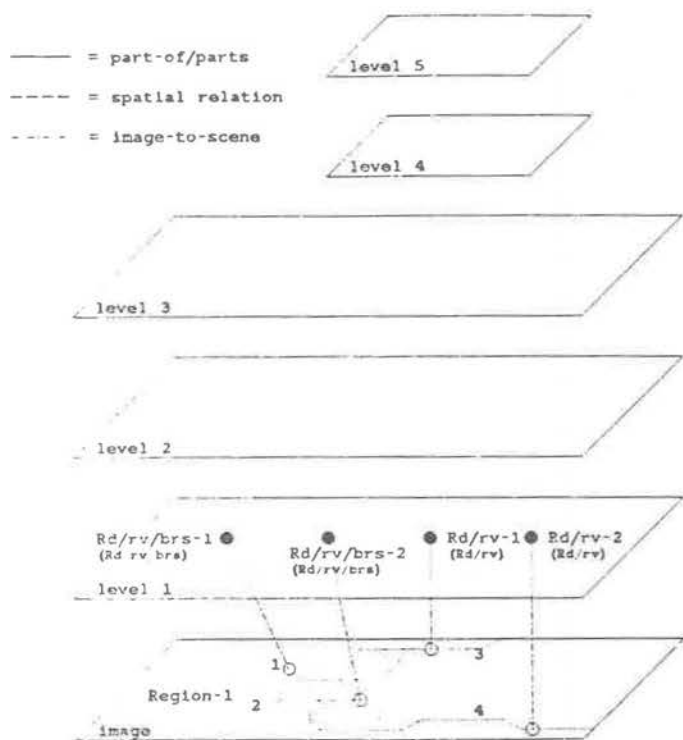


Figure 4.15: Illustration of an interpretation graph, stage 1

HAC tests  $Part-of (Rd/Rv/Br-1, Rd/Rv/Br-1)$ . It takes the labels of both instances and check whether  $Part-of (Rd/Rv/Br, Rd/Rv/Br)$  is in the model constraint graph. If this is the case, then consistency is established. If not, a replacement candidate is searched for using the discrimination graph. A replacement label for which  $Part-of (replacement, Rd/Rv/Br)$  is true constitutes a valid candidate. Failure to find a replacement candidate means failure for the whole interpretation process. In this example the label  $Rd/Rv/Br$  is consistent.

Assembly is next. This simply means that the two instances are now linked to each other, marking the beginning of the interpretation graph. Although we tested whether  $Rd/Rv/Br-1$  was consistent with  $Rd/Rv/Br-1$ , we have not yet done the opposite test. We therefore invoke HAC for a second time before returning to complete the next instance.  $Rd/rv/br-1$  is inserted into the completion queue after  $Rd/rv/brs-2$ .

Completion of the second half of the bridge is more interesting. The completion process invokes the "bridge-side/bridge-side" method. This method is a disambiguating method (see Table 4.3), owned by the bridge schema. Because disambiguating methods can be accessed by all of the owner schema's generalizations,  $Rd/rv/br$  can use this method to establish this relation between any two instances representing matching bridge-side-shaped chains. In addition, HAC now has to test the spatial relation  $P(Rd/Rv/Br, Brs/Br)$  as well. The model constraint graph in Figure 4.9 does not show any spatial relations. Only  $P(brs, brs)$

is true. The label of  $Rd/Rv/Brs-2$  is therefore refined to *bridge-side*. The label of the other bridge-side is refined likewise.

After assembly the interpretation graph consists of three objects and three relations (two composition and one spatial). During the second HAC invocation,  $Rd/Rv/Br-1$  finds its label inconsistent and refines it to *bridge*.

The completion of the two bridge-sides demonstrates how interpretation works. In the breadth-first strategy, we first complete all level 1 instances to level 2. Next, we complete all level 2 instances to level 3 etc. Figure 4.16 shows the interpretation graph for all instances up to composition level 2. The reader is referred to Appendix B for a detailed description of the other levels. The Figures 4.16 - 4.19 show the different stages of the interpretation graph.

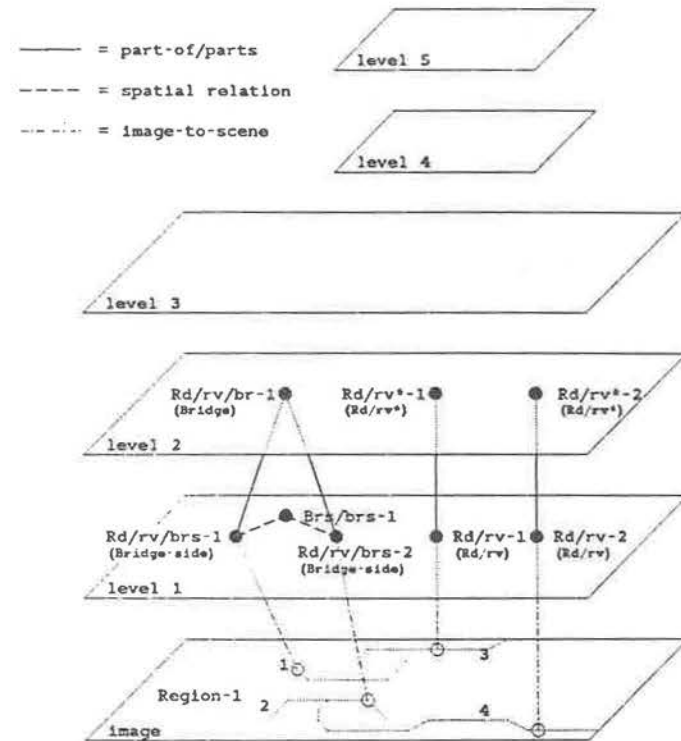


Figure 4.16: Illustration of an interpretation graph, stage 2

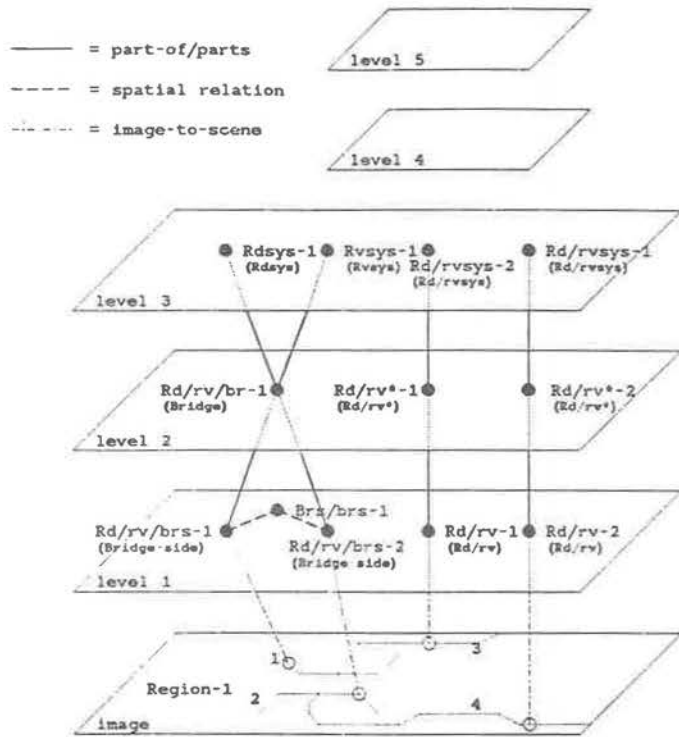


Figure 4.17: Illustration of an interpretation graph, stage 3

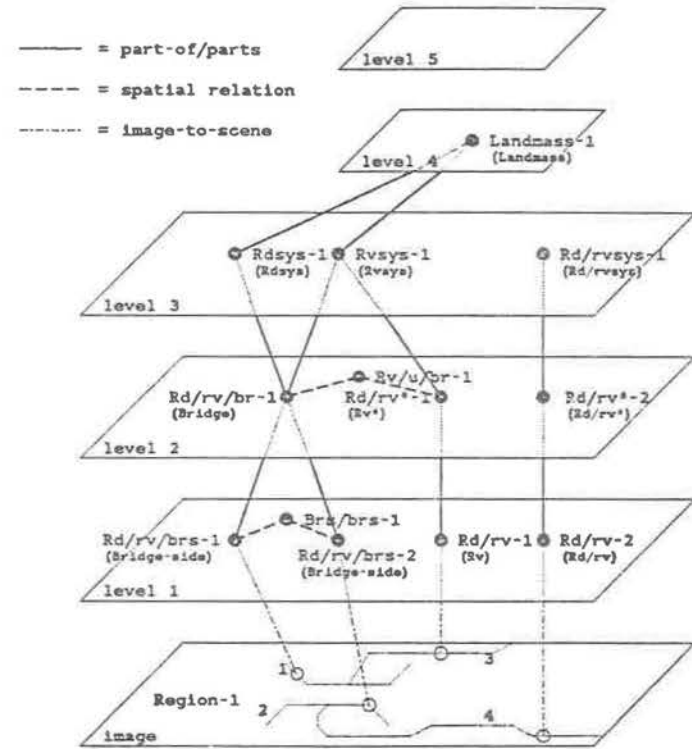


Figure 4.18: Illustration of an interpretation graph, stage 4

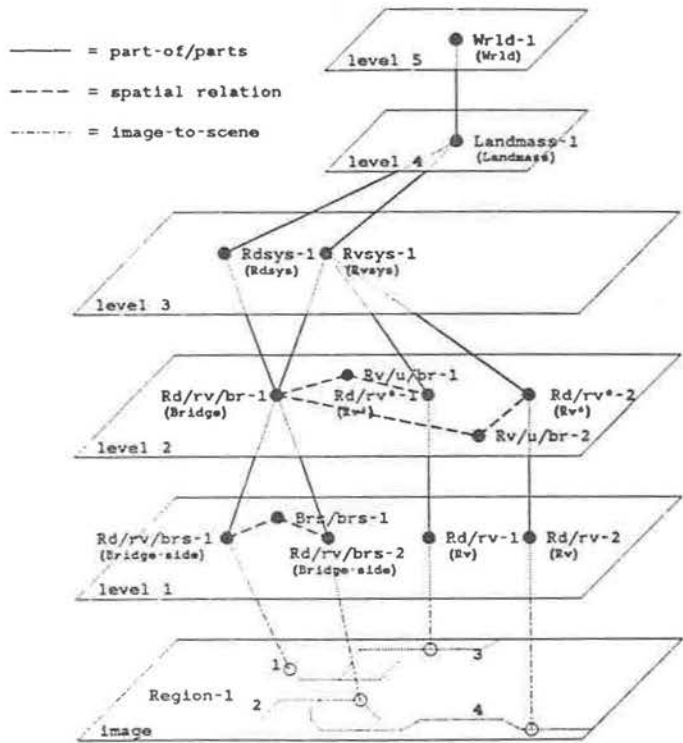


Figure 4.19: Illustration of an interpretation graph, stage 5

4.6. Summary

In this chapter the implementation of Mapsee-3 has been described. A segmentation process takes a set of plotter commands as input, and creates a set of image primitives in the form of chains and regions. An image-to-scene mapping provides the connection between the image and the scene domain. Each image primitive is represented by one schema-instance at the composition leaf level.

The interpretation process creates an interpretation graph consisting of schema-instances and their constraints. A composition process constructs this interpretation graph such that each image primitive depicts at least one instance at each level of composition in the scene domain. At the same time, a discrimination process maintains consistency between the labels of the instances in the interpretation graph.

The Mapsee-3 knowledge base has been constructed in such a way that interpretation can take place mostly from an ambiguity point of view. Object classes and relations are represented as schemata. The Mapsee-3 knowledge base is organized along three orthogonal dimensions: composition/aggregation, discrimination/generalization, and a dimension that provides the connection between the image and the scene. For each dimension Mapsee-3 has a dimension-specific process.

In the next chapter we will discuss the experimental results of trial runs of the system.

## 5. RESULTS AND DISCUSSION

### 5.1. Introduction

This chapter is divided into four sections. In section 2 we report the results of selected test runs of Mapsee-3. In section 3 we discuss the results reported in section 2. In section 4 we look at how we can relax some of the constraints of the Mapsee-3 design, thereby generalizing the design rules. In section 5 we broaden the discussion and discuss the place of discrimination vision in a general-purpose signal interpretation system.

### 5.2. Results

Mapsee-3 has been successfully tested on 10 different examples. Different examples with a varying number of chains were needed in order to obtain some performance measures for the hierarchical arc consistency algorithm and for the overall time complexity of the system. The Figures 5.1 - 5.10 show the different sketches. Apart from Figure 5.8, all examples represent real maps. In the Figures 5.11 and 5.12 which show the segmentations of Figure 5.2 and 5.5 respectively, color has no meaning. However, this is different for the Figures 5.13 - 5.22 which show the interpretations of the sketches. Table 5.1 shows the color scheme used

for these figures.

Primitive	Interpretation	Color
Chain	Shore Coastline Lakeshore Bridge-side	Magenta
	Road	Red
	River	Cyan
	Mountain Town	Yellow
Region	Landmass Mainland Island	Green
	Waterbody Ocean	Blue
	Lake	Cyan

Table 5.1: Interpretation color scheme

The Figures 5.13 - 5.22 show the interpretation results at the highest composition level (the *world* level). The world instance is the only instance at that level. The pictures show all the chains and regions depicted by the instance together with their respective interpretations. The label in the lower left corner of each picture is the label of the instance.

The Mapsee-3 graphics support system enables us to look at any instance at any desired level of composition. The Figures 5.23 - 5.28 show some of the *geo-system* level instances of the Lower Mainland of B.C. (Figure 5.5). The Figures 5.29 - 5.33 show some of the Lower Mainland instances at the *system* level. The Figures 5.34 - 5.38 show instances of some of the spatial relations.

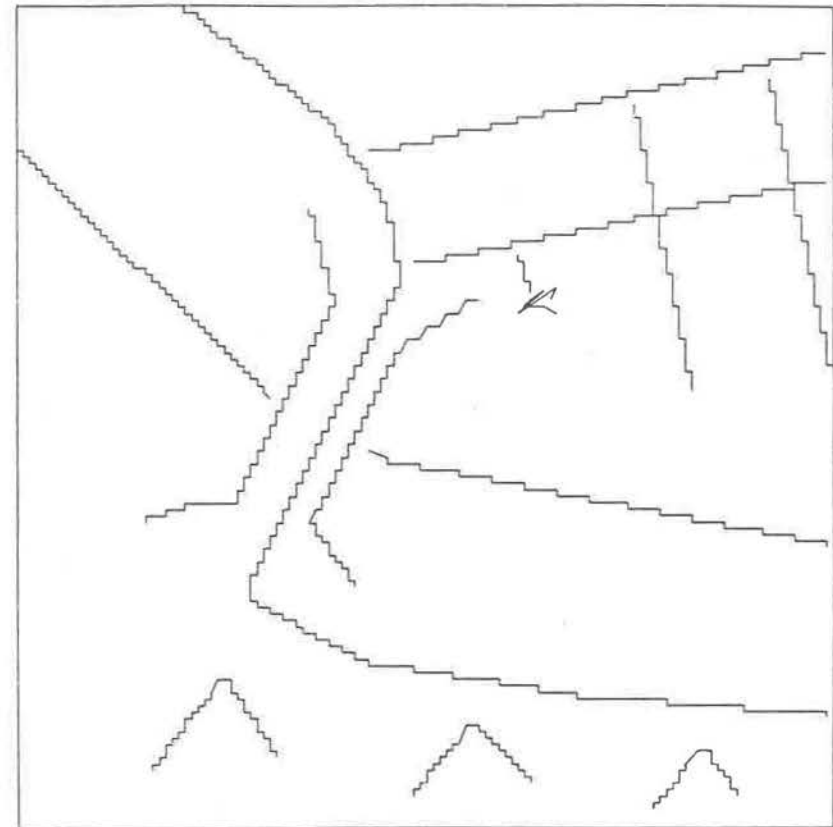


Figure 5.1: Ashcroft B.C.



Figure 5.2: Fraser Valley B.C.



Figure 5.3: Georgia Strait B.C.



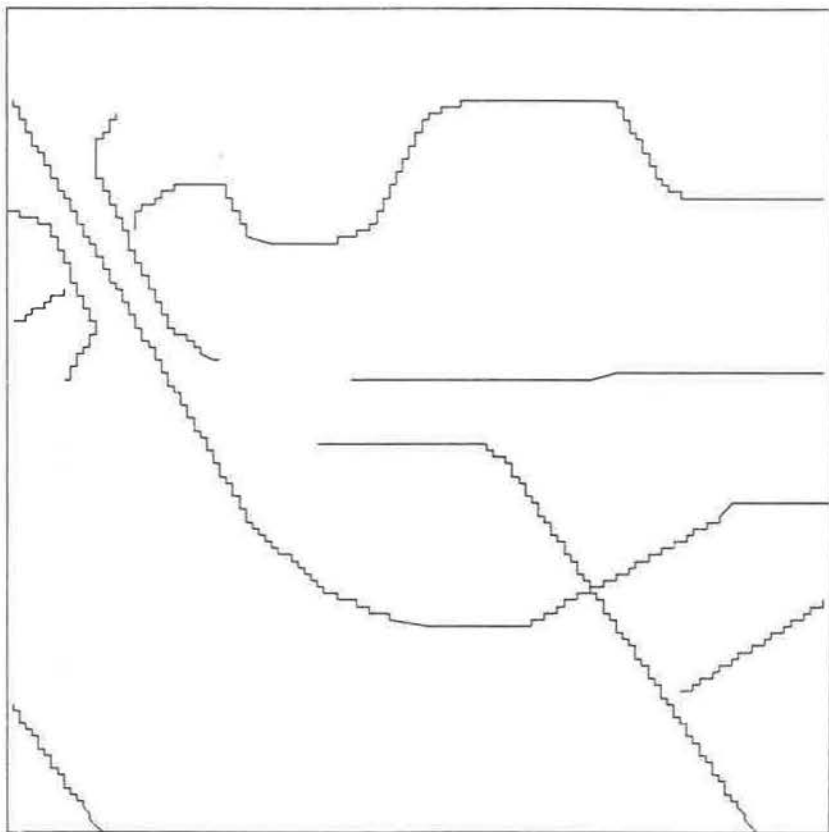


Figure 5.4: Houston B.C.



Figure 5.5: Lower Mainland B.C.

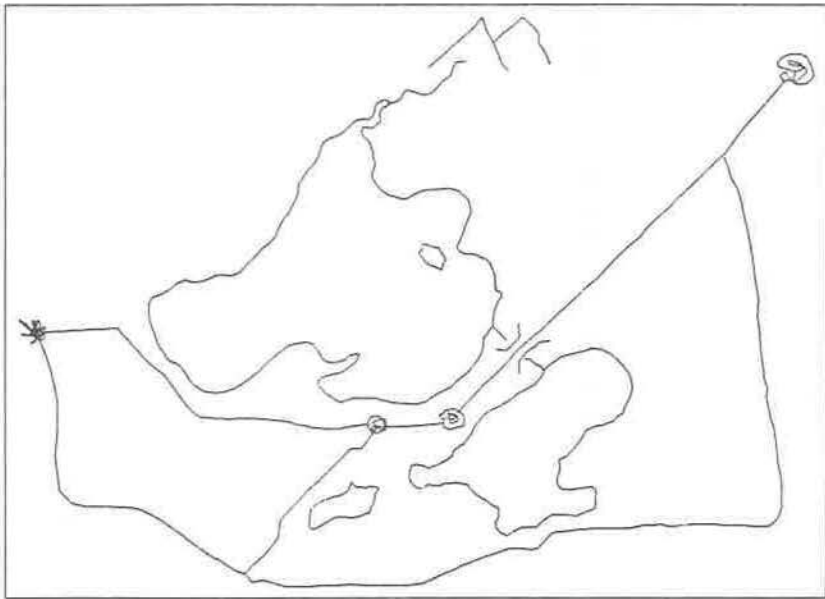


Figure 5.6: Madison, Wisconsin

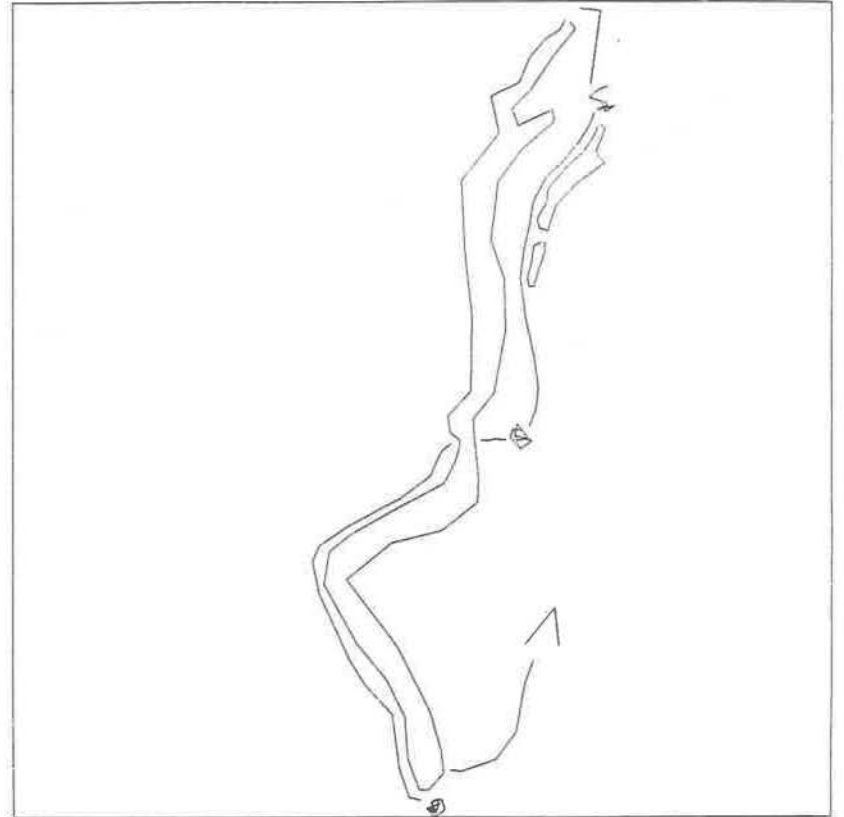


Figure 5.7: Okanagan B.C.

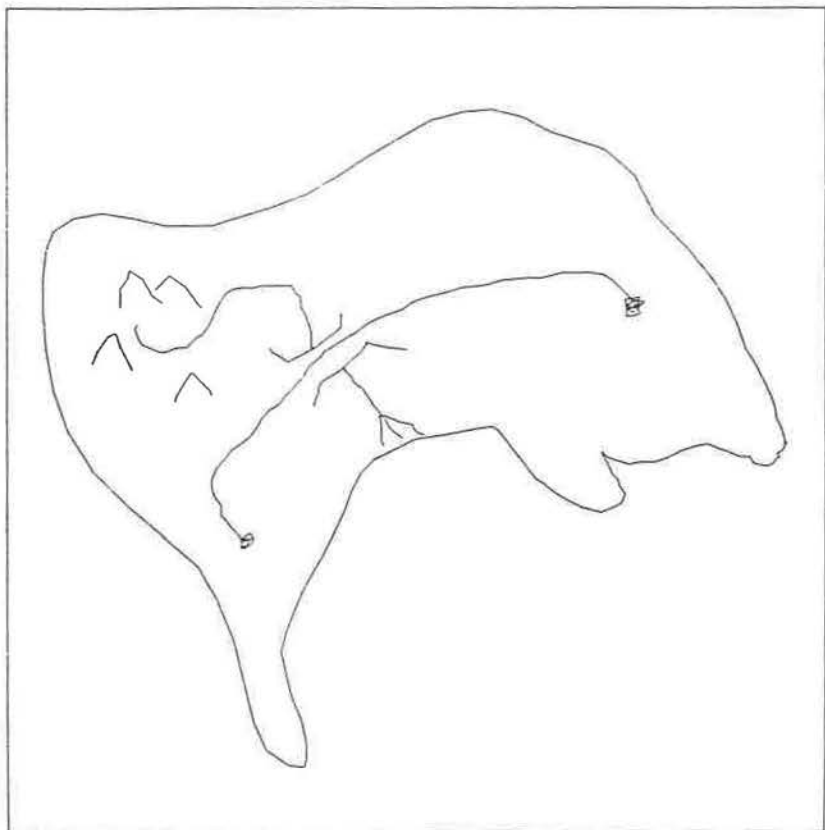


Figure 5.8: Porpoise Island (artificial)

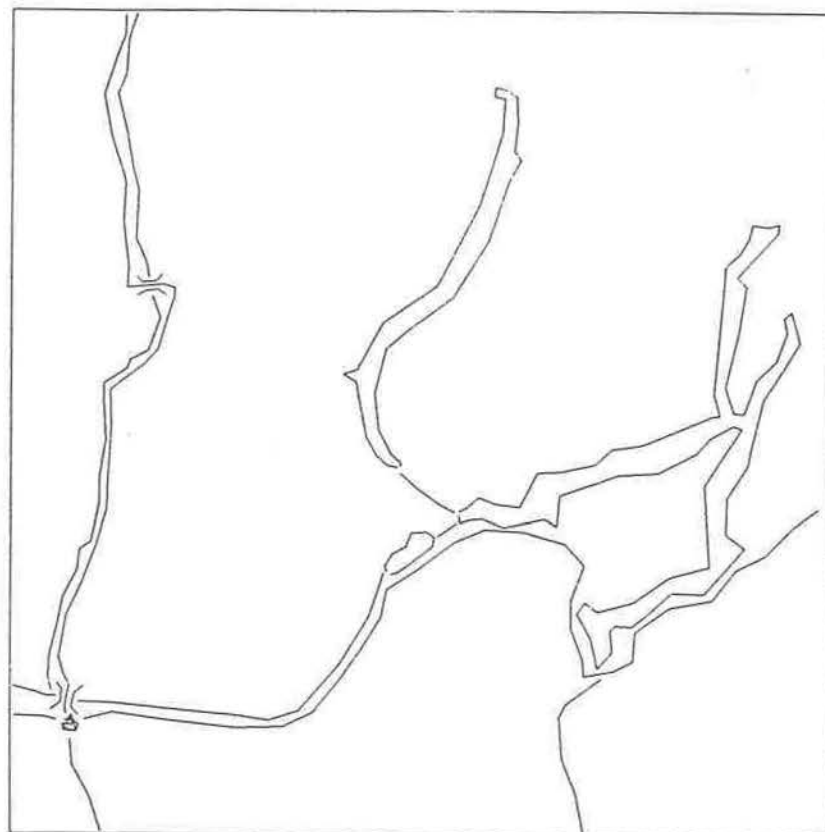


Figure 5.9: Shuswap Lake B.C.



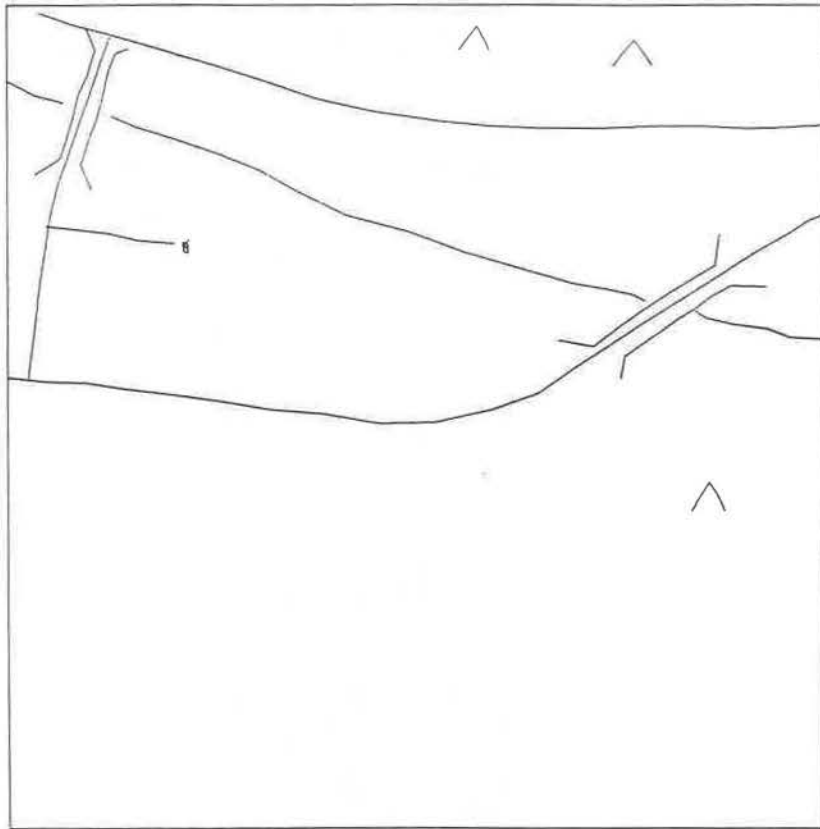


Figure 5.10: Spences Bridge B.C.

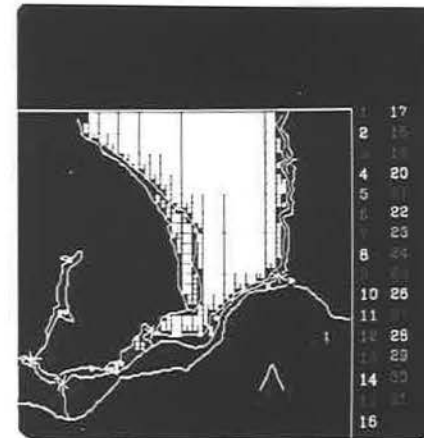


Figure 5.11: segmentation of Fraser Valley

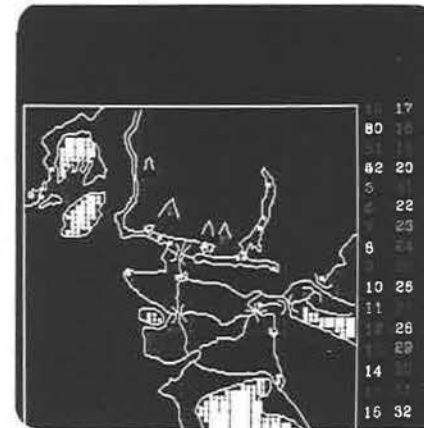


Figure 5.12: segmentation of Lower Mainland

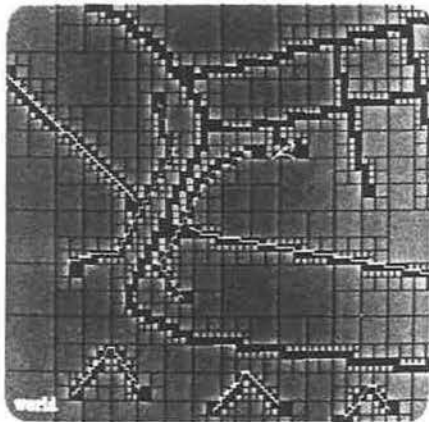


Figure 5.13: Ashcroft interpretation

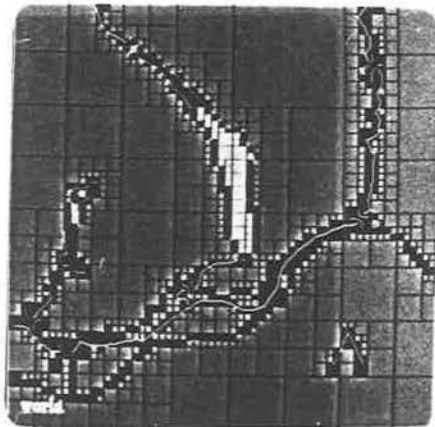


Figure 5.14: Fraser Valley interpretation

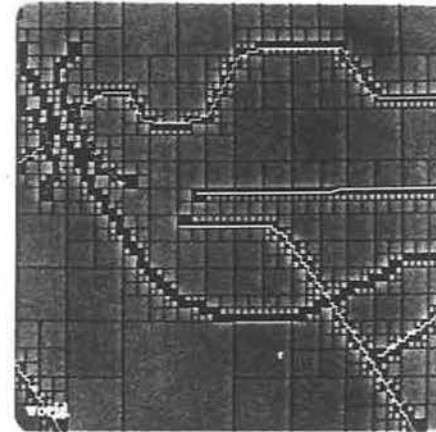


Figure 5.15: Houston interpretation

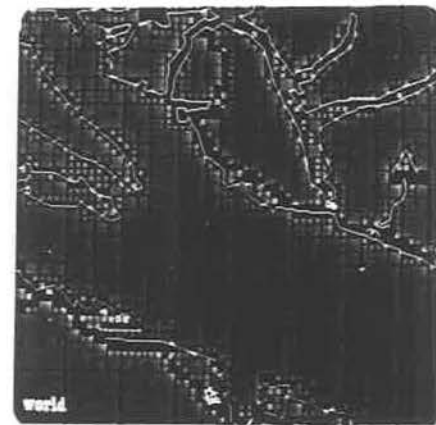


Figure 5.16: Georgia Strait interpretation

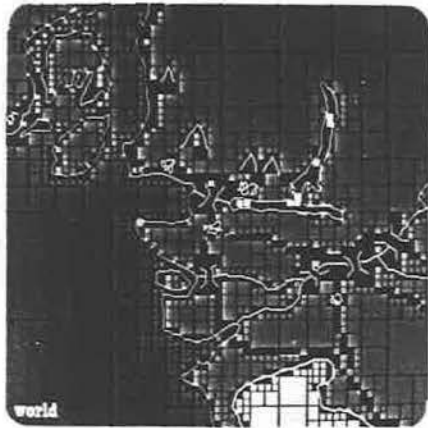


Figure 5.17: Lower Mainland interpretation

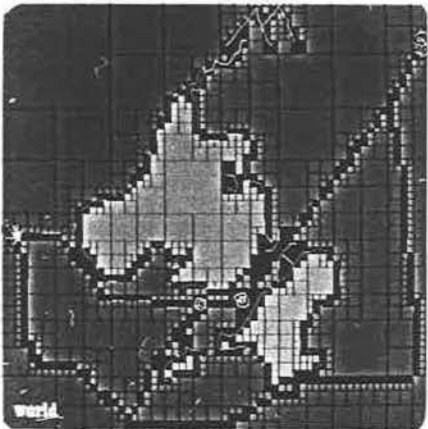


Figure 5.18: Madison interpretation

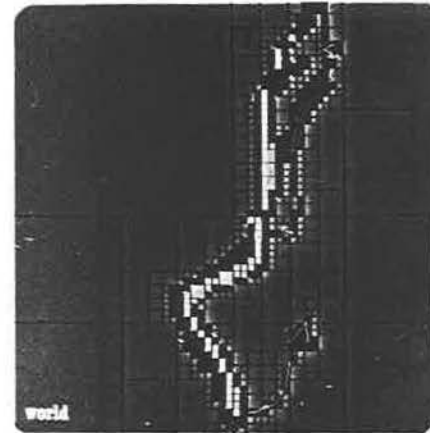


Figure 5.19: Okanagan interpretation

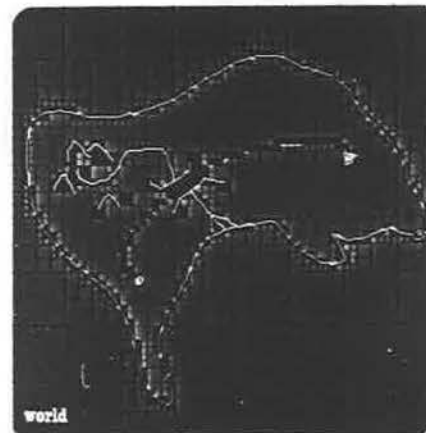


Figure 5.20: Porpoise interpretation

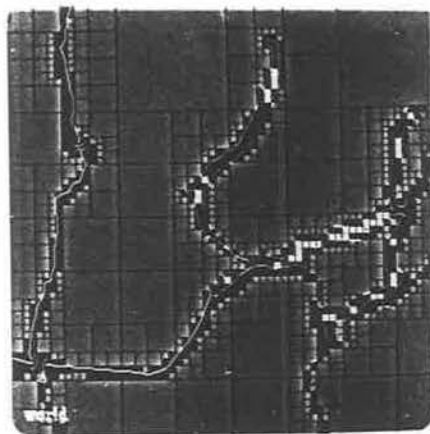


Figure 5.21: Shuswap interpretation

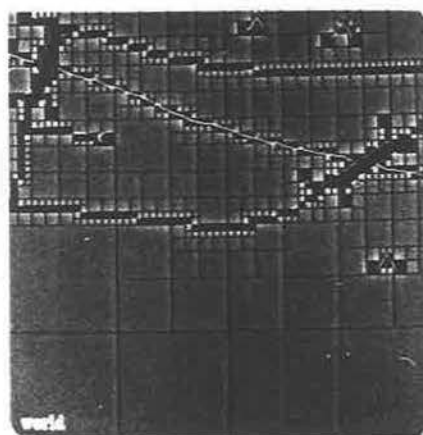


Figure 5.22: Spences Bridge interpretation



Figure 5.23: Landmass of Lower Mainland



Figure 5.24: Howe Sound





Figure 5.25: Indian Arm



Figure 5.26: Gambier Island



Figure 5.27: Boundary Bay



Figure 5.28: Keats Island



Figure 5.29: Lower Mainland road-system



Figure 5.30: Lower Mainland river-system



Figure 5.31: Shore of Indian Arm



Figure 5.32: Coastline of Gambier Island



Figure 5.33: Shore of Lower Mainland



Figure 5.34: T-junction of roads

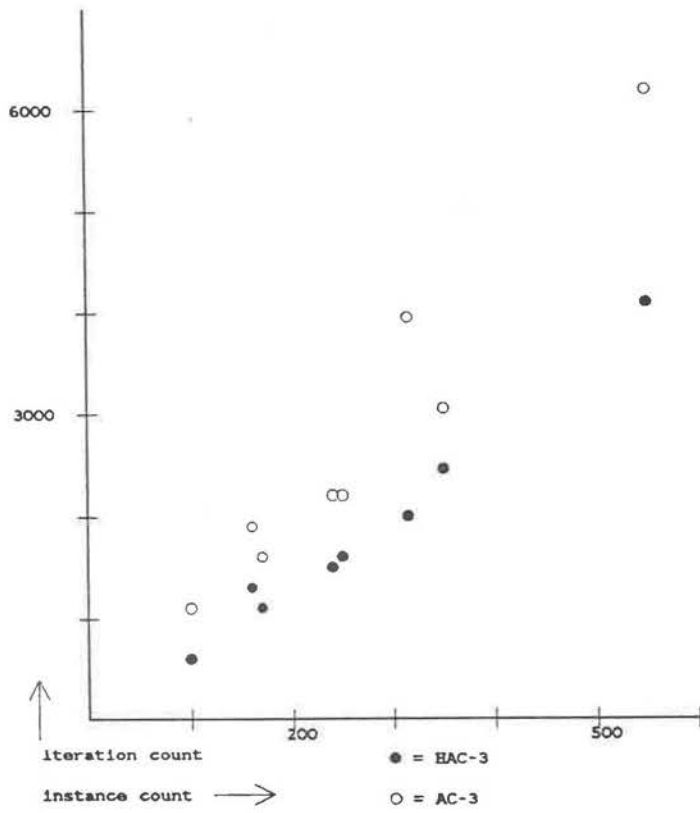
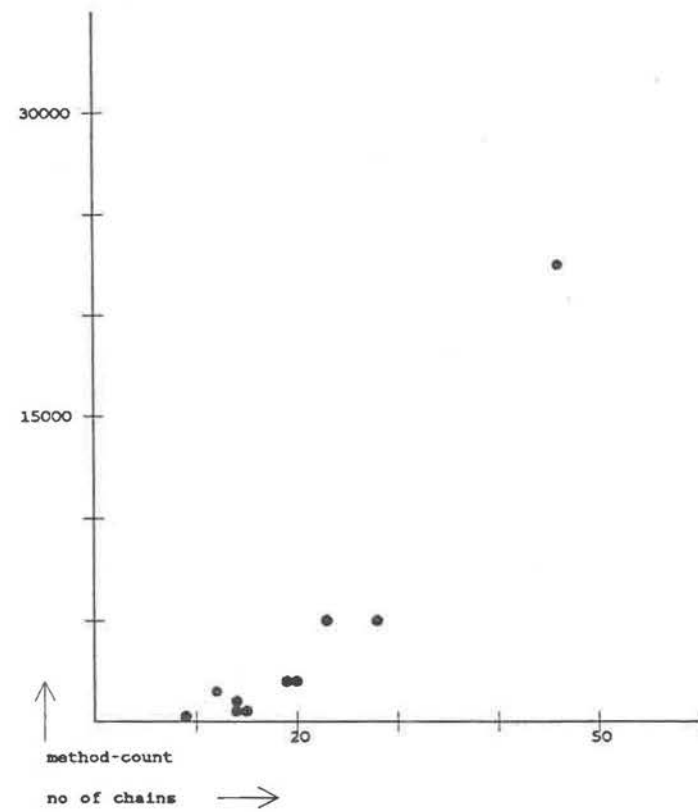


Figure 5.35: T-junction of rivers



Figure 5.36: A road crossing a bridge



Figure 5.40: Number of instances *versus* iteration countFigure 5.41: Number of chains *versus* method count

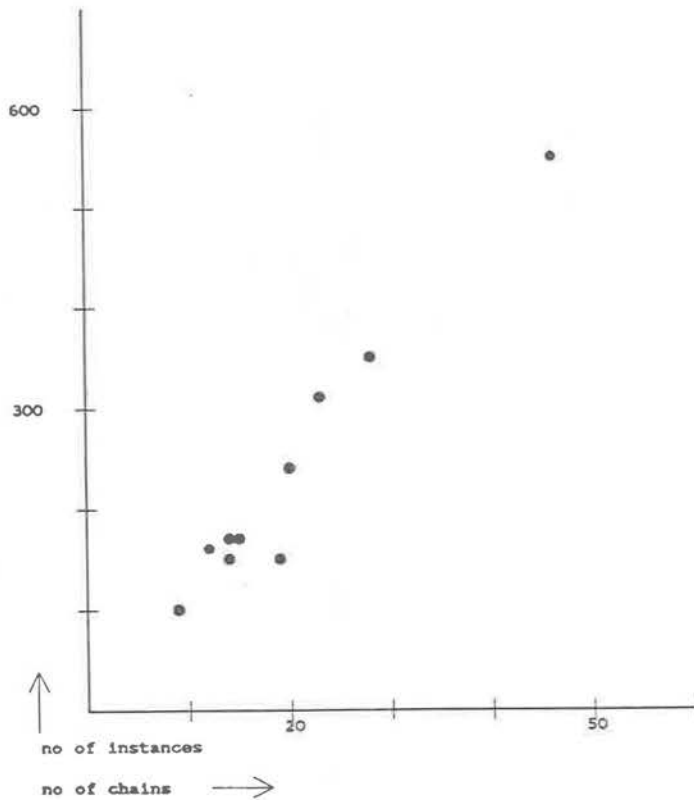


Figure 5.42: Number of chains *versus* number of instances

For each of the examples the behavior of HAC-3 was compared with the behavior of AC-3. For both HAC-3 and AC-3 each label comparison was counted as one iteration. During the test run, the HAC-3 iterations were counted, while the behavior of AC-3 was simulated. In order to count the iterations for AC-3, each label was expanded into its representation at the leaves of the discrimination graph.

The results are shown in two different ways. Figure 5.39 shows the HAC and AC iteration count plotted against the number of chains for each example.<sup>1</sup> Figure 5.40 shows the iterations plotted against the number of (scene) schema-instances in the final interpretation of each example. Note that the number of chains is a measure for the complexity of the input, the number of instances is a measure for the complexity of the output. These measures have been plotted against each other in Figure 5.42.

During discrimination the system spends all of its time on label testing and propagation. During composition, on the other hand, most of the time is spent on method application. In Figure 5.41 we have plotted the number of chains against the number of external methods applied during each test run. The latter has been taken as a time complexity measure for the system as a whole.

Table 5.2 shows the correlation coefficient and the residual variance resulting from a best linear fit between the number of chains and the iteration counts for

<sup>1</sup>Two different examples both have 14 chains. In order to avoid confusion, the smallest AC-3 value goes with the smallest HAC-3 value.

AC-3 and HAC-3 in Figure 5.39. The Tables 5.3, 5.4, and 5.5 show the same data for the Figures 5.40, 5.41, and 5.42 respectively.

For the purpose of discussion in the next section, we should note the following phenomena in the results shown:

1. An instance's label is not always refined to a leaf level label. The magenta geo-system in the Figures 5.17 and 5.27, for instance, has not refined its label at all. Similarly, the Lower Mainland shoreline in Figure 5.33 has not been maximally refined.
2. Regions which are not connected during segmentation can still become part of one and the same geo-system. As an example, compare the lake in the Lower Mainland in Figure 5.12 with Figure 5.17. As well, areas too small for a region to be formed can still become a geo-system. Figure 5.28 illustrates this phenomenon.
3. In the iteration counts plotted in the Figures 5.39 and 5.40, HAC-3 does consistently better than AC-3. Both appear to be highly correlated with the number of chains.
4. The method count is also highly correlated with the number of chains. However, the Lower Mainland with its 46 chains is a potential indicator that the relationship may be non-linear.
5. The number of chains is highly correlated with the number of instances in the interpretation graph.

	HAC-3	AC-3
Correlation Coefficient	0.99	0.95
Residual Variance	3.45	11.818

Table 5.2: Number of chains *versus* iteration count

	HAC-3	AC-3
Correlation Coefficient	0.99	0.97
Residual Variance	544.74	1208.9

Table 5.3: Number of instances *versus* iteration count

Correlation Coefficient	0.94
Residual Variance	13.018

Table 5.4: Number of chains *versus* method count

Correlation Coefficient	0.99
Residual Variance	2.1721

Table 5.5: Number of chains *versus* number of instances

### 5.3. Discussion of Results

#### 5.3.1. Robustness

One criterion for evaluating a system is its robustness, that is, its ability to cope with errors such as inappropriate segmentation. If we compare the Lower Mainland segmentation (Figure 5.12) with its interpretation (Figure 5.17) then we can observe that under certain conditions Mapsee-3 will merge a number of non-connected regions into one geo-system. The merging of the regions of the lake exemplifies this process. Because of a conservative bias, the segmentation process stops prematurely. The interpretation process, however, overcomes the problem when it notices that all regions are surrounded by a shore.

This dissertation would not be balanced without mentioning some problems as well. With respect to robustness, there is a particular form of interpretation-driven segmentation that is not achieved. The formation of junctions is an important aspect of segmentation. The decision whether or not to form a junction is made in the interpretation process, using parameters with fixed values. For a "road-road-tee", for instance, the distance between the end point of the stem and the bar has to be below a certain maximum. If this distance is exceeded in a slopily drawn sketch then the junction is not found. Obviously, a form of interpretation-driven dynamic thresholding is needed, but this has not been implemented.

#### 5.3.2. Graceful Degradation

Perhaps one of the most elegant features of Mapsee-3 is that interpretations degrade gracefully as the information content of the image diminishes. By a diminishing information content we do not mean a poorer quality of picture. Rather, we mean objects displayed in an image under such conditions that they cannot be recognized beyond a certain level of discrimination.

This phenomenon is a natural one. This was illustrated in the example discussed before. Under favorable conditions we can recognize a car up to its make and year. This is not possible if the car is covered with a foot of snow. Mapsee-3 shows exactly this phenomenon. The geo-system in Figure 5.17 is in reality a waterbody connected with the waterbody on the left. The shore adjacent to the waterbody and the chain adjacent to the geo-system are actually one and the same chain, but this cannot be seen in the picture.

The chain adjacent to the geo-system in Figure 5.17 could be interpreted as *road*, *river*, or *shore*. No constraints are available to decide upon the correct interpretation. In the *road* and *river* case the geo-system would become a landmass. In the *shore* case it would become a waterbody. As the system is unable to decide between *landmass* and *waterbody* it follows the principle of least commitment and remains at *geo-system*. As mentioned before, some well known vision systems such as ACRONYM do not follow this principle.



### 5.3.3. Domain Independence

The design of Mapsee-3 is largely domain-independent. The three knowledge representation dimensions and the unit of knowledge representation, the schema, provide only a format in which domain-dependent knowledge can be inserted. The processes operating on the different knowledge representation dimensions, are also largely domain-independent.

The image-to-scene process can deal with any domain that produces features each of which depicts one or more models. The discrimination process is a constraint propagation process which is domain-independent. Composition is also largely domain-independent. It does not need to know about any particularities of a schema as long as it can access the schema's composition relation. The only domain-dependent aspect of composition is formed by the schema's methods. The methods themselves require expert knowledge of the structure and constraints of a particular domain.

A further sign of domain-independence of the Mapsee-3 scene knowledge base is provided by the fact that some of the design principles underlying the Hearsay-II speech understanding system (Erman and Lesser, 1980) are very similar to the one's underlying Mapsee-3. The design principles of the Hearsay-II system have been applied to a wide variety of signal processing domains. The Hearsay-II blackboard consists, among other things, of a multi-level composition hierarchy. Hearsay-II's knowledge sources can be compared with Mapsee's

methods, because both contain domain-dependent knowledge. As well, Hearsay-II has, like Mapsee-3, a data driven control structure. Hearsay-II does not use discrimination graphs, but it is easy to argue that they would be useful in the speech understanding domain. Like line sketches, speech wave forms allow for many possible local interpretations. There are many words that sound alike but which have different meanings. Discrimination graphs could be based on similarities of that nature.

The Mapsee-3 image knowledge base, on the other hand, is rather domain-dependent. The schemata used are intended to describe line sketches. They are only domain-independent to the degree that they can be used for any line sketch domain, be it sketch maps or line drawings of human faces. However, as we mentioned before, this dissertation has focused on the representation of the scene domain. The image schemata were inherited from the Mapsee-2 system.

### 5.3.4. Modularity

Mapsee-3 is very modular in both representation and control. Its knowledge representation dimensions are orthogonal. For each dimension there is a particular process that operates in that dimension only. It should be emphasized that, as a result of the fact that ambiguous and hypothetical interpretations are represented along one knowledge representation dimension only, more process modularity can be achieved than would have been the case with a hypothetical

approach. A comparison between Mapsee-2 and Mapsee-3 will show why.

Figure 5.43 shows a sketch consisting of a shore line and a mountain. Figure 5.44 shows the composition hierarchy these concepts are embedded in. In Mapsee-2 a closed line-segment forms a cue for a shore only. Hence, shores are always non-hypothetical. A shore becomes a component of two geo-systems, an inner- and an outer-geo-system. The mountain shaped line-segment, on the other hand, depicts several objects including a mountain. As a result, the mountain is hypothetical. The modularity problem starts when a hypothetical mountain-range completes to geo-system. The inner-geo-system's label has to be refined to *landmass*, but at the same time the shore's label has to be refined to *coastline*. All these discriminations are hypothetical, because the mountain-range is hypothetical. However, the shore label is non-hypothetical. In order to resolve this contradiction, the discrimination process has to be interrupted in the middle of constraint satisfaction in order to create a new hypothetical instance for shore with the label *coastline*. Once the interpretation graph has been adapted, discrimination can continue.

In Mapsee-3 this never happens. The mountain-range completes as a "road/river/mountain-range" which is non-hypothetical. All three objects: road-system, river-system, and mountain-range require the geo-system to become a landmass, and shore is refined to *coastline*. No interruption of the constraint satisfaction process is required and no structural changes in the interpretation graph are asked for. The interactive behavior between composition and discrimi-

nation is symptomatic for the hypothetical approach.

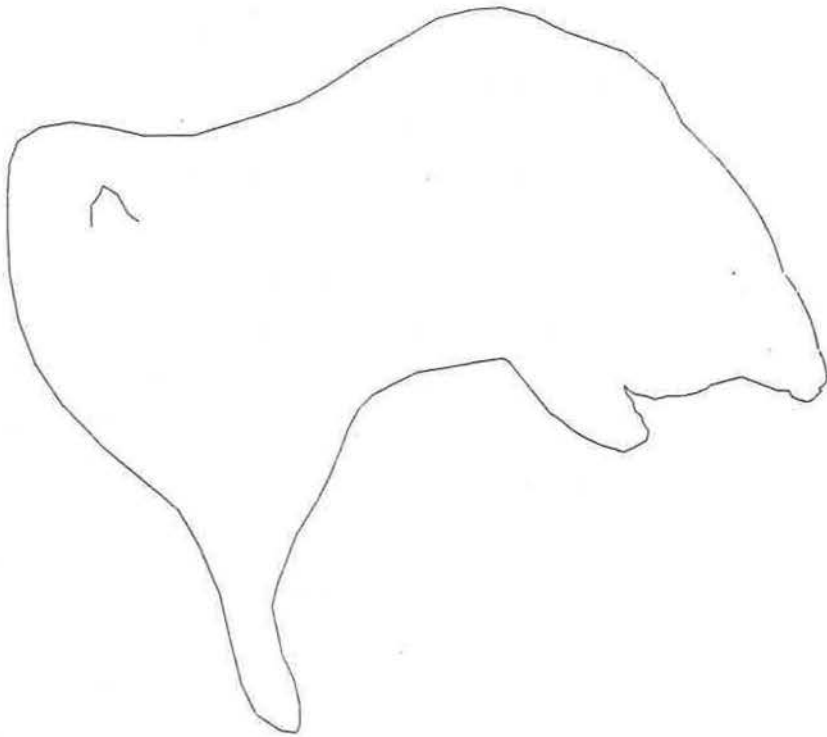


Figure 5.43: An island with a mountain

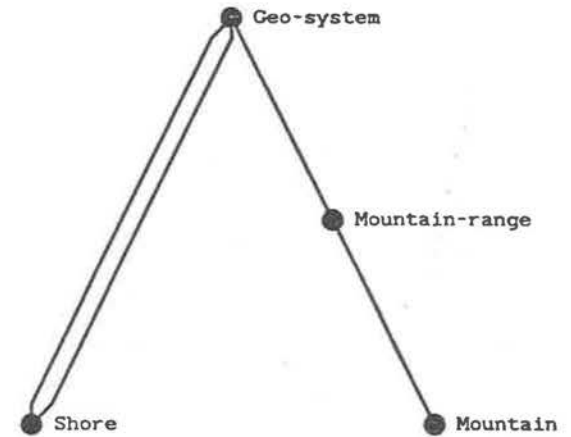


Figure 5.44: Composition hierarchy for the island/ mountain example

### 5.3.5. Efficiency

Mapsee-3 is efficient with respect to two different measures. The first one is the hierarchical representation of the domain of each variable as opposed to a set representation. HAC-3 operates on the former, AC-3 on the latter. A second measure is the complexity of the interpretation graph itself. The next two subsections deal with each of these measures.

### 5.3.5.1. HAC-3 versus AC-3

For all examples HAC-3 outperforms AC-3. Intuitively, this can be explained by the fact that the domain size is always smaller for HAC-3 than AC-3. The one exception to this rule occurs, naturally, when AC-3 has only one label in its domain. As long as the domain size in HAC-3 remains small, we can expect a good performance for this algorithm. HAC-3 will compare even more favorably to AC-3 when the number of levels in the graph increases. The Mapsee-3 discrimination graph consists of a maximum of four levels. For most chains, however, only two or three levels are used.

Theoretically, HAC-3 does not always outperform AC-3. A formal study of the time complexity behavior of HAC-3 for binary discrimination graphs is reported in Mackworth, Mulder, and Havens (1985). If we define  $a$  as the domain size, and  $e$  as the number of arcs in the interpretation graph, then both AC-3 and HAC-3 are of  $O(a^3e)$ . Asymptotically, however, the time complexity of HAC is  $4a^3e$  compared to  $2a^3e$  for AC-3. However, in an appropriately structured discrimination graph, it is reasonable to assume that there is only one label active in each variable's domain. With  $n$  defined as the number of nodes in the graph, the worst case complexity for HAC under the specified condition is  $O((e + 3n/2) \log a)$  which is remarkably better than AC-3's  $O(a^3e)$ .

For both HAC-3 and AC-3 the number of chains and the iteration count are highly correlated. This is not very surprising, because both algorithms are linear

in the number of instances in the interpretation graph, if the graph is planar<sup>2</sup>. This will usually be the case in Mapsee-3. As well, there is a linear relationship between the number of chains and the number of instances in the interpretation graph. The value of the correlation coefficient in Table 5.5 demonstrates this. For each chain there can be no more instances than there are levels in the composition hierarchy (9 in Mapsee-3). Each chain can be involved in a few spatial relationships as well. This adds about 3 instances per chain. Hence, one can expect about 12 instances in the interpretation graph for each chain. Figure 5.42 shows the correctness of this rough calculation.

As a result of the hierarchical representation, the domain size in HAC-3 will remain fairly constant. If we can treat the domain size as a constant then we can expect a strong linear relationship between the number of chains and the iteration count. Figure 5.39 shows this is the case. In AC-3, on the other hand, the domain size is not constant. It starts out large and gradually decreases in size as interpretation progresses. As a result, we can expect much more variance in the AC-3 iteration count. Indeed, the residual variance is considerably larger for AC-3 than it is for HAC-3 (Table 5.2).

<sup>2</sup>This was proved for AC-3 in Mackworth and Freuder (1982), and HAC-3's propagation behavior is identical to AC-3.

### 5.3.5.2. Complexity of the Interpretation Graph

In the previous subsection we argued that the number of nodes in the Mapsee-3 interpretation graph is linear with respect to the number of chains in the image. In this respect the Mapsee-3 approach is a major departure from Mapsee-2, in which each hypothesis and possible combination between different hypotheses is represented as a node in the interpretation graph. In particular, in relatively underconstrained images this leads to an exponential growth of the interpretation graph, as the number of chains increases.

### 5.3.6. Overall Complexity of the Interpretation Process

A good indicator of the complexity of the interpretation process is the amount of search that needs to be done. In Mapsee-3 most of the search is done by external methods. Hence, we have used the number of times an external method was applied during a trial run as an indicator of the overall time taken by the system.

In Mapsee-3 an external method establishes a spatial relationship between 2 chains. The number of pairwise comparisons one can make between  $n$  chains is polynomial in the number of chains. The correlation coefficient in Table 5.4, however, indicates a high correlation. A visual inspection of Figure 5.41 reveals, that the good linear fit is mainly due to the images with a relatively small number of

chains. The Lower Mainland, on the other hand, with its 46 chains appears to deviate from this pattern, possibly indicating a polynomial trend. Although no attempt was made in Mapsee-3 to curb a potential explosion in the number of method applications, this is possible. In Mapsee-3 relations can be established between adjacent chains only. Hence, one can restrict the number of applicable instances in a schema's superdiscrimination set by considering only those methods of instances that are depicted by an adjacent chain.

## 5.4. Generalizing Mapsee-3

### 5.4.1. Constructing an Abstract Composition Hierarchy with Relaxed Restrictions

Starting out with a basic composition hierarchy and basic discrimination graphs at the composition leaf level, we construct an abstract composition hierarchy with discrimination graphs at each level of composition. In the sections 3.2.2 and 3.2.3 the construction is subjected to a number of restricting assumptions. These are:

1. Only "must-be-part-of" links could be used for projection.
2. The discrimination graphs had to be orthogonal to the composition hierarchy.

### 3. Image features were cues for schemata at the composition leaf level only.

The first restriction is a convenient one because in geographic maps the majority of the constraints are organized in a bottom-up direction. For instance, road-systems, river-systems, and mountain-ranges are all *must-be-part-of* a geo-system, but geo-system itself has no mandatory components at all. In principle, there is no problem in projecting the discrimination graph along "must-be-parts" links as well. "May-be" links, however, have to be excluded from the process as they reintroduce hypothetical interpretations.

There are strong objections to lifting the second restriction. It would remove the orthogonality (and thus the modularity) of the composition and discrimination process. If a discrimination results in a change of composition level this necessitates additional completion. It should be observed, however, that orthogonality is not very much of a restriction. If orthogonality does not come naturally, we can always manipulate the composition hierarchy such that orthogonality is achieved.

The third restriction is not very much of a restriction either. In Mapsee-3 cues are formed at one level of detail only, and they have access to the composition leaf level only. In principle, there should be no problem with cues formed at multiple levels of detail with access to different levels of composition (See Browse (1982) for an example of such a system). The discrimination hierarchies can still be constructed level by level starting at composition level 1. With "must-be-

parts" links accounted for as well, the projection process becomes bi-directional. First we project level 1 onto level 2 using "must-be-part-of". Next we project level 2 onto level 1 using "must-be-parts", thereby avoiding class duplication. In the following stage we project level 2 onto level 3 etc. There is only one complication. We first have to construct abstract discrimination graphs at all levels of composition accessed by cues. In the restricted approach, we had to do this for the composition leaf level only. When projecting one level onto the next, we have to merge the projected discrimination graph with the one that already exists at that level.

#### 5.4.2. Relaxing the Discrimination Constraints

##### *n*-ary constraints

In HAC-3 we use unary and binary constraints only. Can HAC-3 deal with *n*-ary constraints? In Mapsee-3 constraints are represented as relations. For the algorithm, it makes no difference whether a variable has one, two, or *n* neighbors. Hence, HAC-3 has no problem dealing with *n*-ary constraints.

One way of creating higher-order relations is to have methods that create higher-order relations out of lower-order ones. Often this is computationally expensive to achieve. In some cases, however, it may be possible to create higher-order relations directly by means of image cues created at a coarse level of detail. For instance, one can think of a 3-tuple relation consisting of a bridge, the road

crossing it, and the river passing under it. A cue for such a relation could be constructed at a coarse level of detail in the image.

- *Generalizing the discrimination graph to a directed acyclic graph*

Mapsee-3 uses an exclusive OR graph. The reason for this choice lies in HAC-3. If there were more than one path from an intermediate label in the graph to a leaf, then the number of iterations necessary to reach the leaf would explode. This is caused by the use of the *P-or* predicate in HAC-3.<sup>3</sup> The only information provided by *P-or* is whether there is a successor label that is consistent. It does not tell us which path leads to this successor label. We can solve this path identification problem by compiling the knowledge about the path. We can then associate with each *P-or* a particular branch that we need to follow in order to reach the consistent successor label. With this correction, HAC-3 can efficiently operate on any directed acyclic graph.

### 5.4.3. Method Generalization

In Mapsee-3 abstract schemata have no methods of their own. By means of the method inheritance mechanisms discussed in section 3.2.6, abstract schemata have access to the methods of their descendants in the discrimination graph. It would be attractive, if abstract schemata had their own abstract methods. It

<sup>3</sup>The reader should consult Appendix A for a proper understanding of the function of this predicate.

would increase the efficiency of discrimination vision. The following example clarifies why this is the case.

A road/river-system schema can apply the "road-road-tee" method to two of its components only if both components are already interpreted as road. Similarly, the "river-river-tee" method can be applied only if both components are rivers. If both constituents of a potential T-junction are labeled *road/river* then the road/river-system schema has no power to enforce any relation. Yet under these conditions we already know that a T-junction will eventually be established because *road* and *river* are never neutral with respect to each other in a T-junction.

If the road/river-system schema had a method that could establish a road/river-road/river-tee then both constituents of the junction would become part of the same road/river-system instance immediately. Currently, in Mapsee-3, both constituents do not join together until the "geo-system" level has been reached. From there it takes many more method applications and instance merges before both junction constituents are finally joined in one road- or river-system instance. Hence, abstract methods enable us to establish relations in a much earlier stage of the interpretation process than would be the case without them.

The problem with abstract methods is, that it is hard to establish them automatically when we create abstract schemata. Method generalization has to be anticipated in advance. For the basic schemata classes of methods would have to

be defined. By investigating the matrix for each class of methods it can be established, whether or not the method can be generalized.

Returning to the example above, all T-junction methods can form a class. Both the road-system and river-system schema own a T-junction method. The sub-matrix for T-junctions in Table 3.2 for *road* and *river* interpretations only contains no "\*"s. This implies that the "road-road-tee" and "river-river-tee" method can be generalized to a "road/river-road/river-tee" method. Similarly we can infer that for a T-junction the stem of which is a *road/river/mountain* and the bar of which is a *road/river*, we cannot generalize the T-junction method because the "\*" between *mountain* and *road* shows that no T-junction can exist between these two interpretations. The problem remains that in anticipation of method generalization we have to define classes in advance.

## 5.5. Discrimination Vision in Context

### 5.5.1. Discrimination Vision and Similar Concepts

The concept that comes closest to discrimination is probably Tsotsos's concept of similarity links (e.g. Tsotsos *et al*, 1980). Both discrimination and similarity links relate classes that have similarities in their respective descriptions. Both relate classes that comprise a discriminatory set, that is, only one class can

be instantiated at any one time. The difference between the two concepts is that similarity links relate classes at the same level of specificity in an is-a hierarchy.<sup>4</sup> Discrimination graphs relate classes at the same level of composition. As well, they may include a specialization hierarchy. Visual similarity is only one of the possible properties inherited in a specialization hierarchy, and not all visually similar objects are embedded in a specialization hierarchy.

The idea of combining a specialization hierarchy with the principle of least commitment was first proposed by Marr and Nishihara (1976). An implementation of this idea by means of a constraint satisfaction algorithm was one of the features of Mapsee-2 (Havens *et al*, 1984). In Mapsee-2, however, the constraints were represented procedurally, in contrast with the declarative representation used in Mapsee-3. The idea of representing and automatically constructing unnatural classes at multiple levels of composition can be found in Mapsee-3 only.

### 5.5.2. Discrimination Vision in a general-purpose Signal Interpretation System

In Mapsee-3 we assume that the image features resulting from a segmentation process are correct. The program deals with only the fact that these features are ambiguous when regarding interpretation. By choosing line sketches we have avoided the problem of an image formation process which delivers unreliable

---

<sup>4</sup>This is a specialization hierarchy with universal implication.



features as a result of noisy data. In a general-purpose signal interpretation system such features could exist. Hence, we have to raise the question how discrimination vision would be affected if we have to work with features which are possibly incorrect or maybe even non-existent.

It is obvious that Mapsee-3 does not have the mechanisms to deal with a noisy image formation process. The issues involved in the design of a system with a knowledge base that interacts with such a noisy image formation process are very complex. Attempts to solve that problem are on the frontier of current research in computational vision.

One way to deal with potentially incorrect or non-existent features is to consider the features themselves as hypothetical. This means, however, that all the original schema invocations in the scene have to be hypothetical as well. Unless there are ways of grouping the original hypotheses we would be forced back into a hypothetical approach.

Some researchers assign certainty values to different hypotheses raised by a single feature. Such a value is based on the assumption that one feature will more commonly give rise to a certain hypothesis than another. With such an approach we can group objects suggested with equal certainty by a particular feature into a discrimination graph. We thus reduce the number of hypothetical invocations, but at the same time we maintain the advantages of discrimination vision. One result of such an approach would be that common situations are dealt with in an efficient manner. Uncommon situations would require some backtracking and

take more time to be resolved.

### 5.6. Summary

In this chapter we have discussed the performance of the discrimination vision approach as implemented in the Mapsee-3 system. Mapsee-3 is robust, but this does not imply that the discrimination vision approach is more powerful than a hypothetical approach. The main advantages of discrimination vision are conceptual clarity and efficiency. The former is reflected in the modularity and uniformity of the system, and in the separation between domain-dependent and domain-independent knowledge. The efficiency of the system has been demonstrated by means of several measures such as the number of iterations required by different algorithms and the number of instances in the interpretation graph. Additionally, the discrimination vision approach is domain-independent to the extent that it should be applicable to any signal processing domain. The compatibility between the Mapsee-3 design principles and those of the Hearsay-II system (Erman and Lesser, 1980) support this point of view.

## 6. SUMMARY AND FUTURE DIRECTIONS

### 6.1. Summary

This dissertation has addressed the problem of representing visual interpretations that are ambiguous and hypothetical. Ambiguity is caused by at least two factors: a segmentation process that has to deal with noisy data, and image primitives which are underconstrained when it comes to interpretation. We have only concerned ourselves with the latter factor.

Hypothetical and ambiguous interpretations have a close relationship. On the one hand, we can maintain a separate representation for each possible interpretation of an image primitive. Such an interpretation is hypothetical. On the other hand, we can join different possible interpretations in a discrimination graph. As a result, we can merge some interpretations into one, more abstract, interpretation. Such an interpretation is ambiguous. In most model-based vision systems, we find a mixture of hypothetical and ambiguous interpretations. The former are maintained along a composition/aggregation dimension, the latter along a specialization/generalization dimension. The representation of ambiguous and hypothetical interpretations along different knowledge representation dimensions causes problems with modularity in representation and control, and with efficiency.

A schema-based program for interpreting sketch maps, Mapsee-3, has been designed and implemented which solves these problems. Conceptual clarity has been the criterion for the design. This is reflected in

1. the modularity in representation,
2. the modularity in control,
3. uniformity of the representation,
4. strict separation between domain-dependent and domain-independent knowledge.

Most important of all, the knowledge about ambiguous and hypothetical interpretations is represented along one knowledge representation dimension: a discrimination/generalization dimension. This dimension is realized by discrimination graphs which form a hierarchical representation of object classes with similarities in visual appearance. The key idea behind this representation is the existence of an abstract object class for each possible combination of local interpretations that can arise from the image. This class enables us to represent each image primitive by means of one (abstract) object class. In Mapsee-3, an object class is represented as a schema, which consists of a list of attributes. The interpretation(s) of each image primitive are expressed by one of the schema's attributes, its label. A single label implies an ambiguous interpretation, whereas multiple labels make the interpretation hypothetical.

Discrimination graphs are reminiscent of specialization hierarchies, but they are different in at least two respects: discrimination graphs often form categorizations of object classes which are unnatural, and there is no universal implication in discrimination graphs. The presence of unnatural object classes is caused by the fact that many object classes with visual similarities cannot be joined in a (natural) specialization hierarchy. However, the unnatural constituent of a discrimination graph can be constructed automatically once its natural counterpart is known.

The Mapsee-3 knowledge base is organized along three dimensions: a composition/aggregation dimension, a discrimination/generalization dimension, and a dimension that contains the relations which connect image primitives with object classes in the scene. These dimensions are constructed orthogonally to each other. Object classes are embedded in both a composition hierarchy and a discrimination graph. Discrimination graphs are constructed for one particular level of composition only. A discrimination graph can therefore never contain object classes from different levels of composition. Furthermore, the scene domain can be accessed only through object classes at the composition leaf level. All these factors enhance modularity in representation. Finally, the knowledge representation dimensions merely provide a general format for representing knowledge about particular (scene) domains. This enhances domain-independence of the system.

The Mapsee-3 control is subdivided in three stages: segmentation, image-to-scene mapping, and interpretation. A segmentation process takes a set of plotter commands as input, and creates a set of image primitives in the form of chains and regions. An image-to-scene process provides the connection between the image and the scene domain. Each image primitive is represented by one (abstract) object class at the leaf level of the composition hierarchy in the scene domain. Interpretation is guided by two modular processes: composition and discrimination. Composition subsequently represents image primitives in terms of different object classes at different levels of composition in a bottom-up manner and discrimination ensures that each object class obtains an appropriate interpretation. A hierarchical arc consistency algorithm achieves this. The composition process can only create and alter data structures along the composition/aggregation dimension. The same is true for the discrimination process along the discrimination/generalization dimension. Thus, Mapsee-3 also achieves modularity in control.

Hierarchical arc consistency is an arc consistency algorithm that uses the principle of least commitment as an operating principle. It operates on a discrimination graph the domain of which is hierarchically organized. Hierarchical arc consistency refines the label(s) of an object class along its discrimination graph up to a level justified by the constraints found in the image. Although hierarchical arc consistency does not solve the constraint satisfaction problem, it enables us to stop the interpretation at a level of discrimination which is not a leaf level in the discrimination graph. This feature reflects a natural phenomenon in human

information processing which cannot be achieved either by arc consistency or depth-first backtrack.

Mapsee-3 is efficient for two reasons. Both result from the use of discrimination graphs which enable us to construct an interpretation graph in which competing hypothetical interpretations are represented by one variable. The interpretations themselves form the labels in the domain of the variable. In most cases the invalidation of a particular hypothesis results in the deletion or replacement of a label, not in a structural change of the interpretation graph. The latter is the case in many model-based vision systems. The other reason is the hierarchical organization of discrimination graphs. Such an organization enables us to represent the domain of each variable in hierarchical manner rather than as a set. As a result, the number of labels that has to be represented in the domain of each variable is relatively small.

The model that underlies the Mapsee-3 design interprets image primitives by means of object classes whose interpretation is at first extremely generic and unspecified. As more and more constraints are discovered in the image, this interpretation becomes more and more specific. Because of this continuing process of interpretation refinement along a discrimination graph, we call this particular approach *discrimination vision*.

## 6.2. Future Directions

### 1. Generalization to other domains.

The basic representational format of Mapsee-3 is domain-independent. It would therefore be natural to try out different domains. Line drawings of human bodies as used by Browse in his dissertation (Browse, 1982) would be a possibility. As well, a system could be designed that can interpret line sketches from more than one domain. This would increase the number of levels in the discrimination graph. However, a real test of Mapsee's domain-independence would come from an implementation in a different signal processing domain such as speech interpretation.

### 2. Top-down control strategy.

The Mapsee-3 control strategy is mainly data-driven. In particular, the composition process works along the composition hierarchy in a direction from leaf to top. The constraints in the sketch map domain are the prime motivation for this strategy. Most of the constraints are pointing upwards along the composition hierarchy. For example, mountain-ranges and road-systems are all mandatory components of a landmass, but a landmass itself has no mandatory components at all. One of the few suitable situations for implementing top-down control is the situation with a river-system and a bridge as its component. In such a situation we know there must be two rivers, both of which flow under the bridge. Line

sketches of human bodies would also be a suitable domain for experimentation with a more mixed control strategy.

### *3. Extensions towards the real world.*

This would entail abandoning the domain of line sketches and reaching in the direction of the outside world. A good intermediate solution would be to stick with the geographic world but to seek a richer input. Such a solution could be found in the interpretation of digitized geographic maps. Without having to deal with the complexity of the real world, one would be able to use edge detection and region formation techniques which reintroduce the problem of hypothetical image features. Such input material would enable us to experiment with discrimination vision in the context of a more general-purpose vision system.

### *4. Automatic construction of a generalized composition hierarchy.*

More work needs to be done in order to develop algorithms for automatic construction of an abstract composition hierarchy from a basic composition hierarchy and basic discrimination graphs at different levels of composition. In the discussion in section 5.4, we have sketched a construction solution only for a situation in which image primitives at different levels of detail are mapped into different levels of composition. Formal projection algorithms for such a situation have yet to be elaborated. Furthermore, the question of how to automatically create abstract methods is very much an open issue.

### *5. Instance hierarchies.*

In Mapsee-3, instance links connecting schemata with their instances only exist in the temporary data base constructed during interpretation. This is the case, because Mapsee-3 represents object classes only, not individuals. Mapsee-3 does not permanently store any knowledge about the particular scenes it has interpreted. A system which amalgamates each new interpretation in its permanent knowledge base such that future recognition of the same scene can be done more efficiently would constitute another worthwhile future enterprise.

### *6. Dynamic thresholding*

The problem of dynamic thresholding was raised in section 5.3.1. Mapsee-3 uses fixed thresholds for forming T-junctions. We have argued that the value of such thresholds should be dynamically controlled by the interpretation process. For instance, if we already know that a particular chain is a road then we should relax all the parameters by means of which this road can form junctions with road compatible object classes. Knowledge-driven thresholding constitutes another realm for future research.

## 6.3. Conclusion

Discrimination graphs are a better way of representing ambiguous and hypothetical interpretations in a model-based vision system than specialization hierarchies. They enhance modularity, uniformity, domain-independence, and the efficiency of the system. As well, the hierarchical arc consistency algorithm is an efficient and natural means of propagating consistency over these graphs.

SIT FINIS LIBRI

NON FINIS QUERENDI

## References

1. R.J. Anderson and G.H. Bower, *Human Associative Memory*, John Wiley, New York, 1973.
2. R. Bajcsy and L.I. Lieberman, "Computer Description of Real Outdoor Scenes," *Proc. of the 2nd Int. Joint Conf. on Pattern Recognition*, pp. 174-179, Copenhagen, 1974.
3. R. Bajcsy and A.K. Joshi, "A Partially Ordered World Model and Natural Outdoor Scenes," in *Computer Vision Systems*, ed. A.R. Hanson and E.M. Riseman, pp. 263-270, Academic Press, New York, 1978.
4. H.G. Barrow and J.M. Tenenbaum, "Recovering Intrinsic Scene Characteristics From Images," in *Computer Vision Systems*, ed. A.R. Hanson and E.M. Riseman, pp. 3-26, Academic Press, New York, 1978.
5. H.G. Barrow and J.M. Tenenbaum, "Computational Vision," *Proc. IEEE*, vol. 69, pp. 572-595, 1981.
6. F.C. Bartlett, *Remembering. A Study in Experimental and Social Psychology*, Cambridge University Press, Cambridge, 1932.
7. D.G. Bobrow and B. Raphael, "New Programming Languages for Artificial Intelligence Research," *Computing Surveys*, vol. 6, no. 3, pp. 153-174, 1974.
8. D.G. Bobrow and D.A. Norman, "Some Principles of Memory Schemata," in *Representation and Understanding: Studies in Cognitive Science*, ed. D.G. Bobrow and A. Collins, pp. 153-174, Academic Press, New York, 1975.
9. D.G. Bobrow and T. Winograd, "An Overview of KRL, A Knowledge Representation Language," *Cognitive Science*, vol. 1, pp. 3-46, 1977.
10. R.C. Bolles, L.H. Quam, M.A. Fischler, and H.C. Wolf, "Automatic Determination of Image-to-Database Correspondences," *Proc. of the 6th Int. Joint Conf. on Artificial Intelligence*, pp. 73-78, Tokyo, 1979.
11. R.J. Brachman, "On the Epistemological Status of Semantic Networks," in *Associative Networks: Representation and Use of Knowledge by Computers*, ed. N.V. Findler, pp. 3-50, Academic Press, New York, 1979.
12. R.J. Brachman, "What 'ISA' Is and Isn't," *Proc. of the 4th Nat. Conf. of the Can. Soc. for Comp. Studies of Intelligence*, pp. 212-221, Saskatoon, Canada, 1982.
13. M. Brady, "Computational Approaches to Image Understanding," *Computing Surveys*, vol. 14, pp. 3-71, 1982.
14. R.A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 285-348, 1981.
15. R.A. Brooks, *Model-based Computer Vision*, U.M.I. Press, Ann Arbor, Michigan, 1983.
16. R. Browse, "Knowledge-based Visual Interpretation Using Declarative Schemata," Ph.D. Thesis, Technical Report TN-82-12, Deptmt of Computer Science, Univ. of British Columbia, Vancouver, B.C., 1982.
17. W.F. Clocksin, "AI Theories in Vision: A Personal View," *AISB Quarterly*, vol. 31, pp. 23-37, 1978.
18. M.B. Clowes, "On Seeing Things," *Artificial Intelligence*, vol. 2, no. 1, pp. 79-112, 1971.
19. L.S. Davis and A. Rosenfeld, "Cooperating Processes for Low-level Vision: A Survey," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 245-263, 1981.

20. R. Davis and J. King, "An Overview of Production Systems," Technical Report AIM-271, Computer Science Department, Stanford University, Stanford, California, 1975.
21. L.D. Erman and V.R. Lesser, "The Hearsay-II Speech Understanding System, a Tutorial," in *Trends in Speech Recognition*, ed. W. Lea, pp. 361-381, Englewood Cliffs, New York, 1980.
22. T.G. Evans, "A Program for the Solution of Geometric-Analogy Intelligence Test Questions," in *Semantic Information Processing*, ed. M. Minsky, pp. 271-353, M.I.T. Press, Cambridge, Mass., 1968. Originally published as a Ph.D. thesis at M.I.T. in 1963.
23. S.E. Fahlman, "Representing and Using Real-World Knowledge," in *Artificial Intelligence, an M.I.T. Perspective, Volume I*, ed. P.H. Winston and R.H. Brown, pp. 453-472, M.I.T. Press, Cambridge, Mass., 1979.
24. G. Falk, "Interpretation of Line Data as a Three-Dimensional Scene," *Artificial Intelligence*, vol. 3, no. 2, pp. 101-144, 1972.
25. E.A. Feigenbaum, "The Simulation of Verbal Learning Behavior," in *Computers and Thought*, ed. E.A. Feigenbaum and J. Feldman, pp. 297-309, McGraw Hill, New York, 1963.
26. E.C. Freuder, "A Computer System for Visual Recognition Using Active Knowledge," AI-TR-345, MIT, 1976.
27. E.C. Freuder, "Synthesizing Constraint Expressions," *Communications ACM*, vol. 21, no. 11, pp. 958-966, 1978.
28. J. Glicksman, "A Cooperative Scheme for Image Understanding Using Multiple Sources of Information," Ph.D. Thesis, Technical Report TN-82-13, Deptmt of Computer Science, Univ. of British Columbia, Vancouver, B.C., 1982.
29. J. Glicksman, "Using Multiple Information Sources in A Computational Vision System," *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence*, pp. 1078-1080, Karlsruhe, W. Germany, 1983.
30. A. Grasselli, *Automatic Classification and Interpretation of Images*, Academic Press, New York, 1969.
31. A. Guzman, "Decomposition of a Visual Scene Into Three-Dimensional Bodies," *Proc. AFIPS 1968 Fall Joint Computer Conference*, pp. 291-304, 1968. also published in Grasselli (1969), pp. 243-276.
32. A.R. Hanson and E.M. Riseman, "VISIONS: A Computer System for Interpreting Scenes," in *Computer Vision Systems*, ed. A.R. Hanson and E.M. Riseman, pp. 303-334, Academic Press, New York, 1978.
33. W.S. Havens, "A Procedural Model of Recognition for Machine Perception," TR-78-3, Computer Science Dept., University of British Columbia, Vancouver, Canada, 1978.
34. W.S. Havens and A.K. Mackworth, "Representing Knowledge of the Visual World," *IEEE Computer*, vol. 16, no. 10, pp. 90-98, 1983.
35. W.S. Havens, A.K. Mackworth, and J.A. Mulder, *Forthcoming Paper on the Mapsee-2 System*, 1985.
36. P.J. Hayes, "The Logic of Frames," in *Frame Conceptions and Text Understanding*, ed. D. Metzger, pp. 46-61, Berlin, 1979.
37. D.H. Hubel and T.N. Wiesel, "Brain Mechanisms of Vision," *Scientific American*, vol. 241, no. 3, pp. 150-162, 1979.

38. D.A. Huffman, "Impossible Objects as Nonsense Sentences," in *Machine Intelligence 6*, ed. B. Meltzer and D. Michie, pp. 295-323, American Elsevier, New York, 1971.
39. R.A. Hummel and S.W. Zucker, "On The Foundations of Relaxation Labeling Processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 5, no. 3, pp. 267-287, 1983.
40. B.J. Kuipers, "A Frame for Frames," in *Representation and Understanding*, ed. D.G. Bobrow and A. Collins, pp. 151-184, Academic Press, New York, 1975.
41. V.R. Lesser and L.D. Erman, "A Retrospective View of the HEARSAY-II Architecture," *Proc. 5th Int. Joint Conf. on Artificial Intelligence*, pp. 790-800, M.I.T., Cambridge, Mass., 1977.
42. V.R. Lesser and L.D. Erman, "HEARSAY-II: Tutorial Introduction and Retrospective View," Technical Report CMU-CS-78-117, Deptmt of Computer Science, Carnegie Mellon University, 1978.
43. H.J. Levesque and J. Mylopoulos, "A Procedural Semantics for Semantic Networks," in *Associative Networks: Representation and Use of Knowledge by Computers*, ed. N.V. Findler, pp. 93-120, Academic Press, New York, 1979.
44. A.K. Mackworth, "Model-Driven Interpretation in Intelligent Vision Systems," *Perception*, vol. 5, pp. 349-370, 1977a.
45. A.K. Mackworth, "On Reading Sketch Maps," *Proceeding of the Fifth International Joint Conference on Artificial Intelligence*, pp. 598-606, Cambridge, 1977b.
46. A.K. Mackworth, "Consistency in Networks of Relations," *Artificial Intelligence*, vol. 8, no. 1, pp. 99-118, 1977c.
47. A.K. Mackworth and W.S. Havens, "Structuring Domain Knowledge for Visual Perception," *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence*, pp. 625-627, Vancouver, B.C., 1981.
48. A.K. Mackworth and E.C. Freuder, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems," Technical Report TR-82-6, Deptmt of Computer Science, Univ. of British Columbia, Vancouver, B.C., 1982.
49. A.K. Mackworth and F. Mokhtarian, "Scale-based Descriptions of Planar Curves," *Proc. of the 5th Nat. Conf. of the Can. Soc. for Comp. Studies of Intelligence (CSCSI)*, pp. 114-118, London, Ont., 1984.
50. A.K. Mackworth, J.A. Mulder, and W.S. Havens, "Hierarchical Arc Consistency: Exploiting Structured Domains in Constraint Satisfaction Problems," Technical Report 85-7, Department of Computer Science, University of British Columbia, June, 1985.
51. D. Marr and H.K. Nishihara, "Representation of the Spatial Organization of Three Dimensional Shapes," Report 377, A.I. Lab, M.I.T., 1978.
52. D. Marr, "Representing Visual Information," in *Computer Vision Systems*, ed. A.R. Hanson and E.M. Riseman, pp. 61-80, Academic Press, New York, 1978.
53. D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman, San Francisco, 1982.
54. W.A. Martin, "Descriptions and the Specialization of Concept," in *Artificial Intelligence: an M.I.T. Perspective, Volume I*, ed. P.H. Winston and R.H. Brown, pp. 377-420, M.I.T. Press, Cambridge, Mass., 1979.



55. J. McCarthy and P. Hayes, "Some Philosophical Problems From the Standpoint of Artificial Intelligence," in *Machine Intelligence 4*, ed. B. Meltzer and D. Michie, American Elsevier, New York, 1969.
56. M. Minsky, *Semantic Information Processing*, M.I.T. Press, Cambridge, Mass., 1968.
57. M. Minsky, "A Framework for Representing Knowledge," in *The Psychology of Computer Vision*, ed. P.H. Winston, pp. 211-277, McGraw-Hill, New York, 1975.
58. J.A. Mulder, "Representation and Control in a Program That Understands Line Sketches of Houses," M.Sc. thesis, Computer Science, University of British Columbia, 1979.
59. M. Nagao, T. Matsuyama, and Y. Ikeda, "Region Extraction and Shape Analysis of Aerial Photographs," *Proc. of the 4th Int. Joint Conf. on Pattern Recognition*, pp. 620-628, Kyoto, 1978.
60. M. Nagao, T. Matsuyama, and H. Mori, "Structural Analysis of Complex Aerial Photographs," *Proc. Sixth International Joint Conference on Artificial Intelligence*, pp. 610-616, Tokyo, 1979.
61. U. Neisser, *Cognition and Reality*, Freeman, San Francisco, 1976.
62. A. Newell and H.A. Simon, "GPS, A Program That Simulates Human Thought," in *Computers and Thought*, ed. E. Feigenbaum and J. Feldman, pp. 279-296, McGraw Hill, New York, 1963.
63. A. Newell, "Production Systems: Models of Control Structure," in *Visual Information Processing*, ed. W.G. Chase, pp. 463-526, Academic Press, New York, 1973.
64. D.A. Norman and D.E. Rumelhart, *Explorations in Cognition*, Freeman and Company, San Francisco, 1975.
65. D.A. Norman and D.G. Bobrow, "On The Role of Active Memory Processes in Perception and Cognition," in *The Structure of Human Memory*, ed. C.N. Cofer, pp. 114-132, Freeman and Comp., San Francisco, 1976.
66. J. Piaget, *Biology and Knowledge*, Gallimard Press, Paris, 1967.
67. M.R. Quillian, "Semantic Memory," in *Semantic Information Processing*, ed. M. Minsky, pp. 227-270, M.I.T. Press, Cambridge, Mass., 1968. Condensed from his Ph.D. thesis, Carnegie-Mellon University, Oct 1976, published as Report AFCRL-66-189
68. M.R. Quillian, "The Teachable Language Comprehender: a Simulation Program and Theory of Language," *Communications of the Acm*, vol. 12, no. 8, pp. 459-476, 1969.
69. B. Raphael, "Sir: a Computer Program for Semantic Information Retrieval," Thesis, Massachusetts Institute of Technology, 1964. Also Reprinted in Minsky (1968), p.p. 33-134
70. R. Reiter, "A Logic for Default Reasoning," *Artificial Intelligence*, vol. 13, pp. 81-132, 1980.
71. C. Rieger and M. Grinberg, "The Declarative Representation and Procedural Simulation of Causality in Physical Mechanics," *Proc. of the 5th Int. Joint Conf. on Artificial Intelligence*, pp. 250-256, Cambridge, Mass., 1977.
72. L.G. Roberts, "Machine Perception of Three Dimensional Solids," in *Optical and Electro-optical Information Processing*, ed. J.T. Tippett, D. Berkowitz, L. Clapp, C. Koester, and A. Vanderburgh, pp. 159-197, MIT

- Press, Cambridge, Mass., 1965.
73. R.B. Roberts and I.P. Goldstein, "The FRL Manual," Memo 409, Artificial Intelligence, Massachusetts Institute of Technology, 1977a.
74. R.B. Roberts and I.P. Goldstein, "The FRL Primer," Memo 408, Artificial Intelligence, Massachusetts Institute of Technology, 1977b.
75. A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Transactions on Systems, Man, and Cybernetics (SMC)*, vol. 6, no. 6, pp. 420-433, 1976.
76. D. Rosenthal and R. Bajcsy, "Conceptual and Visual Focussing in the Recognition Process as Induced by Queries," *Proc. of the 4th Int. Joint Conf. on Pattern Recognition*, pp. 417-420, Kyoto, 1978.
77. A.L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," in *Computers and Thought*, ed. E.A. Feigenbaum and J. Feldman, pp. 71-108, McGraw Hill, New York, 1963.
78. L.K. Schubert, "Extending the Expressive Power of Semantic Networks," *Artificial Intelligence*, vol. 7, no. 2, pp. 163-198, 1976.
79. L.K. Schubert, "Problems with Parts," *Proc. of 6th Int. Joint Conf. on Artificial Intelligence*, pp. 778-784, Tokyo, 1979.
80. T. Shibahara, J. Mylopoulos, J.K. Tsotsos, and H.D. Covey, "CAA: A Knowledge-based System with Causal Knowledge to Diagnose Rhythm Disorders in the Heart," *Proc. of the 4th Nat. Conf. of the Can. Soc. for Comp. Studies of Intelligence*, pp. 71-78, Saskatoon, 1982.
81. T. Shibahara, J.K. Tsotsos, and H.D. Covey, "CAA: A Knowledge-based System Using Causal Knowledge to Diagnose Cardiac Rhythm Disorders," *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence*, pp. 242-245, Karlsruhe, W. Germany, 1983.
82. H.A. Simon and E.A. Feigenbaum, "An Information Processing Theory of Some Effects of Similarity, Familiarization, and Meaningfulness in Verbal Learning," *Journal Of Verbal Learning and Verbal Behavior*, vol. 3, pp. 385-396, 1964.
83. R.G. Smith and P. Friedland, "UNIT Package User's Guide," Technical Memorandum 80/L, DREA, 1980.
84. M.J. Stefik, "An Examination of Frame-Structured Representation Systems," *Proc. of the 6th Int. Joint Conf. on Artificial Intelligence*, pp. 845-852, Tokyo, 1979.
85. J.M. Tenenbaum, M.A. Fischler, and H.C. Wolf, "A Scene-Analysis Approach to Remote Sensing," TN 173, SRI, 1978.
86. J.K. Tsotsos, J. Mylopoulos, H.D. Covey, and S.W. Zucker, "A Framework for Visual Motion Understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 2, no. 6, pp. 563-573, 1980.
87. J.K. Tsotsos, "Representational Axes and Temporal Cooperative Processes," RCBV-TR84-2, University of Toronto, Toronto, 1984.
88. D.L. Waltz, "Generating Semantic Descriptions From Drawings of Scenes with Shadows," Technical Note TR-271, AI Lab, M.I.T., 1972.
89. T.E. Weymouth, "Experiments in Knowledge-Driven Interpretation of Natural Scenes," *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence*, pp. 628-630, Vancouver, B.C., 1981.
90. T.E. Weymouth, J.S. Griffith, A.R. Hanson, and E.M. Riseman, "Rule-based Strategies for Image Interpretation," *Proc. of the 3rd Conf. of the Am.*



- Ass. for Artificial Intelligence*, pp. 429-432, Washington, D.C., 1983.
91. T. Winograd, "Frame Representations and the Declarative/Procedural Controversy," in *Representation and Understanding : Studies in Cognitive Science*, ed. D.G. Bobrow and A. Collins, pp. 185-210, Academic Press, New York, 1975.
  92. P.H. Winston, "Learning Structural Descriptions from Examples," in *The Psychology of Computer Vision*, ed. P.H. Winston, pp. 157-210, McGraw Hill, New York, 1975.
  93. R.J. Woodham, "Analyzing Images of Curved Surfaces," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 117-140, 1981.
  94. W.A. Woods, "What's in a Link : Foundations for Semantic Networks," in *Representation and Understanding : Studies in Cognitive Science*, ed. D.G. Bobrow and A. Collins, pp. 35-82, Academic Press, New York, 1975.
  95. S.W. Zucker, R.A. Hummel, and A. Rosenfeld, "An Application of Relaxation Labeling to Line and Curve Enhancement," *IEEE Transactions on Computers (C)*, vol. 26, no. 4, pp. 394-402, 1977.
  96. S.W. Zucker, "Vertical and Horizontal Processes in Low Level Vision," in *Computer Vision Systems*, ed. A. Hanson and E.M. Riseman, pp. 187-195, Academic Press, New York, 1978.

## APPENDIX A

## HIERARCHICAL ARC CONSISTENCY

This appendix provides a formal description of the algorithms AC-3 and HAC-3 referred to in section 3.3.2.1.

## AC-3

Restricting ourselves to unary and binary constraints, we represent the constraint satisfaction problem as a graph  $G$  in which the nodes are variables, and the links are constraints on these variables. Each variable  $V_i$  has a domain  $D_i$  of labels. Different forms of consistency can be defined over the graph. The particular forms we describe are node consistency and arc consistency.

Node consistency for node  $V_i$  in graph  $G$  can be defined as:

$$(x) P_i(x)$$

where  $x$  is a label in  $D_i$  and  $P_i$  is a unary predicate on  $V_i$ .

$G$  is node consistent if all nodes in  $G$  are node consistent. The following procedure tests node consistency:

```

procedure NC(i)
   $D_i \leftarrow D_i \cap x \mid P_i(x)$ 
1 BEGIN
2 FOR  $i \leftarrow 1$  UNTIL  $n$  DO NC(i)
3 END

```

An arc  $(i,j)$  is consistent, if:

- 1) its source node  $V_i$  is node consistent
- 2) its goal node  $V_j$  is node consistent
- 3) each  $x \in D_i$  is consistent with at least one  $y \in D_j$ .

A directed graph  $G$  is arc consistent if all of its arcs are arc consistent.

AC-3 is an iterative procedure where each iteration consists of an updating of the labels of a particular variable  $V$  under the constraints of a particular predicate. If one or more labels of  $V$  are deleted during an iteration then all the variables that constrain  $V$  are considered next. The labels of those variables are tested under the constraints defined by the predicate that relates them to  $V$ .

The basic action in arc consistency is to remove any  $x \in D_i$  for which rule 3 does not hold. This action is embodied in the Boolean procedure REVISE.

```

procedure REVISE(i,j)
1 BEGIN
2 DELETE  $\leftarrow$  false
3 for each  $x \in D_i$  DO
4   IF there is no  $y \in D_j$  such that  $P_{ij}(x,y)$  THEN
5     BEGIN
6       delete  $x$  from  $D_i$ 
7       DELETE  $\leftarrow$  true
8     END
9 END
10 return DELETE
11 END

```

The effect of any deletion has to be propagated to all nodes whose values could be affected by the deletion. This is done in the following procedure called AC-3 in Mackworth (1977c):

Procedure AC-3

```

1 BEGIN
2 FOR  $i \leftarrow 1$  UNTIL  $n$  DO NC(i)
3  $Q \leftarrow (i,j) \mid (i,j) \in \text{arcs}(G), i \neq j$ 
4
5 REPEAT UNTIL  $Q$  is empty
6   BEGIN
7     select and delete any arc  $(k,m)$  from  $Q$ 
8     if REVISE( $k,m$ ) then
9        $Q \leftarrow Q \cup ((i,k) \mid (i,k) \in \text{arcs}(G), i \neq k, i \neq m)$ 
10    END
11  END
12 END

```

## HAC-3

In AC-3 the domain of a variable is organized as a set of labels. Each instantiation of a variable has this set or a subset of these labels in its domain. In HAC-3 the domain  $D$  is organized in a hierarchical form. Each node in this hierarchy stands for a label that is unique in  $G$ . The labels at the leaves of the hierarchy are the same (basic) labels that are represented in the variable domain in AC-3. The source node of the hierarchy intensionally represents the complete set of labels at the leaves of the hierarchy. Each intermediate node represents a subset of this set. We refer to a leaf label in the hierarchy as a *basic* label. An intermediate label will be called an *abstract* label. In HAC-3 the domain of each variable still contains one or more labels. The difference with AC-3 is that in HAC-3 any label can be either an abstract or basic label whereas in AC-3 each label must be a basic label.

In AC-3 the constraints between two variables  $V_i$  and  $V_j$  with domains  $D_i$  and  $D_j$  are represented in the predicate  $P_{ij}$ . Such a representation takes the form of a truth table. Table A1 is an example of such a table. In the example,  $V_1$  is a geo-system surrounded by a shore ( $V_2$ ).  $V_1$  has four possible labels,  $V_2$  has two.  $P_{12}$  represents the "outer-shore" constraint; that is,  $V_2$  surrounds  $V_1$ . A geo-system can only be surrounded by a shore if the geo-system is an island and the shore a coastline or when the geo-system is a lake and the shore a lakeshore. In order to make the arc from  $V_1$  to  $V_2$  consistent, AC-3 would eliminate the labels mainland and ocean from the domain of  $V_1$  in this example.

An abstract label pair  $(m, n)$  is hierarchically arc consistent if the set of basic labels descending from  $m$  is arc consistent with the set of basic labels descending from  $n$ . An arc  $(i, j)$  is hierarchically arc consistent if each label in  $D_i$  is hierarchically arc consistent with at least one label in  $D_j$ .

In HAC-3 we compile  $P_{ij}$  into two new predicates:  $P\text{-and}_{ij}$  and  $P\text{-or}_{ij}$ . For any label pair  $(m, n) \mid (m \in D_i, n \in D_j)$   $P\text{-and}_{ij}$  expresses whether or not the pair is hierarchically arc consistent. For any label pair  $(m, n) \mid m \in D_i, n \in D_j$   $P\text{-or}_{ij}$  expresses whether or not *at least one* of  $m$ 's descendants is hierarchically arc consistent with  $n$ .

With  $P_{ij}$  given, we can automatically construct  $P\text{-and}_{ij}$  and  $P\text{-or}_{ij}$  for each pair of labels, irrespective of their location in the hierarchy. An example will show how this is done. For this purpose we have expanded the labels in Table A1 into a hierarchy. Figure A1 shows the result. The constructed  $P\text{-and}$ 's and  $P\text{-or}$ 's are shown in Table A2. For illustrative purposes we have expanded the predicates into a 4-tuple (e.g.  $P\text{-and}_{ijkl}$ ). The indices  $k$  and  $l$  indicate the level in the hierarchy whereby  $k = \text{level number in } D_i$ , and  $l = \text{level number in } D_j$ .  $P\text{-and}_{ij11}$  is identical to  $P_{ij}$  in Table A1. This is the basic table. The other tables are constructed by properly *AND*ing and *OR*ing together different values from the basic table.

$P_{i,j}$	Lakeshore	Coastline
Island	0	1
Mainland	0	0
Lake	1	0
Ocean	0	0

Table A1: A truth table representing the constraints between two variables in AC-3

$P\text{-and}_{i,j,1,1}$	Lakeshore	Coastline
Island	0	1
Mainland	0	0
Lake	1	0
Ocean	0	0

$P\text{-and}_{i,j,2,1}$	Lakeshore	Coastline
Landmass	0	0
Waterbody	0	0

$P\text{-or}_{i,j,2,1}$	Lakeshore	Coastline
Landmass	0	1
Waterbody	1	0

$P\text{-and}_{i,j,2,2}$	Shore
Landmass	0
Waterbody	0

$P\text{-or}_{i,j,2,2}$	Shore
Landmass	1
Waterbody	1

$P\text{-and}_{i,j,3,1}$	Lakeshore	Coastline
Geo-system	0	0

$P\text{-or}_{i,j,3,1}$	Lakeshore	Coastline
Geo-system	1	1

$P\text{-and}_{i,j,3,2}$	Shore
Geo-system	0

$P\text{-or}_{i,j,3,2}$	Shore
Geo-system	1

Table A2: P-and and P-or truth tables for the hierarchies in Figure A1

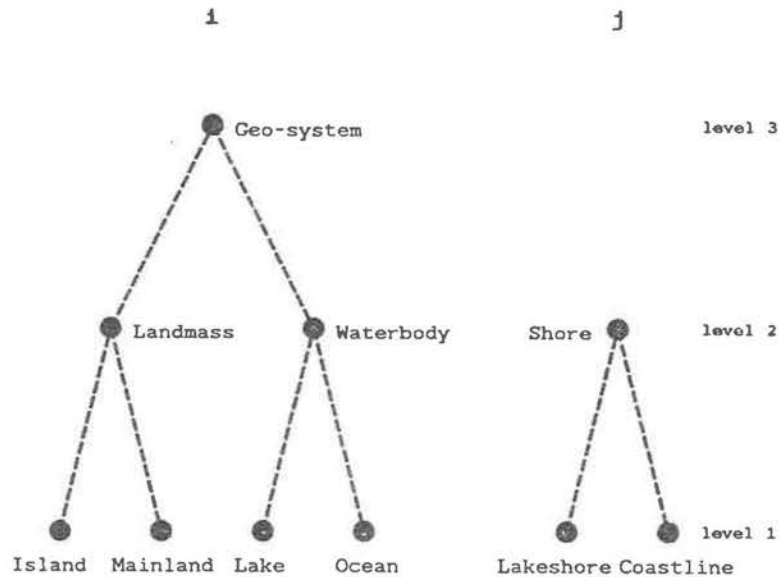


Figure A1: The labels of two variables expanded into a discrimination hierarchy

We can now discuss the HAC-3 algorithm. It is similar to AC-3 except that the procedure *REVISE* is different. A label pair  $\{(m, n) \mid m \in D_i, n \in D_j\}$  is consistent if  $P\text{-and}_{mn}$  is true. If this is not the case then we test  $P\text{-or}_{mn}$ . If this is false as well then we have to delete  $m$  from  $D_i$ . If  $P\text{-or}_{mn}$  is true, however, then we replace  $m$  by its successors in the hierarchy and we repeat the sequence of tests for each of the successors. We continue this testing until we have found one or more successors  $k$  for which  $P\text{-and}_{kn}$  is true. We call this procedure *HACREVISE*.

Procedure *HACREVISE*( $i, j$ )

```

1 BEGIN
2   DELETE ← false
3   Q1 ← Di
4   ND ← empty
5   WHILE Q1 non-empty DO
6     select and delete any element x from Q1
7     Q2 ← Dj
8     MATCH ← false
9     WHILE Q2 non-empty and not MATCH DO
10      select and delete any element y from Q2
11      IF P-andij(x,y) THEN
12        BEGIN
13          append x to ND
14          MATCH ← true
15        END
16      END
17    END
18  END
19  IF not MATCH THEN
20    BEGIN
21      Q2 ← Dj
22      WHILE Q2 non-empty and not MATCH DO
23        select and delete any element y from Q2
24        IF P-orij(x,y) THEN
25          BEGIN
26            append all successors of x to Q1
27            MATCH ← true
28          END
29        END
30      END
31    END
32    DELETE ← true
33  END
34 END
35
36
37 Di ← ND
38 return DELETE
39
40 END
  
```

It is preferable to organize the domain of IIAC-3 as an OR graph. This will prevent the existence of more than one path between two nodes when traveling in a discrimination direction. Multiple paths can cause an explosion in search, because *P or* requires us to follow each path until a consistent label is found.

*APPENDIX B*  
*AN EXAMPLE*

In this appendix an in-depth discussion of the interpretation process for the example in Figure 4.14 is provided. The interpretation scheme is shown in Figure 4.13. Table 4.7 shows the name abbreviations used for the object classes and relations. Interpretation takes place in three stages: segmentation, image-to-scene mapping, and interpretation. The segmentation process segments the image into primitives and cues. The segmentation of Figure 4.14 results in the creation of one region and four chains. The image-to-scene process places each one of the chains into a shape category and invokes the abstract schema depicted by this shape category. Four schema instances are created at the composition leaf level: *Rd/Rv/Brs -1* for chain-1, *Rd/Rv/Brs -2* for chain-2, *Rd/Rv -1* for chain-3, and *Rd/Rv -2* for chain-4 (Figure 4.15). The name inside the brackets is the label of the instance. All instances are inserted in this order in a completion queue.

The interpretation process consists of two processes: composition and discrimination. Composition consists of two steps: completion and assembly. Discrimination is equivalent to hierarchical arc consistency, implemented as IIAC-3. Both the composition and discrimination processes take their input from a queue. Composition has a completion queue, discrimination a label consistency queue, abbreviated as lc-queue. The composition and discrimination process communicate with each other by pushing elements on each other's queue. Figure 4.13

shows how the processes alternate. The interpretation process is initiated by the appearance of schema instances on the completion queue.

Completion takes place in a depth-first or breadth-first manner, depending on whether we treat the completion queue as a stack, or whether we order instances by composition level. The first element on the queue is always picked first. In this example we will follow a breadth-first strategy. The interpreter in Appendix E also follows a breadth-first strategy.

completion-queue:  $Rd/Rv/Brs-1, Rd/Rv/Brs-2, Rd/Rv-1, Rd/Rv-2$ .

$Rd/Rv/Brs-1$  is selected for completion and is deleted from the queue. It completes to  $Rd/Rv/Br-1$ .<sup>1</sup> A matching instance is therefore looked for among the instances in the superdiscrimination set of *road/river/bridge*. As none is found, the default rule takes effect. That is, a new instance  $Rd/Rv/Br-1$  is created.

This is the first step in the scheme shown in Figure 4.13.  $Rd/Rv/Brs-1$  is not yet linked to  $Rd/Rv/Br-1$ . First, we have to test the consistency of the arc which will connect the two instances. The arc  $(Rd/Rv/Brs-1 . Rd/Rv/Br-1)$  is pushed on the lc-queue and HAC-3 is invoked.<sup>2</sup> This is the second step shown in Figure 4.13. HAC-3 takes elements from the lc-queue and pushes elements on

<sup>1</sup>There is a level of composition relations between every two levels of composition. Completion always involves the creation of "part-of" and a "parts" relations linking two instances at adjacent composition levels. For reasons of simplicity, however, we will act as if one composition object level connects with the next one up directly.

<sup>2</sup>See Appendix A for the exact algorithm.

this queue if label refinements take place. HAC-3 continues to operate until the lc-queue is empty. In the current case there is only one element on the queue.  $P\text{-and}(Rd/Rv/Brs-1, Rd/Rv/Br-1)$  is true. Thus, consistency has been established.

Step 3 in the interpretation scheme (Figure 4.13) is assembly. This results in the linkage of  $Rd/Rv/Brs-1$  and  $Rd/Rv/Br-1$ .  $Rd/Rv/Br-1$  is inserted into the completion queue. Instances are ordered by composition level. Thus  $Rd/Rv/Br-1$  is inserted at the end of the queue.

Up to now consistency has been tested in one direction only (from  $Rd/Rv/Brs-1$  to  $Rd/Rv/Br-1$ ). During the second HAC test (step 4 in Figure 4.13) we push the arc  $(Rd/Rv/Br-1 . Rd/Rv/Brs-1)$  on the lc-queue. As is the case in the previous test, there is immediate consistency. No label changes have taken place so far, but we have now completed a cycle in the interpretation scheme as illustrated in Figure 4.13.

completion-queue:  $Rd/Rv/Brs-2, Rd/Rv-1, Rd/Rv-2, Rd/Rv/Br-1$ .

$Rd/Rv/Brs-2$  is next for completion. A matching instance is looked for once more in the superdiscrimination set of *road/river/bridge*.  $Rd/Rv/Br-1$  is found and tried. *Road/river/bridge* has access to the "bridgesidep/bridgesidep" method. This schema successfully applies this method to  $Rd/Rv/Brs-1$  & 2. As a result, an instance of the *bridgesidep/bridgesidep* relation is created which will eventually link  $Rd/Rv/Brs-1$  & 2. The arcs  $(Rd/Rv/Brs-2 . Rd/Rv/Br-1)$ ,

$(Rd/Rv/Brs-1 . Rd/Rv/Brs-2)$ , and  $(Rd/Rv/Brs-2 . Rd/Rv/Brs-1)$  are now pushed on to lc-queue. The first arc does not cause any problem, the second one does. The *bridgesidep/bridgesidep* relation cannot exist between two road/river/bridge sides.  $P(Rd/Rv/Brs, Brs/Brs)$  is false. HAC searches the discrimination graph to find a consistent discrimination.  $Rd/Rv/Brs$  is specialized to *bridge-side*. The same occurs for  $Rd/Rv/Brs-2$  when the third arc is tested.

As label changes take place, HAC-3 pushes all the schemata that have arcs pointing at  $Rd/Rv/Brs-1$  & 2 onto the lc-queue. As a result, the arc  $(Rd/Rv/Br-1 . Rd/Rv/Br-1)$  is pushed onto the lc-queue. During the next HAC invocation, the label of  $Rd/Rv/Br-1$  is refined to *bridge*, because only  $P\text{-and}(\textit{bridge}, \textit{bridgeside})$  is true. As a result of all the label changes  $Rd/Rv/Brs-1$  and  $Rd/Rv/Br-1$  are inserted back into the completion queue. However, the latter is already on the queue.

Next, assembly not only establishes the link between  $Rd/Rv/Brs-2$  and  $Rd/Rv/Br-1$ , but the spatial relations between  $Rd/Rv/Brs-1$  & 2 as well. During the second HAC test we push the arc  $(Rd/Rv/Br-1 . Rd/Rv/Brs-2)$  on the lc-queue, but this has no further effect.

completion-queue:  $Rd/Rv/Brs-1, Rd/Rv-1$  & 2,  $Rd/Rv/Br-1$ .

$Rd/Rv/Brs-1$  has already been completed once. It is our first case of what is called post-completion. The reason for post-completion is the label refinement

of  $Rd/Rv/Brs-1$  to *bridge-side* which has just taken place. As a result of this refinement methods may now apply which did not apply before. Application of new methods may result in the establishment of new relations, further label refinements, or the merge of one or more instances in  $Rd/Rv/Brs-1$ 's super-discrimination set. The latter may happen because  $Rd/Rv/Brs-1$  may now complete to instances other than  $Rd/Rv/Br-1$ . As  $Rd/Rv/Brs$  has only one super-schema, this implies that the super-instances must be one and the same. Hence, they must be merged.

Nothing of this kind happens in this case.  $Rd/Rv/Brs-1$ , because of its label, now completes to *bridge*. We have to search *bridge*'s superdiscrimination set for a matching instance, other than  $Rd/Rv/Br-1$ , but none is found.

completion-queue:  $Rd/Rv-1$  & 2,  $Rd/Rv/Br-1$ .

$Rd/Rv-1$  is next. It will complete to the newly created  $Rd/Rv*-1$ . It will still take some time, before the *river-under-bridge* relation involving chain-3 is found. "River-under-bridge" is a regular method and we need the help of a negative relation to constrain chain-3 to be a river. Actually, this does not happen until we reach the *landmass* schema at level 4.

completion-queue:  $Rd/Rv-2, Rd/Rv*-1, Rd/Rv/Br-1$ .

The completion of  $Rd/Rv-2$  proceeds in exactly the same manner as the completion of  $Rd/Rv-1$ . Thus,  $Rd/Rv-2$  becomes a component of the newly



created instance  $Rd/Rv^*-2$ . The current situation is shown in Figure 4.16.

completion-queue:  $Rd/Rv^*-2$  & 1,  $Rd/Rv/Br-1$ .

All level 1 instances have now been completed and we continue with level 2. We can discuss the completion of both  $Rd/Rv^*-2$  & 1 at the same time because the processes are similar.  $Rd/Rv^*-2$  becomes a component of a newly created instance  $Rd/Rvsys-1$ ,  $Rd/Rv^*-1$  becomes a component of  $Rd/Rvsys-2$ . At this point we have no way of knowing that  $Rd/Rvsys-1$  & 2 should be one and the same instance.

completion-queue:  $Rd/Rv/Br-1$ ,  $Rd/Rvsys-2$  & 1.

The completion of  $Rd/Rv/Br-1$  (with label *bridge*) is the next level 2 completion. The completion path of an instance, however, is determined by the instance's label, not by the parent schema. As a result,  $Rd/Rv/Br-1$  now completes as if it were an instance of *bridge*. Its two super-schemata are *road-system* and *river-system*.

Let us assume that  $Rd/Rv/Br-1$  completes to *road-system* first. It will try to match any of the instances in *road-system*'s superdiscrimination set. There are two instances in this set:  $Rd/Rvsys-2$  & 1. Neither can successfully apply any method. Hence, a new instance  $Rdsys-1$  is created. During the following assembly  $Rd/Rv/Br-1$  is linked with  $Rdsys-1$ . The second HAC invocation does not result in any label changes.

completion-queue:  $Rdsys-1$ ,  $Rd/Rvsys-2$  & 1.

$Rd/Rv/Br-1$  still has to complete for a second time, to *river-system*. As we explained before, the *road/river-system* schema has no access to the "river-under-bridge" method, because it is a regular method. This means that this method only applies to two instances, one with the label *bridge-side*, the other with the label *river*. It does not apply to any generalizations of *river* (e.g.  $Rd/Rv$ ).  $Rd/Rv/Br-1$  therefore completes to the newly created instance  $Rvsys-1$ . The current situation is shown in Figure 4.17.

completion-queue:  $Rvsys-1$ ,  $Rdsys-1$ ,  $Rd/Rvsys-2$  & 1.

We can now start completion from level 3 to level 4.  $Rvsys-1$  completes to *landmass*. As there are no instances created at level 4 yet,  $Rvsys-1$  will become a component of the newly created instance  $Lm-1$ .

completion-queue:  $Rdsys-1$ ,  $Rd/Rvsys-2$  & 1,  $Lm-1$ .

$Rdsys-1$  (with label *road-system*) completes to  $Lm-1$  as well, because both  $Rdsys-1$  and  $Rvsys-1$  share the same component, and thus the same chains and region.

completion-queue:  $Rd/Rvsys-2$  & 1,  $Lm-1$ .

$Rd/Rvsys-2$  will also complete to  $Lm-1$ , because the region surrounding chain-3 overlaps with the region depicting  $Lm-1$ . This is checked by the method "surface-overlap". This part of the completion has no effect on the labels of any of the instances involved. The *landmass* schema, however, has internal methods it can apply to its components. Of particular interest in this case is the very powerful "not-roadp" method. The *not-roadp* relation is imposed on any component of the *landmass* which represents a chain whose end point comes close to a river or bridge. The result of this operation is that the *not-roadp* relation is imposed on  $Rd/Rv*-1$ .

The label *road/river\** cannot coexist with *not-roadp*. During the next HAC invocation a chain of specializations will therefore take place:  $Rd/Rv*-1$  to *river\**,  $Rd/Rv-1$  to *river*, and  $Rd/Rvsys-2$  to *river-system*. All three instances are inserted into the completion queue for post-completion. After assembly and a second HAC invocation  $Rd/Rvsys-2$  is linked with  $Lm-1$ .

completion-queue:  $Rd/Rv-1$ ,  $Rd/Rv*-1$ ,  $Rd/Rvsys-1$ ,  $Lm-1$ .

Post-completion of  $Rd/Rv-1$  has no effects. However, when we post-complete  $Rd/Rv*-1$ , we are faced with the situation that  $Rd/Rv-1$  now has the label *river*, and  $Rd/Rv/Br-1$  has the label *bridge*. When  $Rd/Rv*-1$  attempts to match  $Rvsys-1$ , the "river-under-bridge" method is applied successfully. Although the labeling is stable,  $Rd/Rv*-1$  is not allowed to be part of two super-instances. These two super-instances ( $Rd/Rvsys-2$  and  $Rvsys-1$ ) will

therefore merge into one newly created super-instance, which will be a structurally modified  $Rvsys-1$ .

As a result of this structural change, we have to insert the "new"  $Rvsys-1$ , and, recursively, all of its super-components into the completion queue. This recursion is necessary because some "high level" schemata may have methods that apply far down in the hierarchy (e.g. "not-roadp"). Figure 4.18 shows the current situation.

completion-queue:  $Rvsys-1$ ,  $Rd/Rvsys-1$ ,  $Lm-1$ .

Post-completion of  $Rvsys-1$  with its two components has no further effects. The completion of  $Rd/Rvsys-1$  proceeds in a way totally symmetric to the completion of  $Rd/Rvsys-2$ . "Not-roadp" will cause  $Rd/Rv*-2$  to specialize to *river\**, and, finally,  $Rd/Rvsys-1$  will also merge with  $Rvsys-1$ , leaving the latter with three components: two rivers and a bridge. At this point we will bypass all the intermediate states of the completion queue, and we will start with the completion at level 4.

completion-queue:  $Lm-1$ .

$Lm-1$  completes to a newly created instance of *world*,  $Wrld-1$ , and this constitutes the end of the first interpretation cycle, in which all chains have been depicted up to the world level.

The second cycle is rather trivial for this particular example. The "completed-system" methods are now deblocked. As a result, particular checks are made. The "incomplete-river-system" method, for instance, will check for each river-system, or one of its generalizations, whether their components can actually be rivers, or bridges. A chain can depict a river, only if one of the following conditions is satisfied:

1. It makes a T-junction with another chain which can be labeled as a river.
2. It makes a T-junction with a chain labeled as a bridge-side.
3. It makes a T-junction with a chain labeled as mountain.
4. It makes a T-junction with a chain labeled as shore.
5. It runs of the edge on one or both sides.

If none of these conditions is satisfied, a *not-river* relation is imposed on the instance concerned. During the next invocation of HAC the label of the instance is refined. This example shows the rules for *river-system*. Compatible rules exist for road-systems and bridges.

The second cycle runs in exactly the same way as the first. Label changes cause the completion process to be reinvoked which, in turn, reinvokes HAC *etc.* Figure 4.19 shows the final interpretation graph.

## APPENDIX C

### *The Mapsee-3 Projection Algorithm*

The only reason that the Mapsee-3 projection algorithm deviates from the general projection algorithm discussed in section 3.2.5 is that this algorithm is inefficient in many-to-one mapping situations. In Mapsee-3 there exists a composition relation level between two object class levels. A projection from object class level  $l$  to object class level  $l+1$  must therefore be done in two stages. First we project object class level  $l$  onto composition relation level  $l+0.5$ . Next we project composition relation level  $l+0.5$  onto object class level  $l+1$ . In the first stage there are only two kinds of mapping situations: one-to-one, or one-to-many. For both cases the general projection algorithm works fine. In the second stage there are also two kinds of mapping: one-to-one, or many-to-one. For this reason we use a different algorithm. This algorithm does efficient many-to-one mapping. It also takes advantage of the fact that no one-to-many mapping can occur.

The Mapsee-3 projection algorithm has three stages. In the first stage we subdivide the discrimination graph into subtrees. In the second stage we project each subtree from level  $l$  to level  $l+0.5$ . In the third stage we project subtrees from level  $l+0.5$  to level  $l+1$ .

- *Subdividing the discrimination graphs into subtrees*

The first step in projecting a discrimination graph from level  $l$  onto level  $l+1$  is a subdivision of each discrimination graph into two-level subtrees. Each node in the graph and its direct descendants in a discrimination direction form a subtree. The next step is the assigning of a level number to each node in the discrimination graph. The leaves of the graph become level 1, their parents level 2 etc. In case of conflict the highest level number prevails. The projection takes place subtree by subtree. All trees with source node at level 2 are projected first, because all leaves at level 1 are already contained in a basic composition hierarchy. Next the trees at level 3 etc.

- *Projecting subtrees from composition level  $l$  to level  $l+0.5$*

For this stage we use the general projection algorithm specified in section 3.2.5. For the convenience of the reader it will be repeated here.

Each subtree  $ST$  consists of a source schema  $S$  and a set of leaves  $L$ .  $L$  has  $q$  elements:  $L_1, \dots, L_q$ . A discrimination link  $d_{i,l}$  connects  $S$  with  $L_i$  ( $1 \leq i \leq q$ ).  $n_i$  "must-be-part-of" links emanate from each  $L_i$  ( $1 \leq i \leq q$ ). The set of superschemata in the composition hierarchy of  $L_i$  we call  $SL_i$ .  $SL_i$  has  $r$  elements:  $SL_{i,1}, \dots, SL_{i,r}$  ( $r = n_i$ ). Figure C1 shows the situation.

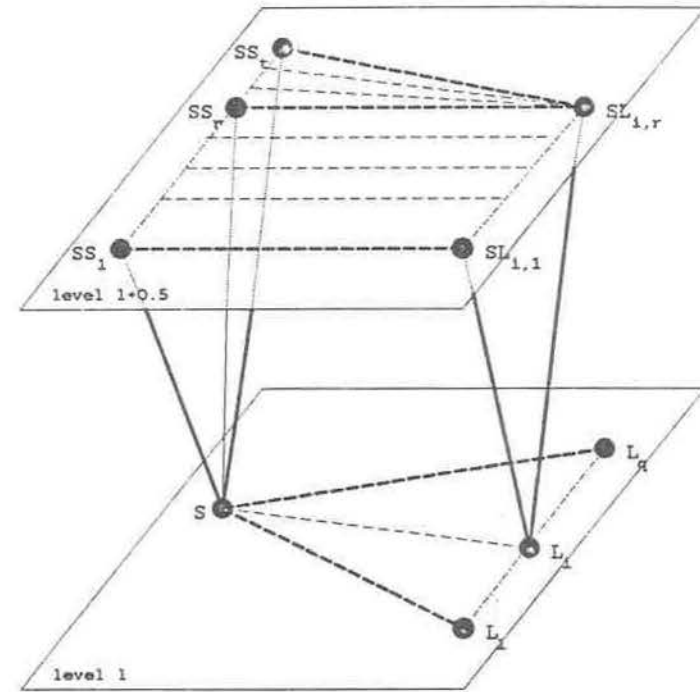


Figure C1: An illustration of the general projection algorithm

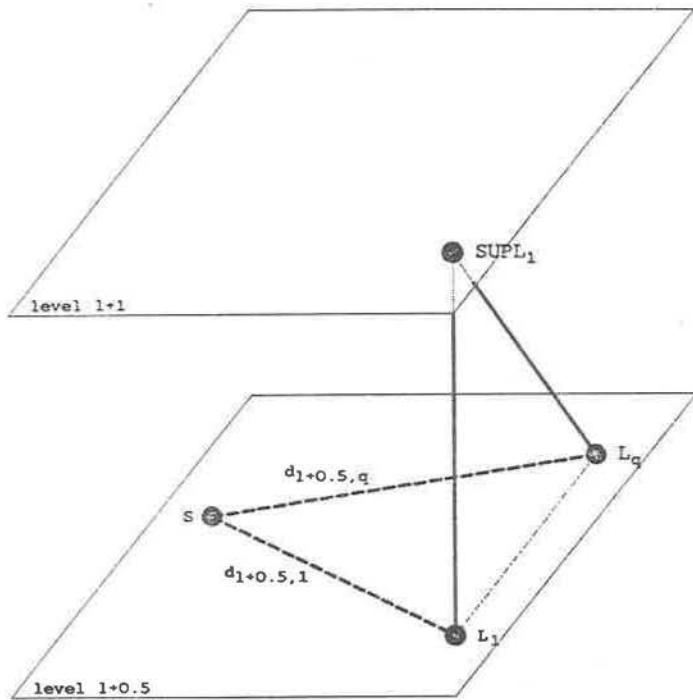


Figure C2: An extreme case of reduction in arity

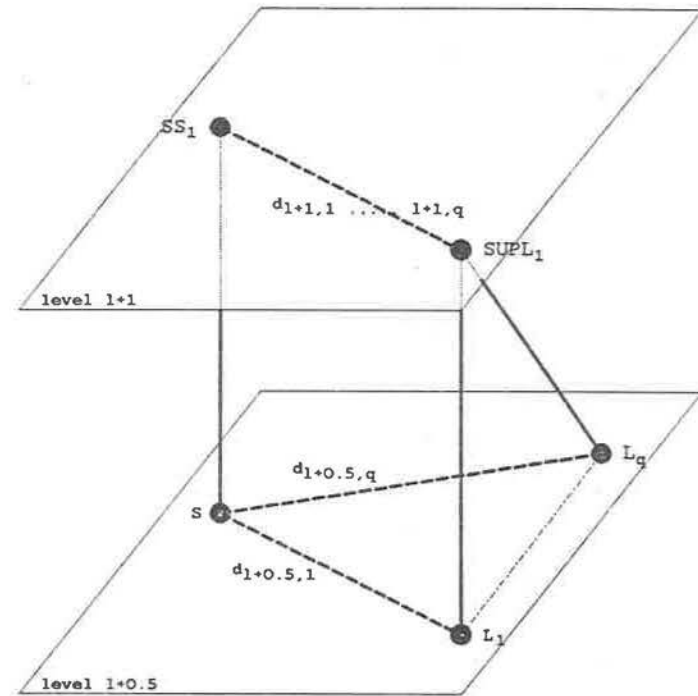


Figure C3: A decrease in arity as dealt with by the general projection algorithm

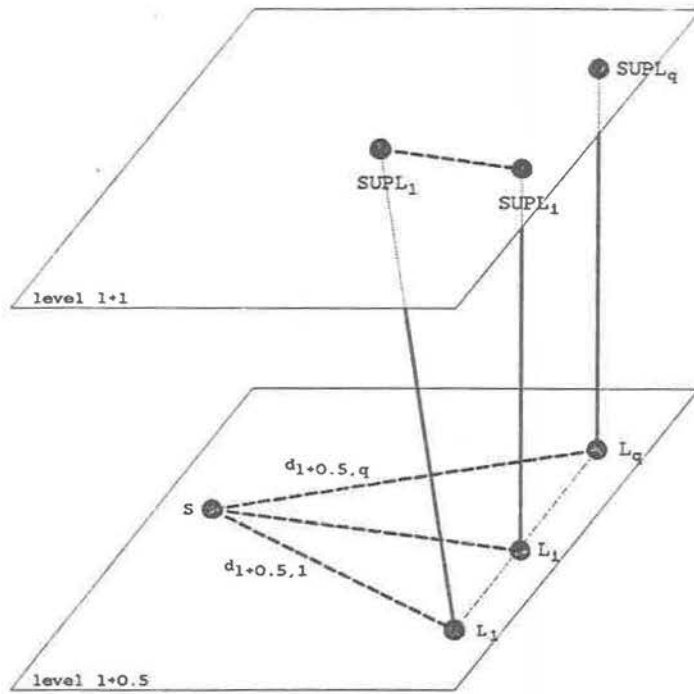


Figure C4: An example of two super-schemata which are a discrimination of another

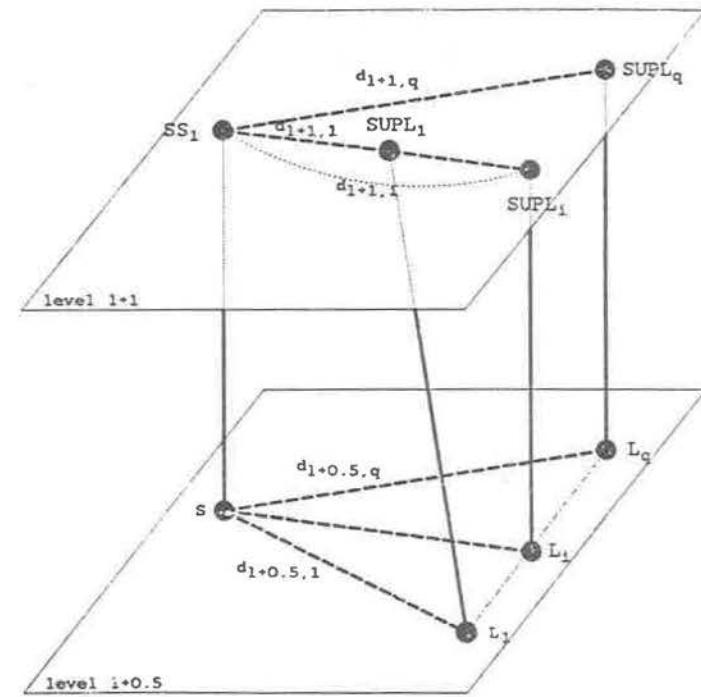


Figure C5: The solution of Figure C4 as provided by the general projection algorithm

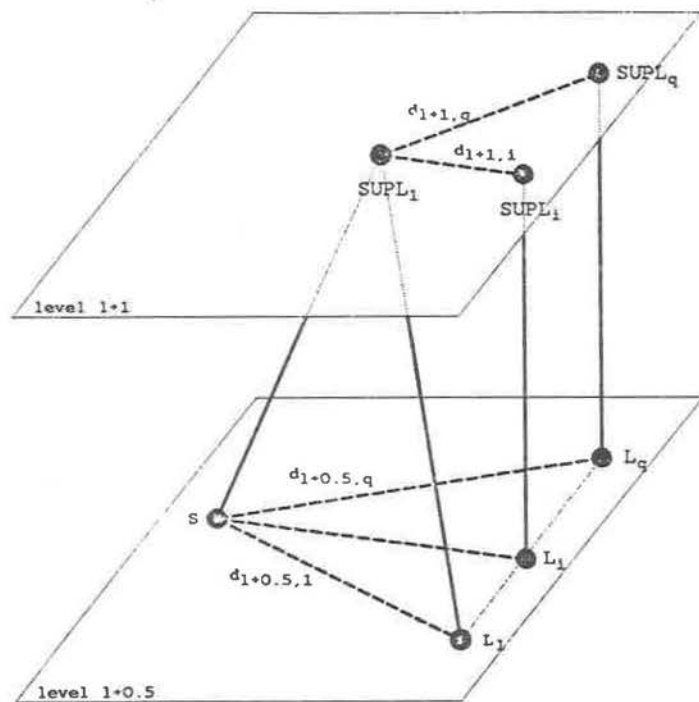


Figure C6: The desirable solution to the problem illustrated in

Figure C4

The projection algorithm consists of two steps:

1. Create a new set of super-schemata  $SS$  and create a "must-be-part-of" link between  $S$  and each element of  $SS$ .  $SS$  has  $t$  elements:  $SS_1, \dots, SS_t$  ( $t = \max n_i \{1 \leq i \leq q\}$ ).
2. For each set  $SL_i$  ( $1 \leq i \leq q$ ) do:
  - Create new discrimination links connecting  $SL_i$  with  $SS$ . Each of these links represent  $d_{i,l}$  at level  $l+1$ . The elements in both sets are connected in the following way:
    - $SL_{i,1}$  connects with  $SS_1$ ,  $SL_{i,2}$  connects with  $SS_2, \dots, SL_{i,r}$  connects with  $SS_r$ .
    - If  $r < t$  then create additional links that connect  $SL_{i,r}$  with  $SS_k$  for each possible value of  $k$  ( $r < k \leq t$ ).

- Projecting subtrees from composition level  $l+0.5$  to level  $l+1$

In this stage some efficiency measures are introduced. A "must-be-part-of" link connects two object class schemata. As a result each subtree leaf at level  $l+0.5$  can have at most one super-schema at level  $l+1$ . As well, different "must-be-part-of" links can point at the same super-schema. This causes a decrease in the arity of the subtree at level  $l+1$ . In the extreme case the arity is reduced to one. Figure C2 shows such a situation. The general projection algorithm deals inefficiently with this situation (Figure C3). Irrespective of the label to which  $S$  is refined,  $SS_1$  will always be refined to  $SUPL_1$ . A merge between  $SS_1$  and  $SUPL_1$  would therefore be appropriate.

Figure C4 shows another situation in which the general projection algorithm leads to inefficiencies. Discrimination graphs are not necessarily of uniform depth. As a result, the super-schemata of two different leaves at level  $l+0.5$  may be

discriminations from another at level  $l+1$ . Application of the general projection algorithm results in Figure C5. The existing link between  $SUPL_1$  and  $SUPL_i$  could have been used as a projection for  $d_{l+0.5,i}$  but this possibility has not been taken advantage of. A more efficient approach for such a situation is to find the most general schema in  $SUPL$  and to use that schema as super-schema for  $S$ . Figure C6 shows the result.

Finally, we can simplify the general projection algorithm by taking advantage of the restriction that  $S$  can have at most one super-schema. The revised projection algorithm consists of three steps:

1. Create the set  $SUPL$  consisting of the union of all  $SL_i$  ( $1 \leq i \leq q$ ).  
For each element  $el$  in  $SUPL$  do:  
  Check whether  $el$  is a discrimination of any other element in  $SUPL$ . If this is the case then delete  $el$  from  $SUPL$ .  
If  $|SUPL| = 1$   
  then execute step 2  
  else execute step 3
2. Create a "must-be-part-of" link between  $S$  and  $SUPL_1$ .
3. Create a new schema  $SS_1$  at level  $l+1$  and create a "must-be-part-of" link between  $S$  and  $SS_1$ .  
For each  $L_i$  ( $1 \leq i \leq q$ ) do:  
  If  $L_i$ 's super-schema  $SL_{i,1} \in SUPL$   
    then create a discrimination link between  $SL_{i,1}$  and  $SS_1$ . This link represents  $d_{l+0.5,i}$  at level  $l+1$ .

## APPENDIX D

### SYNTAX OF THE MAPSEE-3 SCHEMATA

A schema is either a scene schema or an image schema.

$\langle \text{schema} \rangle ::= \langle \text{image-schema} \rangle$  or  $\langle \text{scene-schema} \rangle$

A scene schema can represent an object class or relation. A scene schema can be described either by a list of attributes or by a single label that uniquely identifies the class or relation to the system. The former description is useful if we want to look at the constraints that exist between a schema and its neighbors in a constraint graph. The latter description is appropriate, if we want to treat a schema as an element in the domain of a variable for constraint satisfaction purposes.

The following attributes can be found in every scene schema  $X$ :

- 1) A schema-label: uniquely identifies an object class or relation to the system (e.g. \*S99).
- 2) type: indicates whether the schema represents an object class or relation.
- 3) composition level: each scene schema is embedded in a composition hierarchy.
- 4) discrimination level: each schema is also embedded in a discrimination graph, the concept of which is explained in section 3.2.2.
- 5) links-in: the list of schemata which have pointers directed at  $X$ .
- 6) links-out: the list of schemata to which  $X$  has pointers.



7) mandatory components: the list of schemata that enter in a "must-be-parts" relation with  $X$ .

8) other components: the list of schemata that enter in a "may-be-parts" relation with  $X$ .

9) mandatory super-components: the list of schemata that enter in a "must-be-part-of" relation with  $X$ .

10) other super-components: the list of schemata that enter in a "may-be-part-of" relation with  $X$ .

11) discriminations:  $X$ 's successors in the discrimination graph.

12) generalizations:  $X$ 's parents in the discrimination graph.

13) methods: a schema can represent both declarative and procedural knowledge. The schema's methods are procedures which are "owned" by the schema. Each Mapsee-3 method consists of a function that takes one or two arguments. A more detailed explanation of the operation of methods is provided in section 3.3.1.

14) instances: during interpretation each schema can be instantiated zero or more times. Each instantiation is represented as a uniquely identifiable unit. A scene schema instance  $Y$  has the following attributes:

- a) instance label: uniquely identifies  $Y$  to the system.
- b) iinverse: each relation in Mapsee-3 has an inverse. If  $Y$  is an instantiation of a relation, then it must have an inverse which is an instantiation of the inverse of  $Y$ 's parent schema.
- c) parent: the schema  $Y$  is an instance of.
- d) ilinks-out: the instance equivalent of "links-out".

e) ilinks-in: the instance equivalent of "links-in".

f) icomponents: the established components of  $Y$ .

g) isuper-components: the established super-components of  $Y$ .

h) labels: the list of current interpretation(s) of  $Y$ . At the time of creation  $Y$  inherits the label of its parent  $X$ . This label can be replaced by any of  $X$ 's successors in the discrimination graph, if the situation requires it.

i) idepicted-by: each schema instance is depicted by one or more image primitives.

15) inverse: see iinverse.

16) depicted-by: see idepicted-by.

```

<scene-schema> ::= <<schema-label><type>
                  <composition-level><discrimination-level>
                  <links-in><links-out>
                  <mandatory-components><other-components>
                  <mandatory-super-components>
                  <other-super-components>
                  <discriminations><generalizations>
                  <methods><instances>
                  <inverse><depicted-by>>

<schema-label> ::= <schema-label>
<type>          ::= *S-positive integer
                ::= object class
                ::= relation
                ::= positive number
                ::= positive number
<composition-level> ::= positive number
<discrimination-level> ::= positive number
<links-in>          ::= <scene-schema>*
<links-out>         ::= <scene-schema>*
<mandatory-components> ::= <> or <scene-schema>*
<other-components>   ::= <> or <scene-schema>*
<mandatory-super-components> ::= <> or <scene-schema>*
<other-super-components> ::= <> or <scene-schema>*

```

```

<discriminations> ::= <> or <scene-schema>*
<generalizations> ::= <> or <scene-schema>*
<methods> ::= <internal-methods>
               <external-methods>
<internal-methods> ::= <> or <<function-name <scene-schema>>*>
<external-methods> ::= <>
               <<function-name <<scene-schema>*>
                   <<scene-schema>*>*>
<function-name> ::= alpha-numeric string
<instances> ::= <> or <scene-schema-instance>*
<inverse> ::= <> or <scene-schema>
<depicted-by> ::= image primitive
<scene-schema-instance> ::= <<instance-label><parent><iilinks-in>
                           <iilinks-out><icomponents><isuper-components>
                           <iinverse><labels><idepicted-by>>
                           <instance-label>
<instance-label> ::= *S positive integer - positive integer
<iinverse> ::= <scene-schema-instance>
<parent> ::= <scene-schema>
<iilinks-in> ::= <scene-schema-instance>*
<iilinks-out> ::= <scene-schema-instance>*
<icomponents> ::= <> or <scene-schema-instance>*
<isuper-components> ::= <> or <scene-schema-instance>*
<labels> ::= <scene-schema>*
<idepicted-by> ::= image primitive instance

```

An image schema can represent a point, link, line, chain, patch, or region.

```

<image-schema> ::= <point-schema>
                <link-schema>
                <line-schema>
                <chain-schema>
                <patch-schema>
                <region-schema>

```

The input to Mapsee-3 is a line drawing given as a set of plotter commands (e.g. plot (x,y) and goto (x,y)). For each point the coordinates are given. Each pair of connected points forms a link. A set of links forms a chain. A line hierarchy is created for every chain by following the procedure described in section

4.3.2. Each point instance is specified by its parent, coordinates, and the link it is part of (if any). A link instance is specified by its start-point, end-point, parent, and the chain it is part of.

```

<point-schema> ::= <*point <point-instances>>
                ::= *point
<point-instances> ::= <> or <point-schema-instance>*
<point-schema-instance> ::= <<point-identifier> *point
                           <ppart-of><coords>>
                           <point-identifier>
<point-identifier> ::= *point - positive integer
<ppart-of> ::= <link-schema-instance>*
<coords> ::= real . real
<link-schema> ::= <*link <link-instances>>
                ::= *link
<link-instances> ::= <> or <link-schema-instance>*
<link-schema-instance> ::= <<link-identifier> *link
                           <start-point><end-point>
                           <lpart-of>>
                           <link-identifier>
                           <start-point>
                           <end-point>
                           <lpart-of>
                           ::= <chain-schema-instance>

```

A line instance is specified by its parent, end-points, the chain it is part of, its components in the line hierarchy (see section 4.3.2), the distance to the furthest point in its set (deviance), some line parameters, its length, and the point associated with the deviance (deviant).

```

<line-schema> ::= <*line <line-instances>>
               ::= *line
<line-instances> ::= <<> or <line-schema-instance>*
<line-schema-instance> ::= <line-identifier>
                          ::= <<line-identifier> *line
                             <deviance><lnparam>
                             <length><components>
                             <deviant><chain><ends>>

<line-identifier> ::= *line - positive integer
<deviance> ::= positive real
<lnparam> ::= real . real . real
<length> ::= positive real
<components> ::= <<line-schema-instance>
                 <line-schema-instance>>

<deviant> ::= <point-schema-instance>
<chain> ::= <chain-schema-instance>
<ends> ::= <<point-schema-instance>
          <point-schema-instance>>

```

A chain instance is characterized by its parent, the top-line in the chain, the links that constitute a chain, the first link (forward), the last link in the chain (reverse), and a set of features ( $f_1, \dots, f_n$ ) that characterize the chain's shape.

```

<chain-schema> ::= <*chain <chain-instances>>
                ::= *chain
<chain-instances> ::= <<> or <chain-schema-instance>*
<chain-schema-instance> ::= <chain-identifier>
                          ::= <<chain-identifier> *chain
                             <top-line><links><forward>
                             <reverse><f1.....fn>>

<chain-identifier> ::= *chain - positive integer
<top-line> ::= <line-schema-instance>
<links> ::= <link-schema-instance>*
<forward> ::= <link-schema-instance>
<reverse> ::= <link-schema-instance>
<f1>...<fn> ::= True or False

```

The image is subdivided into square patches. The patch formation process is described in section 4.3.3. Regions consist of interconnected empty patches. Each

patch has four neighbors, and can be subdivided into four subpatches. The mid-coordinates are the intersection point of the diagonals. Empty patches are part of a region.

```

<patch-schema> ::= <*patch <patch-instances>>
                ::= *patch
<patch-instances> ::= <<> or <patch-schema-instance>*
<patch-schema-instance> ::= <patch-identifier>
                          ::= <<patch-identifier> *patch
                             <neighbors><subpatches>
                             <mid-coords><area>
                             <where><ptchpart-of>>

<patch-identifier> ::= *patch - positive number
<neighbors> ::= <<patch-schema-instance>
                <patch-schema-instance>
                <patch-schema-instance>
                <patch-schema-instance>>

<mid-coords> ::= real . real
<area> ::= positive real
<where> ::= real
<ptchpart-of> ::= <<> or <region-schema-instance>

<region-schema> ::= <*region <region-instances>>
                 ::= *region
<region-instances> ::= <<> or <region-schema-instance>*
<region-schema-instance> ::= <region-identifier>
                          ::= <<region-identifier> *region
                             <patches>>

<region-identifier> ::= *region - positive integer
<patches> ::= <patch-schema-instance>*

```

APPENDIX E  
THE MAPSEE-3 INTERPRETER

The Mapsee-3 interpreter represents each image primitive at all levels of composition with an appropriate interpretation. Interpretation consists of two processes: composition and discrimination. The former, in turn, consists of two stages: completion and assembly. A flow chart of the interpretation process is provided in Figure 4.13.

The composition process is constrained to operate in the composition/aggregation dimension just as discrimination is constrained to operate in the discrimination/generalization dimension. The modularity of the two processes is further enhanced by their means of communication. Neither process can call the other directly. They communicate through two different queues: a completion queue and a consistency queue. The completion process has read and write access to the first queue and write access to the second queue. The discrimination process has read and write access to the consistency queue and write access to the completion queue. Control is switched between completion, discrimination, and assembly according to the flow chart in Figure 4.13. Discrimination is the equivalent of H.A.C. (Hierarchical Arc Consistency).

Image-to-scene mapping results in the creation of a number of instances at the composition leaf level, one for each image primitive. The objective of composition is to represent these leaf level instances at each of the other levels of composition, thereby establishing spatial relationships between them. Completion of

an instance results in a representation of the instance at the next level up in the composition hierarchy. Discrimination, on the other hand, ensures that each primitive is represented at an appropriate level of discrimination.

The interpreter is described by the following procedure:

*Procédure INTERPRET*

```

1 BEGIN
2   WHILE completion-queue non-empty
3     BEGIN
4       select and delete first element x from completion queue
5       new-super-components,new-relations ← COMPLETE (x)
6
7       IF HAC-3 THEN
8         BEGIN
9           ASSEMBLE (x, new-super-components,new-relations)
10          HAC-3
11
12          IF new-super-components THEN
13            ((sc) | sc ∈ new-super-components) insert
14            sc into the completion-queue by composition level
15
16          END
17
18        ELSE return failure
19
20      END
21
22    return success
23  END

```

The procedure *COMPLETE* represents completion. It returns one or more super-components for *x*, found or created at the next higher level of composition. If new spatial relations are created during completion then *COMPLETE* returns

these as well. *HAC-3* represents discrimination. The procedure *ASSEMBLE* represents assembly. Composition is subdivided into completion and assembly, because we want to ensure that the labels of  $x$  are made consistent with those of its super-component(s) before we connect  $x$  with the existing interpretation graph. During the first invocation of *HAC-3*, the labels of  $x$  are made consistent with the labels of its super-component(s). The reverse is done during the second *HAC-3* invocation. This, however, can be done after assembly, because consistency is symmetrical, i.e. if consistency can be obtained in one direction then it can also be achieved in the other. The *ASSEMBLE* procedure assembles  $x$  and the newly created spatial relations into the existing interpretation graph. An insertion by composition level in the completion queue will cause *COMPLETE* to operate in a breadth-first manner. A push-pop mechanism will result in a depth-first operation.

Procedure *COMPLETE* ( $x$ )

```

0 BEGIN
1  new-super-components  $\leftarrow$  empty
2  new-relations  $\leftarrow$  empty
3   $l \leftarrow$  nearest common generalization of the labels of  $x$ 

4  For each super-component  $s$  of  $l$  DO
5    BEGIN
6      merge-queue  $\leftarrow$  empty
7      match  $\leftarrow$  false
8       $sdi \leftarrow \{(i) \mid i \text{ is an instance of } d, \\ d \in \text{superdiscrimination set of } s\}$ 

9      For each  $i \in sdi$  DO
10       BEGIN
11         comps  $\leftarrow$  COMPONENTS ( $i$ )

```

```

12   For each  $j \in \text{comps}$  DO
13     IF SPATIAL-RELATION  $R(x,j)$  THEN
14       BEGIN
15         match  $\leftarrow$  true
16          $r \leftarrow$  NEW-INSTANCE  $R(x,j)$ 
17         new-relations  $\leftarrow$  new-relations  $\cup r$ 
18         push ( $x,r$ ) onto consistency-queue
19         push ( $x,i$ ) onto consistency-queue
20         merge-queue  $\leftarrow i \cup \text{merge-queue}$ 
21       END

22   END

23   IF match = false THEN
24     BEGIN
25        $i \leftarrow$  NEW-INSTANCE ( $s$ )
26       push ( $x,i$ ) onto consistency-queue
27       merge-queue  $\leftarrow i \cup \text{merge-queue}$ 
28     END

29    $a \leftarrow$  1st element of merge-queue
30    $B \leftarrow$  remainder of merge-queue

31   for each  $b \in B$  DO
32     BEGIN
33        $a \leftarrow$  MERGE ( $a,b$ )
34     END

35   new-super-components  $\leftarrow a \cup \text{new-super-components}$ 
36   consistency-queue  $\leftarrow$  consistency-queue  $\cup \{(a,n) \mid (a,n) \in \text{arcs}(G), a \neq n\}$ 
37   consistency-queue  $\leftarrow$  consistency-queue  $\cup \{(n,a) \mid (n,a) \in \text{arcs}(G), a \neq n\}$ 

38   END

39   return new-super-components, new-relations

40 END

```

Completion is a data driven process. In order to find the super-components of schema instance  $x$ , we first investigate  $x$ 's labels. If  $x$  has only one label, say  $v$ , then we only have to trace  $v$ 's super-schema in the composition hierarchy. However, if  $x$  has more than one label (say  $v$  and  $w$ ) then we cannot just take  $v$  and  $w$ 's super-schema because this would introduce a hypothetical interpretation in the composition/aggregation dimension. Instead, we take the nearest common generalization of  $v$  and  $w$  in the discrimination graph, say  $l$ , and trace down  $l$ 's super-schema(ta) instead. In this way completion takes place in a non-hypothetical manner. The two HAC-3 invocations in *INTERPRET* will ensure that any new super-component for  $x$  will obtain again a proper and consistent label set.

If a schema  $l$  is a component of schema  $s$  then an instance of  $l$  can be a component of *at most one* instance of  $s$ . Whenever an instance of  $l$  is compatible with two instances of  $s$  then this is a sign that the two instances should be merged. For this purpose a *merge-queue* is used. The potential set of instances to which  $x$  can complete is determined by the superdiscrimination set of  $s$ . This was explained in section 3.3.1.

In the lines 10 - 21 a search is initiated for a spatial relation between  $x$  and an existing component  $j$  of a potential super-component for  $x$ . The Mapsee-3 composition hierarchy is based on the premise that an instance can only become a component of a super-instance if a spatial relationship exists between the instance and an existing component of the super-instance. The search for such a

relationship is carried out by the super-instance's methods which are invoked by *COMPLETE*. This is done by the subroutine *SPATIAL-RELATION*, invokes all of  $d$ 's methods in an attempt to establish a spatial relation between  $x$  and  $j$ . If a method is successfully applied then a new instance of a spatial relation is created. Notice that only  $(x,r)$  is pushed onto the consistency queue and not  $(r,x)$ . The latter is done during assembly.

If no matching super-component can be found for  $x$ , then a new instance is created for  $s$  which becomes the super-component for  $x$ . If  $|merge-queue| > 1$ , then all of its elements are merged into one. This is done in the lines 29 - 34.

The procedure *COMPLETE* has been kept as simple as possible at the cost of sacrificing some detail and efficiency. The fact that there is a level of composition relations between  $x$  and  $s$  has been omitted. As well, *SPATIAL-RELATION* checks whether the spatial relation under investigation has already been created. The line 35 - 37 are necessary only when a merge between elements has actually taken place. If this has not been the case then they are redundant.

procedure HAC-3

```

1 BEGIN
2   for  $i \leftarrow 1$  until  $n$  DO NC( $i$ )
3   REPEAT until consistency-queue is empty
4     BEGIN
5       select and delete first arc ( $k,m$ ) from consistency-queue

6     IF HACREVISE( $k,m$ ) THEN
7       BEGIN
8         consistency-queue  $\leftarrow$  consistency-queue  $\cup$   $\{(i,k)$ 
            $\mid (i,k) \in \text{arcs}(G),$ 
            $i \neq k, i \neq m\}$ 
9         insert  $k$  into completion-queue by composition level

10      END

11    END

12  END

```

HAC-3 has already been described in Appendix A. The only additional point of interest is that HAC-3 inserts instance  $k$  into the completion queue whenever  $k$ 's label(s) have changed. A subsequent completion may reveal new spatial relations and may lead to new mergers between super-components.

ASSEMBLE is a very simple procedure. It links the completing instance with its new super-components and it assembles the new relations. As the label(s) of the super-components have not yet been made consistent with the label(s) of their new components the appropriate links are pushed on the consistency queue. The label(s) of the super-components will then be updated during the second HAC-3 invocation in INTERPRET.

procedure ASSEMBLE (instance, comps, rels)

```

1 BEGIN
2   for each rel  $\in$  rels DO
3     BEGIN
4       link rel to its source and destination
5       push (source,rel) on consistency-queue
6       push (dest,rel) on consistency-queue
7     END

8   for each comp  $\in$  comps DO
9     BEGIN
10      link instance with comp and vice versa
11      push (comp,instance) on consistency-queue
12    END

13  END

```

The Mapsce-3 interpreter is largely domain-independent. It can operate on any schema-based representation that adheres to the syntactic constraints specified in Appendix D. The only domain-dependent subroutine is SPATIAL-RELATION in the procedure COMPLETE. This routine invokes procedures which are specific for the schema involved and the domain concerned.