KNOWLEDGE STRUCTURING AND CONSTRAINT SATISFACTION: THE MAPSEE APPROACH

Jan A. Mulder[1], Alan K. Mackworth[2], and William S. Havens[3]

Technical Report 87-21

June 1987

[1]Department of Mathematics, Statistics, and Computing Science, Dalhousie University, Halifax, N.S., Canada B3H 3J5

[2]Fellow, Canadian Institute for Advanced Research,
Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada V6T 1W5

[3]Tektronix Research Laboratories, p.o. box 500, Portland, Oregon, 97005 U.S.A.

Abstract

This paper shows how to integrate constraint satisfaction techniques with schema-based representations for visual knowledge. This integration is discussed in a progression of three sketch map interpretation programs: Mapsee-1, Mapsee-2, and Mapsee-3. The programs are evaluated by the criteria of descriptive and procedural adequacy. The evaluation indicates that a schema-based representation used in combination with a hierarchical arc consistency algorithm constitutes a modular, efficient, and effective approach to the structured representation of visual knowledge. The schemata used in this representation are embedded in composition and specialization hierarchies. Specialization hierarchies are further expanded into discrimination graphs.

1 Introduction

This paper is concerned with the use of constraint satisfaction techniques in model-based computational vision. Particular attention is paid to the integration of constraint satisfaction techniques with schema-based representations. This methodology is examined under two criteria: ***descriptive adequacy***, the ability of a representational formalism to capture essential visual properties of objects and relationships in the visual world, and *procedural adequacy*, the capability of the representation to support efficient processes of recognition, generation, knowledge acquisition, and search [27].

Imagine a "general-purpose" vision system which takes as input a digitized picture of a natural scene and which produces as output a meaningful description of such a scene. Knowledge of a wide variety of domains would be represented in such a system. The system would be able to interpret virtually any image without specific prior expectations of the domain of interpretation. The system would be capable of describing each image and scene at different levels of detail and abstraction (descriptive adequacy). As well, it would be able to transform a description of the image in terms of significant features into a domain-specific scene description correctly, quickly, and flexibly (procedural adequacy).

The prospect of such a "general-purpose" system remains a vision of the not so near future. The main dilemma is that computational vision is inherently an underconstrained task. The image formation process confounds information about the objects and their spatial relationships, the image projection process, the surface reflectance properties of scene objects, the occlusion of surfaces, the viewing position of the observer, and the like. To overcome this dilemma, computational vision systems must use additional sources of knowledge to provide the necessary constraints to the recognition process. These constraints can range from general knowledge assumed valid for all scenes (*Early Vision*) to specific knowledge about scene objects and their legitimate configurations (*Model-based Vision*) [23].

Research in computational vision has successfully exploited assumptions about surface descriptions [28], scene illumination and surface continuity [14], viewing position [9,1], lighting position [48], surface reflectance [42], motion [45,18], surface topology in scene analysis [36,15,6,46], and surface orientation [19,16]. Unfortunately, these constraints, while apparently necessary for producing scene descriptions in a particular application, are not valid for all scene domains. General knowledge is frequently embedded in the particular theory being implemented and not explicitly represented. Furthermore, many scene level constraints are only valid for particular objects and particular scene configurations. Computational vision requires explicit object-oriented representations of visual knowledge. The schema-based methodology discussed in this paper provides such a representation.

Research directed towards characterizing effective representations for visual knowledge forms one part of efforts to establish a coherent theory for visual perception. The coordination among different knowledge sources through constraints forms another part. Most model-based vision systems obtain access to the knowledge base of a particular domain after a segmentation process has scanned the image in search of particular features. The description of these features imposes constraints on the interpretations possible. Features, constraints, and interpretations can be represented as a graph (or hypergraph) with the features as variables, each with an associated domain of possible interpretations, and the constraints as relations. Suppose there are n variables each with domain size a and there are e relations among the variables to be satisfied. The problem of finding globally consistent interpretations for the image then becomes the problem of finding all possible n-tuples, such that each n-tuple is an instantiation of the n variables satisfying the relations. This problem is the Constraint Satisfaction Problem (CSP)[20]. Since, in its full generality, CSP is NP-hard the existence of algorithms that solve this problem in less than exponential time is unlikely. Depth-first backtracking, for instance, is of $O(ea^n)$ in its worst case performance [24]. Such performance causes major difficulty in a general-purpose vision system in which we can expect domain sizes to be very large.

A variety of approaches have been taken to solving CSP's [26]. In particular, since the vision CSP is NP-hard various polynomial constraint satisfaction *approximation algorithms* that use *constraint propagation* techniques are attractive. In the next section we briefly discuss this approach.

This paper focusses on the integration of constraint propagation techniques with schema-based representations. We discuss this interaction in a progression of three sketch map interpretation progams. In these programs a scene interpretation takes the form of a *scene constraint graph.* Schema-based representations are involved in the construction of this graph, whereas network consistency algorithms control the propagation of constraints through the graph.

Sketch maps provide a profitable domain for studying visual knowledge in relative isolation while not ignoring the problems of segmenting and interpreting real imagery. Sketch maps capture in a simple form fundamental problems of representing and applying knowledge. Furthermore, techniques for understanding sketch maps have application in interpreting real imagery [8,43]. All three programs, Mapsee-1, 2, and 3 are evaluated using the evaluation criteria of descriptive and procedural adequacy. A schema-based representation is introduced in Mapsee-2.

2 Constraint Propagation

In a graph that contains n variables, a constraint satisfaction algorithm has to test all possible explicit and implicit k-ary constraints ($k \leq n$). The term constraint propagation is often used as an umbrella for a group of algorithms which propagate constraints over a graph but which do not always solve the constraint satisfaction problem. They are approximation algorithms in that they enforce necessary but not always sufficient conditions for the existence of a solution. Constraint propagation algorithms evolved from research in the polyhedral domain [46,20], but have a much wider applicability

[26]. Most of these algorithms are characterized by the fact that they test constraints in a local neighbourhood only. Node consistency [20], for example, tests for unary constraints only, arc consistency [20,24] tests unary and binary constraints, whereas k-consistency [7] tests up to k-ary constraints ($k \leq n$).

In order to find all n-tuple variable instantiations that satisfy the constraints, constraint satisfaction algorithms have to test all possible combinations between values in the variable's domain. Algorithms like depth-first backtracking keep an explicit record of consistent value combinations. Image features have many possible local interpretations. However, when more global constraints are imposed most of the local interpretations are invalidated. One particular weakness of depth-first backtracking is that it keeps track of many interpretation combinations which will eventually be eliminated. The advantage of constraint propagation algorithms such as arc consistency is that it does not keep track of all interpretation combinations. Arc consistency tests and eliminates domain values that are inconsistent with all values in the domains of adjacent variables.

The algorithm *AC-3* [20] illustrates this operation which is characteristic for constraint propagation algorithms. Various derivatives of this algorithm were used in the Mapsee programs to be described later. *AC-3* tests for unary and binary constraints. Each edge in the constraint graph is replaced by two directed arcs. First, *AC-3* tests the unary constraints in the graph. Node i, which consists of vertex i, its associated domain (D_i), and the unary constraint P_i, is ***node consistent*** iff

$$(\forall x)\{(x \in D_i) \supset P_i(x)\}$$

This consistency can be obtained by applying the domain restriction operation:

$$D_i \longleftarrow D_i \cap (x|P(x))$$

Next, we test binary constraints. Arc (i, j) is ***arc consistent*** iff

$$(\forall x)\{(x \in D_i) \supset (\exists y)[(y \in D_j) \wedge P_{ij}(x, y)]\}$$

That is, for every element in D_i there exists at least one element in D_j such that the binary constraint P_{ij} is satisfied. The corresponding domain restriction operation ensures that all elements in D_i which are not arc consistent are removed from D_i:

$$D_i \longleftarrow D_i \cap \{x|(\exists y)[(y \in D_i) \wedge P_{ij}(x,y)]\}$$

With the set of operations given above, *AC-3* has the following structure:

```
procedure NC((i))
1.          begin
2.          $D_i \longleftarrow D_i \cap (x|P_i(x))$
3.          end
```

```
procedure REVISE ((i,j))
1. begin
2.       $DELETE \longleftarrow$ 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.                           $DELETE \longleftarrow$ true
8.                      end
9.       return $DELETE$
10. end
```

```
procedure AC-3
1. begin
2. for $i \longleftarrow 1$ until $n$ do NC((i))
3.    $Q \longleftarrow \{(i,j)|(i,j) \in arcs(G), i \neq j\}$
4.    while $Q$ not empty do
5.          begin
6.          select and delete any arc $(k,m)$ from $Q$
7.          if REVISE $((k,m))$ then $Q \longleftarrow Q \cup \{(i,k)|(i,k) \in arcs(G)$
                                                   $i \neq k\}$
8.          end
9. end
```

The operation described in line 7 of *AC-3* ensures that upon deletion of an element in the domain of vertex k arcs incoming to k are added to Q. *AC-3* does not maintain an explicit record of value compatibility. REVISE discontinues the search at the first y in D_j for which $P_{ij}(x, y)$.

AC-3 is a polynomial time algorithm which is a major advantage. Based on the number of predicate evaluations, *AC-3* is of $O(ea^3)$ in the worst case and of $O(ea^2)$ in the best case [24]. This is a major improvement over exponential depth-first backtracking.

The price, of course, is that *AC-3* is not guaranteed to solve the constraint satisfaction problem. The strength of algorithms like *AC-3* is that they can be used as a pre-processor for a constraint satisfaction algorithm such as backtracking or recursive arc consistency with domain subdivision. *AC-3* removes all elements that are locally inconsistent. This has the net effect of reducing the domain size of each variable. Thus, backtracking can operate with a relatively small domain size. Experience with the Mapsee programs has taught us that *AC-3* often reduces the domain size of each variable to one. In this case, this corresponds to a solution. Under this circumstance *AC-3* provides a linear time algorithm for finding the one solution satisfying all constraints.

The complexity properties make constraint propagation algorithms an attractive tool to use in image interpretation. As mentioned before, general-purpose vision systems will have to cope with large domain sizes. Efficient constraint satisfaction algorithms are therefore necessary. Efficient constraint propagation, however, is not the only problem model-based vision systems have to overcome. The construction of a constraint graph over which the constraints are propagated is a problem by itself. We will see that combining a schema-based representation with constraint propagation techniques can provide a modular and efficient integration of graph construction and constraint propagation. Additionally, this methodology will be seen to provide a high level of both descriptive and procedural adequacy in a vision system.

All programs discussed in the next three sections are part of the Mapsee project, a project that studies different schemes for representing, and evaluating visual knowledge. Different knowledge representation schemes have been implemented as sketch map interpretation programs. We will describe three such programs: Mapsee-1, 2, and 3. None of these programs, however, will be described in detail. A description of Mapsee-1 appeared in [21]. No implementation-level description of Mapsee-2 exists, other than the one provided here. Some of Mapsee-2's design principles have been reported before [11,12]. Some aspects of the Mapsee-3 program have been reported elsewhere [31,32,33,34].

3 Mapsee-1

Sketch maps are man made images whose semantics are rich and clean, and they can be codified and exploited. Sketch maps can be freehand sketches, or, as is the case in Fig. 1, they can be created by tracing an aerial photograph or cartographic map. The semantics of sketch maps are partially natural (e.g. the shape of rivers, roads and coastlines) and partially conventional (e.g. bridges and mountains). An important property shared with real imagery is that image features are highly ambiguous when it comes to interpretation. Line segments, for instance, can be roads, rivers, shores, bridges, town, and mountains, whereas the "empty" areas in between can be land or water.

3.1 Segmentation

All Mapsee programs go through a more or less uniform segmentation stage first. We therefore discuss this stage as a common basis for all three programs. The sketches are encoded as a series of "goto" and "plot" commands. Both commands specify a move between two coordinates in an x-y frame while the pen is up or down.

The objective of segmentation is the construction of primitives and features. The

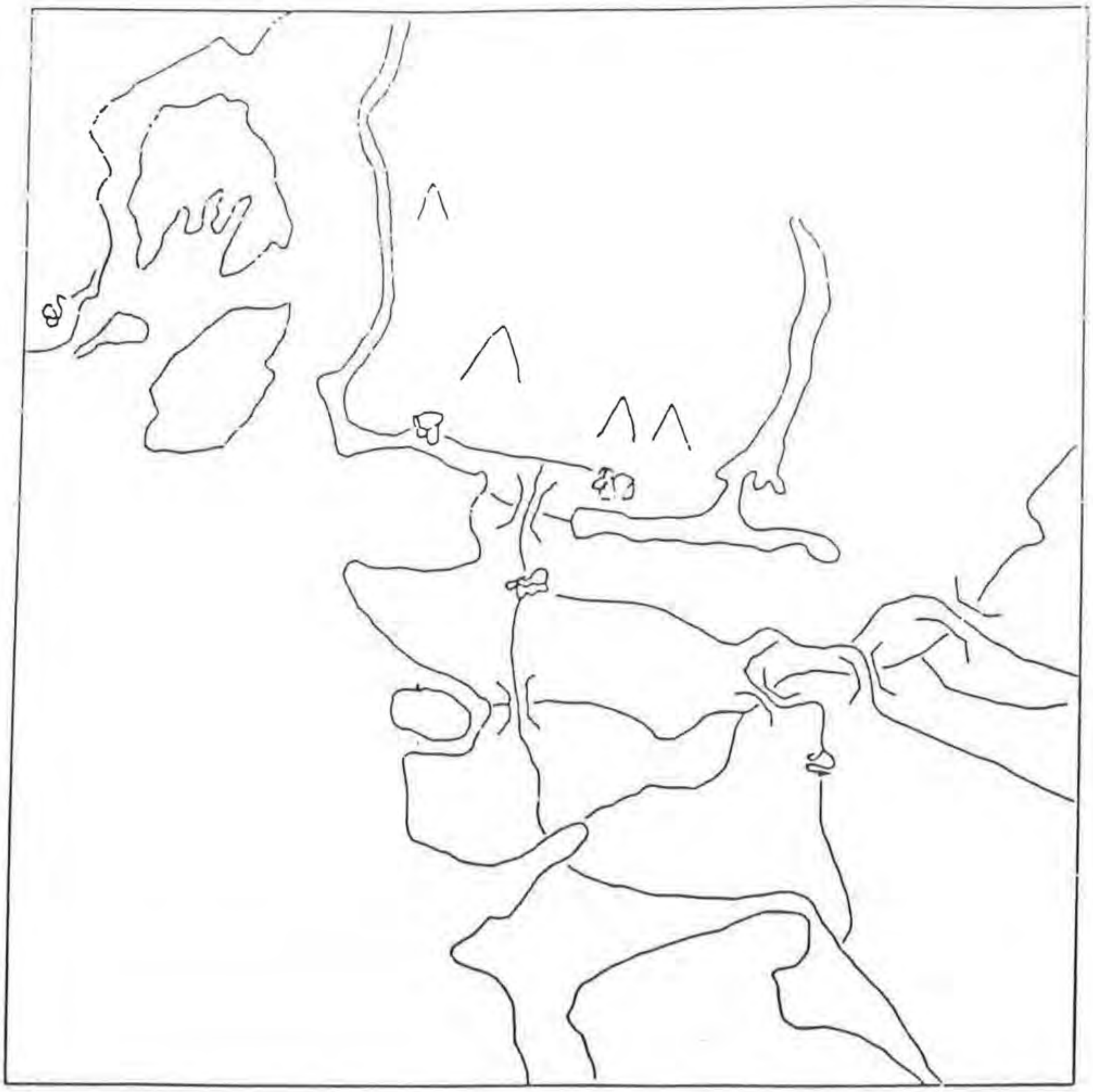

Figure 1: Lower Mainland of British Columbia

primitives are the elements of the image. They are considered to represent a scene object or parts of it. The features, on the other hand, are the image properties that constrain the possible interpretations for the primitives of which they are a part. Features also serve as cues for interpretation. Regions and line segments (called *chains*) are the primitives in Mapsee. Figure 2 shows the different junction types used as features in Mapsee-1. Figure 3 shows the possible interpretations for each of them. Each feature constrains the interpretations for each of the adjacent chains and regions. For instance, the stem of a Tee junction can be a river, the bar a shore, one of its regions (RA) sea, and the other two land. Another possibility is a mountain interpretation for both the stem and bars, in which case all regions must be land.

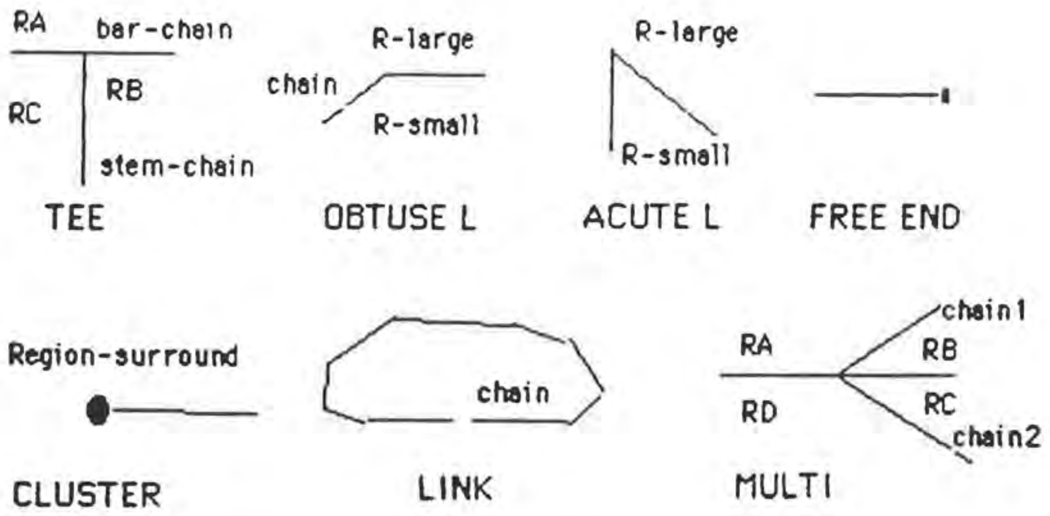

Figure 2: Mapsee-1 features

No search is necessary for the line segments as they are given in the input. The computation of features, on the other hand, requires some shape distinction. A hierarchical line approximation of each chain provides such a distinction [31]. The points of each chain are also represented in a coarse array (32x32). Such a representation allows for quick answers to adjacency questions. This representation also supports the region formation process.

TEE	stem-chain	bar-chain	RA	RB	RC	
	river	shore	sea	land	land	
	river	shore	lake	land	land	
	river	river	land	land	land	
	road	road	land	land	land	
	mountain	mountain	land	land	land	
	river	bridge	land	land	land	
OBTUSE L	chain	R-large	R-small			
	shore	lake, sea	land			
	shore	land	lake, sea			
	road, bridge, river	land	land			
ACUTE L	chain	R-large	R-small			
	shore	lake, sea	land			
	shore	land	lake, sea			
	road, mountain, river	land	land			
FREE END	chain	Region-surround				
	river	land				
	mountain, bridge	land				
CLUSTER	chain	Region-surround				
	road	land				
LINK	chain					
	shore					
MULTI	through-chain	chain1	chain2	RA	RB	RC
	river	river	river	land	land	land
	road	road	road	land	land	land

Figure 3: Mapsee-1 cue interpretation catalogue

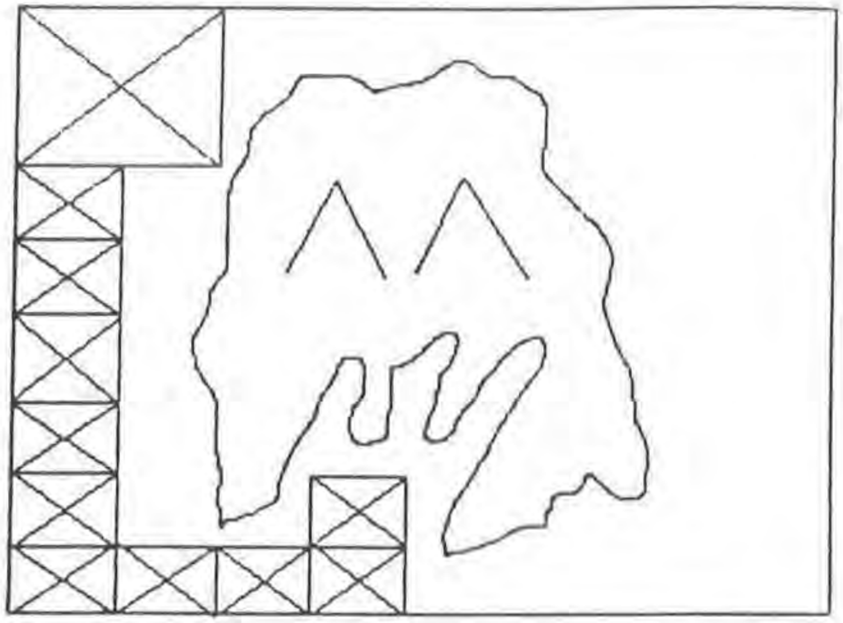

Figure 4: region formation

Region formation uses a split-and-merge technique [35]. A recursive algorithm partitions the image into empty patches. Each non-empty patch is subdivided into four sub-patches. The subdivision continues until a patch is empty or until a predefined minimum patch size is reached. Each set of interconnected empty patches is merged to form a region. The patch formation process is conservatively biased. That is, subdivision stops well before any "leakage" through (non-intended) gaps between chains occur. Figure 4 shows an example of region formation. The minimum patch size is relatively large. The waterbody surrounding the island will therefore consist of two regions, one of which is shown. This problem is solved during interpretation.

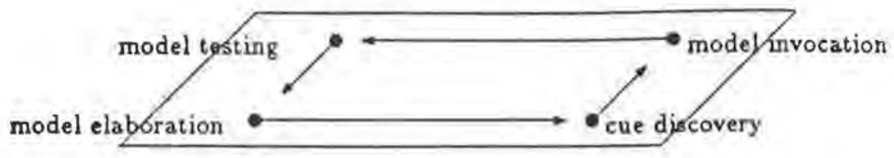

Figure 5: The cycle of perception

3.2 Interpretation

The segmentation process cumulates in the formation of chains, regions, and cues. Mapsee-1 represents its scene models in the form of cue/model tables such as the one shown in figure 3. A constraint satisfaction algorithm, called *network consistency* (*NC*) closely interacts with this table. This interaction takes the form of a cycle of perception as shown in figure 5 [22]. The program goes through a sequence of stages during each pass: cue discovery, model invocation, model testing, and model elaboration. The cycle is entered at cue discovery which is the equivalent of segmentation. Model invocation embodies the construction of a *scene constraint graph* in which the the primitives (chains and regions) are the "variables", each with a domain of possible interpretations, the arcs are instantiations of the cues found during segmentation. Model testing is the equivalent of node consistency described in the previous section, whereas model elaboration is an implementation of *NC*.

Mapsee-1 closes the cycle. The results of model elaboration are used to resegment the image, if necessary. Thus, the two regions surrounding the island in Fig. 4 can be merged. The cycle of perception is a useful metaphor for describing and comparing control structures in computer vision programs [22].

NC is a generalization of *AC-3*. *NC* relaxes *AC-3*'s binary constraints in that it can deal with arbitrary k-ary relations. Its manner of operation is similar to *AC-3* except that upon completion of the *AC-3* part, *NC* will test whether each domain contains one label only. If this is the case then it terminates. If one domain contains more than one label (say b) then NC returns b different scene constraint graphs. If more than one domain contains more than one label then one domain is split in approximately equal halves and NC is invoked recursively for the two newly generated subproblems.

NC tests the full range of relationships and is therefore a complete constraint satisfaction algorithm. However, NC's operation may yield exponential complexity because of the recursive domain splitting. A surprising finding, however, was that for the sketches tested, the domain splitting operation did not need to be invoked.

3.3 Discussion

Mapsee-1 was successful as a demonstration of the applicability of constraint propagation algorithms outside the domain of polyhedral scenes. As well, it is an existence proof that cues and descriptive models can also be used outside the polyhedral world. On the negative side, Mapsee-1 can be criticized both for descriptive and procedural inadequacies [11,12]. Some of these criticism are:

Descriptive inadequacies.

1. The cue/model structure has one level only.
2. A scene interpretation is a label in the domain of a variable. These labels have no internal structure.
3. Many real world scene domain constraints are represented poorly, if at all.

Procedural inadequacies.

1. The interpretation process is essentially data-driven.
2. The complete scene constraint graph has to be constructed before any model can constrain another.
3. Local control of recognition is not possible.
4. Alternative scene interpretations are not explicitly available to guide the program.

Mulder [30] has designed and implemented a program that interprets line sketches of houses. This program has a multi-level cue/model structure. As a result, more real world scene domain constraints can be represented. The multi-level cue/model structure allows for a distinction between edge types, surface orientations, surface interpretations, and objects. The perceptual cycle in this program is not closed. Interpretations at one level serve as cues for the next level up (Fig. 6). The program is an interesting illustration of how cue/model hierarchies can be implemented, but at the same time it continues to suffer from some of the Mapsee-1 maladies, in particular, the absence of a structure for describing the scene models and the possible relations between them. An altogether different approach for representing models was required. The result was Mapsee-2.

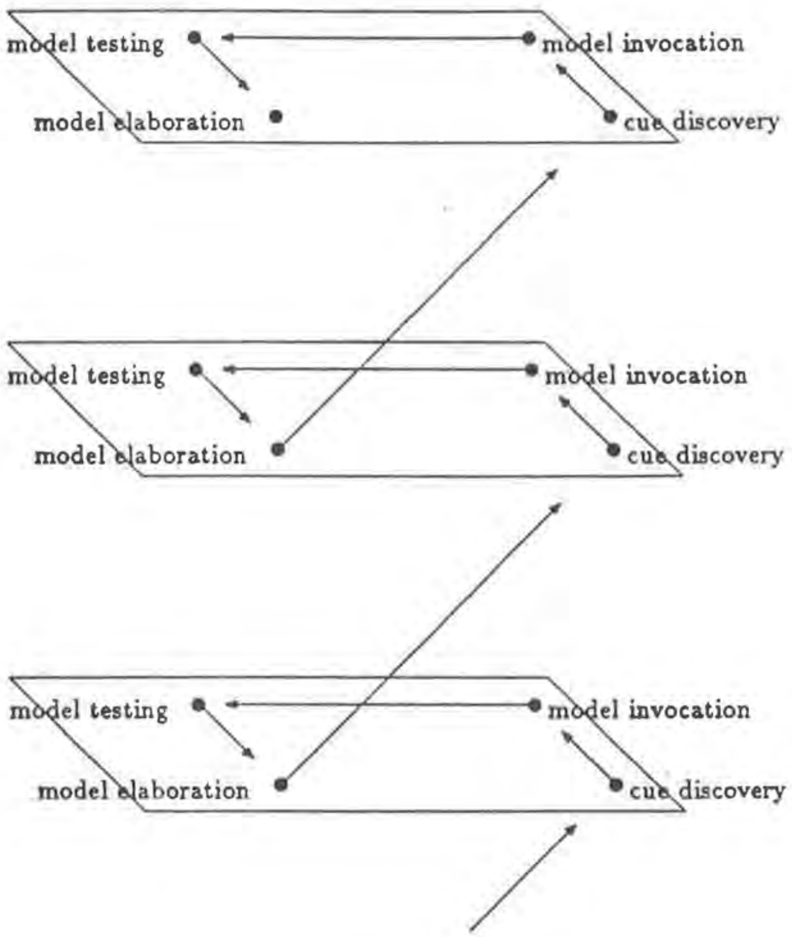

Figure 6: a multi-level cycle of perception

4 Mapsee-2

Mapsee-2 explores schemata [3] as a remedy for the descriptive and procedural inadequacies mentioned above. Schemata and related representations [4,10,17,29,37,38,39,40,41] are receiving considerable theoretical interest in Artificial Intelligence. Schema-based representations can be formalized. Havens [13] has proposed a schema labelling theory in which a schema represents a class by specifying its membership as a small number of subclasses that satisfy a set of constraints. The Mapsee-2 knowledge base is a partial reflection of this theory.

Mapsee-2's knowledge base is a collection of schema models. Each model, usually called a *class* represents a set of semantically related scene objects, either concrete physical objects or abstract configurations of other objects. The class provides a generic description for each of its members by specifying the legitimate relationships of the class with the other schemata in the knowledge base.

Each Mapsee-2 schema class has the following properties:

- A unique class name which is globally known in the knowledge base.
- A static *label set* for the class. Each *label* in the label set provides a symbolic name for a particular role that the schema can play in a global scene interpretation. The label set must be discrete, finite, and its labels known *a priori* for the class.
- A set of other schema classes from the knowledge base which constitute components and super-components for the class.
- A set of constraints defined over these component classes. Each constraint is a k-ary relation over the label set of the class and the label set of some subset of its components. The constraints restrict the role of a class as a function of the possible roles of its components.
- A set of procedures called *methods*. This property is an optional one. By means of a method a schema can assume local control of the search for a particular constraint among its component classes. Methods implement the concept of procedural attachment [47].

Figure 7 shows the properties of a geosystem, a generic interpretation for regions. A geosystem can be either a landmass, waterbody, lake, ocean, mainland, island, or it can have itself as label if none of the others are appropriate. Any geosystem with a mountain-range, road-system, or river-system as a component is constrained to be a landmass, mainland, or island. On the other hand, a geosystem with a coastline or lakeshore as component will take on different interpretations, depending on whether the coastline or lakeshore surrounds the geosystem (outer-shore), or forms a closed chain inside (inner-shore). No methods are shown for this schema class.

name:	geosystem
labelset:	{ geosystem, landmass, waterbody, island, mainland, lake, ocean }
components:	{ road-system, river-system, mountain-range, shore }
super-components:	{ world }

constraints	components	inner-shore	outer-shore
mountain-range	landmass, mainland, island		
road-system	landmass, mainland, island		
river-system	landmass, mainland, island		
shore		geosystem	geosystem
coastline		waterbody, lake, ocean	island
lakeshore		landmass, mainland, island	lake

Figure 7: properties of the schema class geosystem

4.1 Composition

Schema representations exploit composition to represent complex objects and their configurations. Primitive objects (chains and regions) depict low-level schema-classes. Complex scene objects are represented as compositions of simpler schemata. The resulting hierarchical structure forms a *composition hierarchy*. The recognition of a complex scene object is achieved by recursively recognizing its component parts so that the internal constraints of its schema remain satisfied. Figure 8 shows the Mapsee-2 composition hierarchy and the depiction relations which form the link between image and scene objects. Note, that a shore is always part of two different geosystems, the one it surrounds, and the one it is surrounded by.

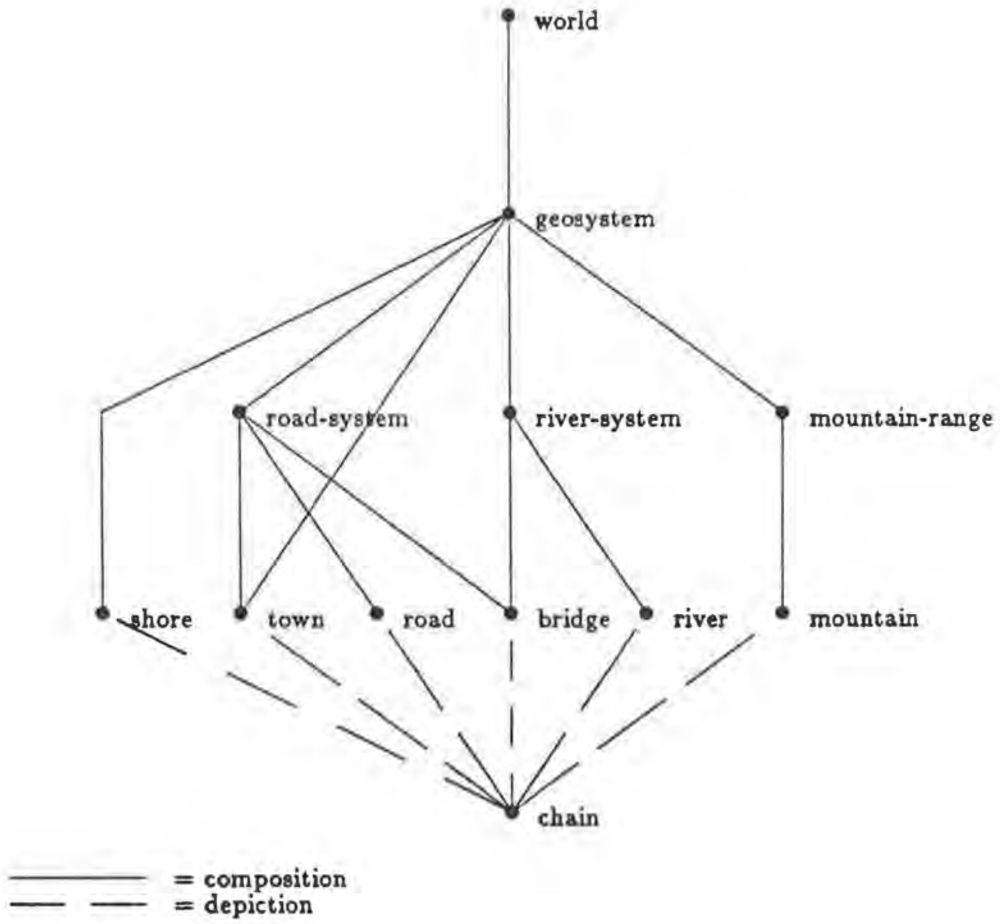

Figure 8: Mapsee-2 composition hierarchy

4.2 Specialization

For any schema label set that has more than one label there is a strict hierarchical organization which is used by a hierarchical constraint propagation algorithm to be discussed later. A label in such a hierarchy intensionally represents the labels that descend from it. Two schemata have non-trivial label hierarchies: *geosystem* and *shore*. Figure 9 illustrates these hierarchies. Hence, the *shore* label implicitly represents the fact that the current interpretation is either a *coastline* or *lakeshore*.

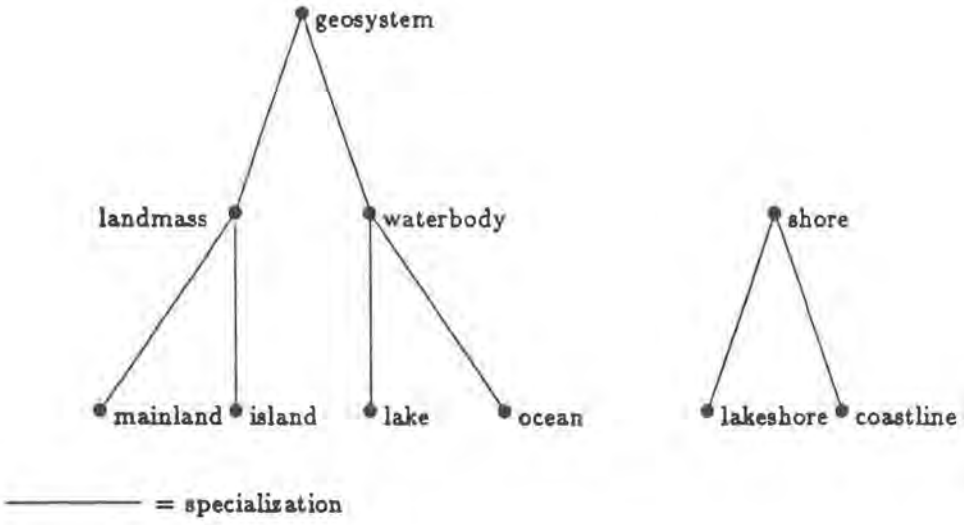

Figure 9: Mapsee-2 specialization hierarchies

The motivation for a hierarchical label organization is to curb the size of the label set of each schema. For instance, the label *geosystem* can represent the complete set of labels that descend from it.

```
class: { geosystem }
name: { geosystem-2 }
label set: { island }
super-components: { world-1 }
components: { mountain-range-1, shore-1 }
```

Figure 10 : a geosystem instance

4.3 Instantiation

When a schema is used to represent a particular scene object which is known or hypothesized to exist in the scene, then the class is used to generate a schema instance. For example, figure 10 shows an instance of the class *geosystem*. The instance, named *geosystem-2*, represents the island part of the Gambier Island sketch (Fig. 11). This island is an enlargement of one of the islands in Howe Sound (Fig. 1). Upon creation, a schema instance inherits the attributes of its parent class. Like their parent classes, schema instances are embedded in a composition hierarchy as well, with depiction relations to the image primitives that depict them. Figure 14 shows the network for the schema instances created for the Gambier Island sketch. This network constitutes the *scene constraint graph* for the particular image. The variables (nodes) are the schema instances, the label set of the instance constitutes the domain (shown in brackets in figure 14), and the edges represent the binary composition constraints.

4.4 Hypothesis trees

Figure 14 shows the final scene constraint graph (SCG). Each instance has one label only, and thus one interpretation. At the start of the interpretation process, however, the situation is quite different. Because of the ambiguity in many image cues, different schemata compete with each other in interpreting a particular primitive. Hence,

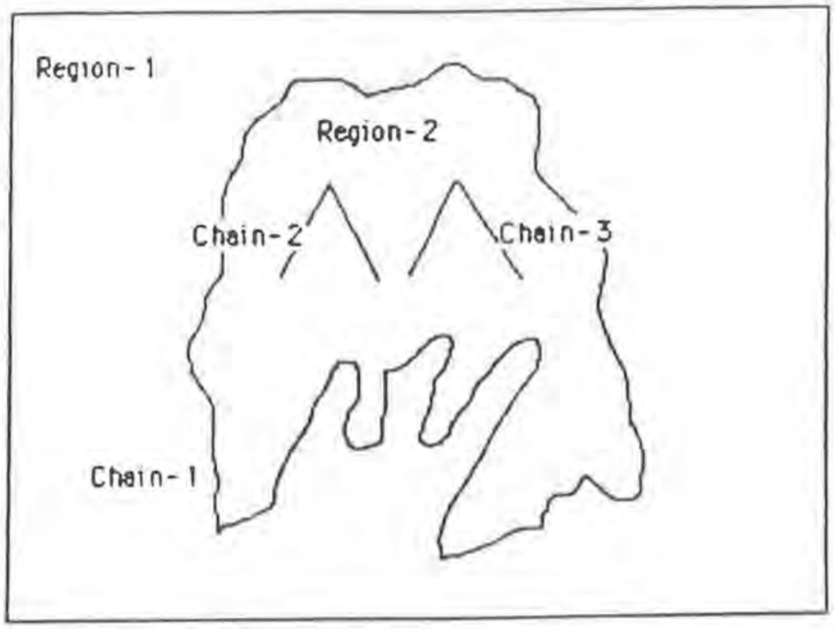

Figure 11: Gambier Island

instantiations of competing schemata have to be maintained as hypotheses. At any time, one has to be able to answer the question about competition between different hypotheses. At the same time higher level schemata are faced with the problem that some of their instances are hypothetical and some are not. Each schema instance is therefore organized as a *hypothesis tree*.

Each node in this tree represents the schema instance under different constraints. The root node represents the instance under its non-hypothetical constraints. Each of its descendants represents the instance under additional hypothetical constraints. The hypothesis tree is an explicit representation of the competition between different components. Each path from the root to a leaf in the tree represents a set of incremental and compatible constraints on the instance. Every time a schema instance obtains a new component (i.e. constraint), a successor node is created in the tree for each node in the hypothesis tree which is not in competition with the new component. If the new component is non-hypothetical then it is linked directly to the source node.

Any hypothetical components that are competing with a non-hypothetical component are subsequently removed from the tree.

Examples of hypothesis trees for the Gambier Island sketch are shown in figure 12 and 13. The situation in figure 12 exists after *chain-1* has been interpreted. *Chain-1* is a closed line segment interpretable as a *shore* only. This interpretation is ambiguous but not hypothetical. (That is, it must be a shoreline but the water could be inside or outside while the land is either outside or inside). The instance *shore-1* is therefore constrained at the root node only. At a higher level *chain-1* is represented by *geosystem-1* and *-2* representing the outer and inner region respectively. In accordance with the Mapsee-2 composition hierarchy (Fig. 8) both geosystems are part of the world. *Chain-2* is a mountain shaped chain. Such a feature allows for several hypothetical interpretations, two of which (*mountain* and *road*) are shown in figure 13. Hypothetical interpretations cannot be represented at the root node of an instance. Both *road-1* and *mountain-1* therefore create a new node (*1-1*) which represents the hypothetical constraint. These interpretations are represented in a similar way at the next level up by a road-system and mountain-range instance. Both hypothetical interpretations are a component of the geosystem which represents the inner-region. The two interpretations are incompatible. *Geosystem-2* must therefore absorb these constraints in two different branches, because every path from root to leaf must contain compatible constraints.

The SCG at this point is an exact representation of compatible interpretations. There are two global interpretations represented as *world-1-1* and *-1-2*. One global interpretation contains a shore and a road, the other a shore and a mountain.

The hypothesis trees collapse when *chain-3* is interpreted. The Mapsee-2 semantics state that two adjacent mountain shaped chains can be mountains only. This causes the *mountain-1* constraint to become non-hypothetical. Accordingly it moves up to the root node of the instance. The same happens at its super-components *mountain-range-1*, *geosystem-2*, and *world-1*. This structural change causes a disturbance in the integrity constraints of the trees with the result, that *world-1-1* and, recursively, all of its components are removed. This leads to the final situation illustrated in figure 14.

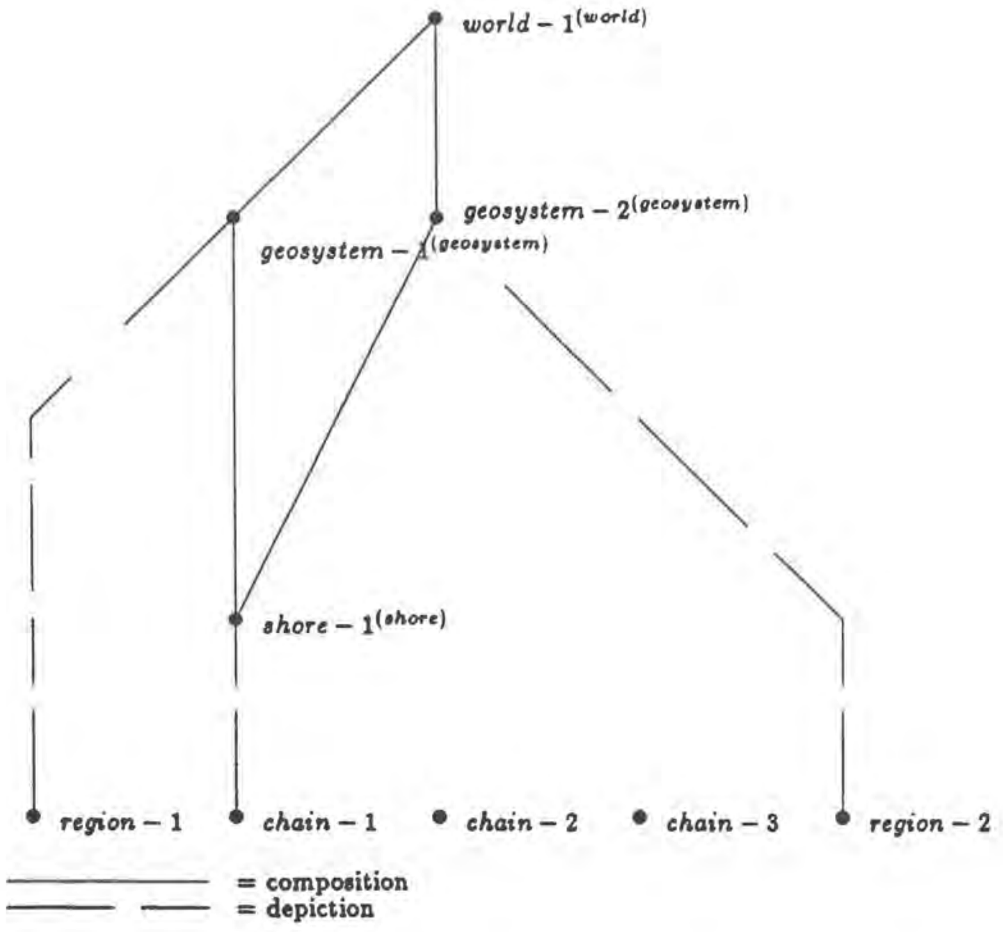

Figure 12: Gambier Island scene constraint graph after interpretation of chain-1. The current label of a schema instance is shown in parentheses.

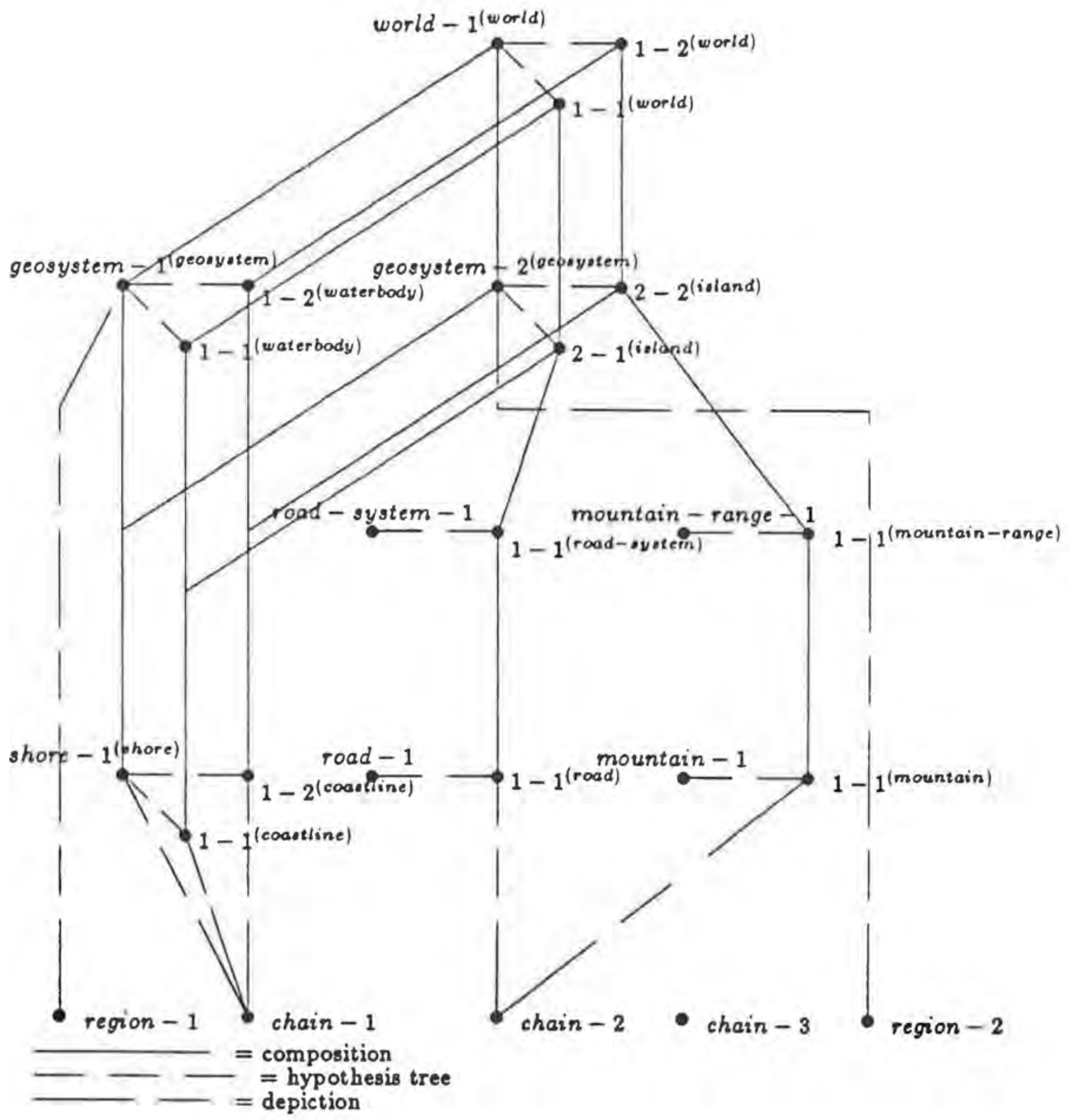

Figure 13: Gambier Island scene constraint graph after interpretation of chain-1 and chain-2

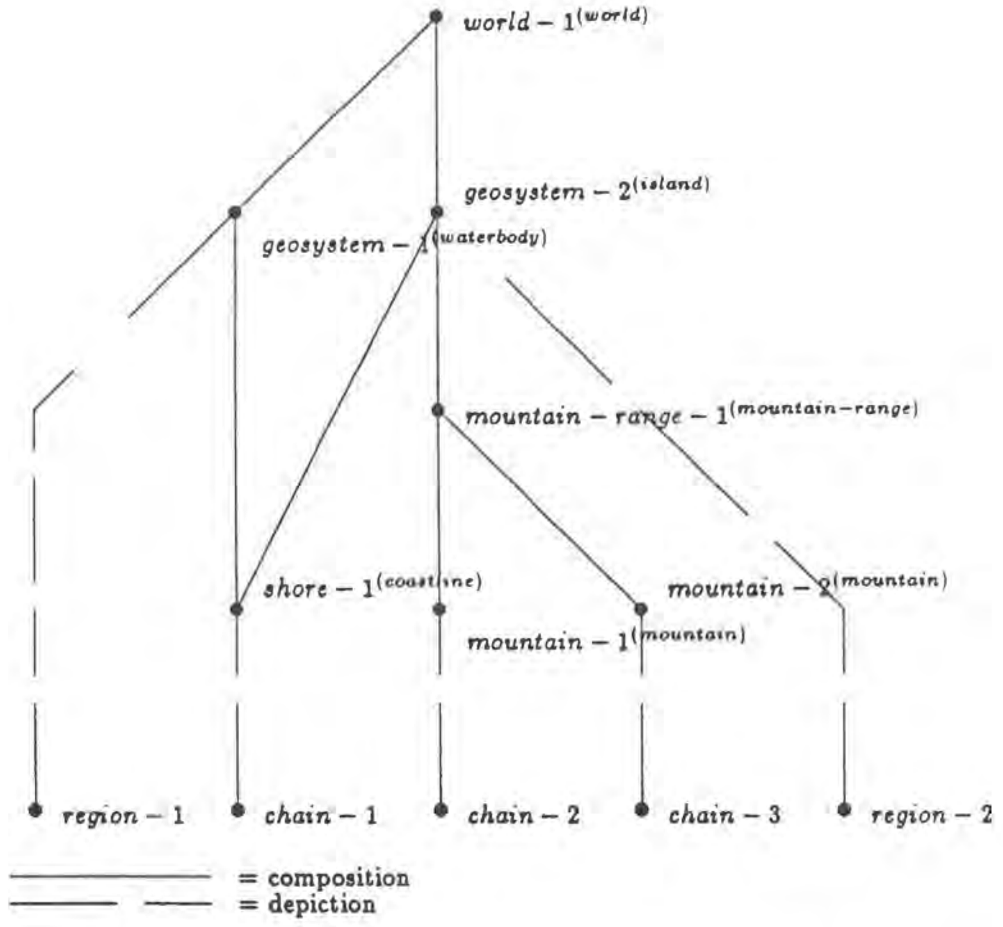

Figure 14: final scene constraint graph for Gambier Island

4.5 Hierarchical Constraint propagation

Mapsee-2 differs from Mapsee-1 in that the domain of each variable in the SCG has a hierarchical organization. All labels are embedded in a specialization hierarchy (Fig. 9). *AC-3* cannot handle such an organization. A new algorithm, *hierarchical arc consistency (HAC)* was therefore designed [25].

The essential difference between *AC-3* and *HAC* is that instead of the predicate P_{ij} used by *AC-3*, *HAC* uses two different predicates, $Pand_{ij}$ and Por_{ij}. For any arc (i,j) and a label pair (k,m) $(k \in D_i, m \in D_j)$ $Pand_{ij}$ is true iff all of k's successors in the specialization hierarchy are compatible with at least one of m's successors. Por_{ij}, on the other hand, is true iff at least one of k's successors is compatible with at least one of m's successors.

The *HAC* procedure is similar to the *AC-3* procedure except that REVISE is replaced by a new procedure: REVISE-HAC. The new procedure is used in both Mapsee-2 and Mapsee-3, and is shown below.

```
procedure REVISE-HAC ($(i, j)$)
1. begin
2. $DELETE \longleftarrow$ false
3. $Q_1 \longleftarrow D_i$
4. $ND_i \longleftarrow$ empty
5. while $Q_1$ not empty do
6.        begin
7.        select and delete any element $x$ from $Q_1$
8.        $Q_2 \longleftarrow D_j$
9.        $FOUND \longleftarrow$ false
10.       while $Q_2$ not empty and not $FOUND$ do
11.              begin
12.              select and delete any element $y$ from $Q_2$
13.              if $Pand_{ij}(x, y)$ then
14.                 begin
15.                 append $x$ to $ND_i$
16.                 $FOUND \longleftarrow$ true
17.                 end
18.              end
19.       if not $FOUND$ then
20.              begin
21.              $DELETE \longleftarrow$ true
22.              $Q_2 \longleftarrow D_j$
23.              while $Q_2$ not empty and not $FOUND$ do
24.                     begin
25.                     Select and delete any element $y$ from $Q_2$
26.                     if $Por_{ij}(x, y)$ then
27.                        begin
28.                        append all successors of $x$ to $Q_1$
29.                        $FOUND \longleftarrow$ true
30.                        end
31.                     end
32.              end
33.       end
34. $D_i \longleftarrow ND_i$
35. return $DELETE$
36. end
```

REVISE-HAC first tests $Pand_{ij}(k,m)$. If true then k is preserved and inserted into V_i's revised domain ND_i. If the test returns false then $Por_{ij}(k,m)$ is tested. If this test returns false as well then we know that none of k's successors is consistent with any of m's successors. Thus, we have to remove k from D_i. However, if $Por_{ij}(k,m)$ is true then we replace k by its immediate successors and we iteratively repeat the *Pand* and *Por* tests until for some descendant l of k $Pand_{ij}(l,m)$ is true.

$Pand_{ij}(k,m)$ must be provided for the leaves of the specialization hierarchy. From these predicates we can compile the *Pand* and *Por* predicates for each of the other nodes in the hierarchy [25].

HAC is still $O(ea^3)$. Better behavior, however, can be expected because the hierarchical domain organization causes the domain size of each variable to shrink. As well, if the label hierarchy forms a binary tree and if we assume that the domain size of each variable is always one than *HAC* is $O((e+3n/2)loga)$ [25].

An example of the operation of *HAC* can also be found in the Gambier Island example in figure 12 - 14. Upon creation each schema instance inherits the label set of its parent class. Thus, in figure 12 *shore-1* inherits the *shore* label (shown in parentheses), whereas both geosystem instances inherit the *geosystem* label. Very little can be concluded after interpreting *chain-1*. From the *geosystem* constraints in figure 15 one can infer that a geosystem with label set *geosystem* and an inner or outer-shore with label *shore* is constrained to be a geosystem. This situation changes after considering *chain-2*. In Mapsee-2, a geosystem with label set *geosystem* and a road-system, or mountain-range as a component must be a landmass. However, if the same geosystem instance (now with label set *landmass*) is also surrounded by a shore, then it must be an island. For this reason *geosystem-2-1* and *-2-2* have *island* as label after interpreting *chain-2*. Once all incorrect hypotheses have been rejected during the interpretation of *chain-3*, *geosystem-2* also receives the *island* label (Fig. 14). For geosystems surrounding a shore the following constraints hold. If the shore is a coastline then the geosystem becomes a waterbody. If the shore is a lakeshore then the geosystem is constrained to be a landmass. On the other hand, a shore whose inner-geosystem is a landmass, or island, or whose outer-geosystem is a waterbody, lake, or ocean, is constrained to be a coastline (Fig. 16). If the inner-geosystem is a waterbody or lake, or the outer-geosystem a landmass, mainland, or island, then the shore is constrained to be a lakeshore.

4.6 Constructing a scene constraint graph

HAC propagates constraints over the scene constraint graph, but the algorithm does not construct this graph. The graph is constructed by the processes of *completion* and *assembly*, which operate according to the control scheme illustrated in figure 17.

As is the case in Mapsee-1, Mapsee-2 uses primitive cues. These cues are simple shape features of the input chains such as free-ends, acute angles, closed chains and blobs. These features are used to create instances of schema classes which are leaves of the composition hierarchy (Fig. 8). Blobs, for instance, form a unique cue for a town, whereas a mountain-shaped chain can be either a road or mountain (Fig. 13).

The composition hierarchy represents mandatory relationships between schemata. Thus, any instance of a mountain schema must also be a component of a mountain-range instance, every river instance must be part of a river-system instance and so forth. It is the task of the *completion* process to ensure that these constraints are satisfied, thereby constructing a SCG. If the mountain instance, for example, has not yet been completed to a mountain-range instance, then the completion process will search for a suitable one. If one or more instances already exist, then the mountain will be completed to the instance that contains a mountain with which it forms a T-junction. If no such a mountain-range can be found, then the completion process will create a new mountain-range instance to which the mountain is connected.

Although *completion* searches for a suitable super-component, it will not actually

label set:	{geosystem}		
constraints	components	inner-shore	outer-shore
mountain-range	landmass		
road-system	landmass		
river-system	landmass		
shore		geosystem	geosystem
coastline		waterbody	island
lakeshore		landmass	lake
label set:	{landmass}		
constraints	components	inner-shore	outer-shore
mountain-range	landmass		
road-system	landmass		
river-system	landmass		
shore		landmass	island
coastline		incompatible	island
lakeshore		landmass	incompatible
label set:	{island}		
constraints	components	inner-shore	outer-shore
mountain-range	island		
road-system	island		
river-system	island		
shore		island	island
coastline		incompatible	island
lakeshore		island	incompatible

Figure 15: *geosystem* constraints

label set:	{waterbody}		
constraints	components	inner-shore	outer-shore
mountain-range	incompatible		
road-system	incompatible		
river-system	incompatible		
shore		waterbody	lake
coastline		waterbody	incompatible
lakeshore		incompatible	lake

Figure 15 (continued): *geosystem* constraints

establish links between the completing component and a potential super-component. First, a label compatibility test is required between the label of the component and the labels of the super-component. *HAC* is used here. It will test label compatibility and, if necessary, specialize the labels of the component. Once label compatibility is established, then the instances are linked by means of a component and part-of link. This is done by the *assembly* process.

If two schema instances are consistent over a part-of relation, then they are also consistent over the (reverse) composition relation. During the first pass *HAC* tests consistency over the part-of relation only. If this test fails, then Mapsee-2 halts without affecting the labels of the super-components. The super-component's labels are adjusted during the second invocation of *HAC* which takes place after *assembly* (Fig. 17). After this, the interpreter moves to another instance and starts *completion* again.

The interpretation process is cyclic. Each cycle consists of the completion of a single schema instance to an instance of its super-class in the composition hierarchy. Essentially we are using the composition hierarchy as a cue/model hierarchy.

Another effect of the cycle is an incremental construction of the SCG. Every time

label set:	{shore}	
constraints	inner-geosystem	outer-geosystem
geosystem	shore	shore
landmass	coastline	lakeshore
mainland	coastline	lakeshore
island	coastline	lakeshore
waterbody	lakeshore	coastline
lake	lakeshore	coastline
ocean	incompatible	coastline
label set:	{coastline}	
constraints	inner-geosystem	outer-geosystem
geosystem	coastline	coastline
landmass	coastline	incompatible
mainland	coastline	incompatible
island	coastline	incompatible
waterbody	incompatible	coastline
lake	incompatible	coastline
ocean	incompatible	coastline
label set:	{lakeshore}	
constraints	inner-geosystem	outer-geosystem
geosystem	lakeshore	lakeshore
landmass	incompatible	lakeshore
mainland	incompatible	lakeshore
island	incompatible	lakeshore
waterbody	lakeshore	incompatible
lake	lakeshore	incompatible
ocean	lakeshore	incompatible

Figure 16: *shore* constraints

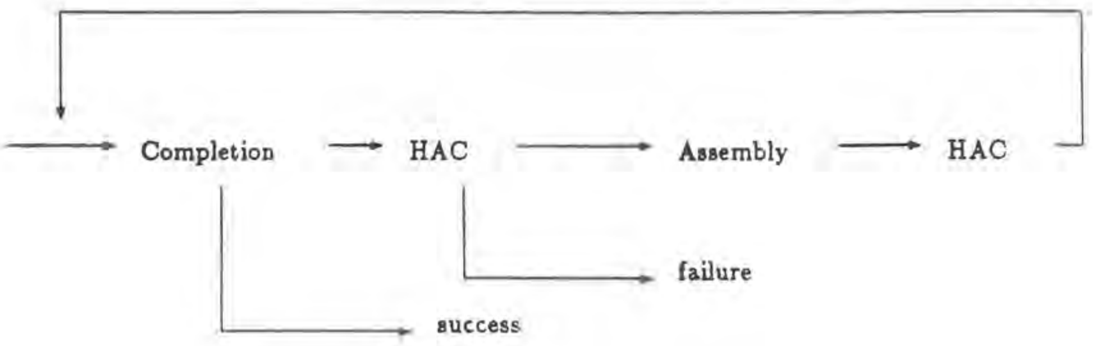

Figure 17: control cycle for Mapsee-2 and Mapsee-3

a new instance (node) is added to the graph, its labels are made consistent with the existing structure. In this way we allow modular interaction between the schema-based construction process and the *HAC*-based constraint propagation. Construction composes a scene interpretation while propagation specializes the interpretation.

Mapsee-2 operates in both a data-driven and model-driven manner. It takes one chain at a time and completes its interpretation all the way up to the world level (Fig. 8) before starting on the next chain. Once more, taking Fig. 11 as an example, we first interpret *chain-1* which is ambiguously interpreted as a shore. Because it is the only interpretation possible, it is non-hypothetical. The shore becomes a component of two geosystems (*regions* 1 and 2), both of which are part-of the world. Next, *chain-2* is pursued, causing the creation of two hypothetical interpretations (Fig. 13). These interpretations remain hypothetical until *chain-3* is interpreted, at which point the *road* interpretation is eliminated, leaving only one interpretation for each primitive (Fig. 14).

The model-driven aspect of the program resides in a schema's *methods*. These schema owned procedures are invoked during *completion*. Each method searches for particular constraints among its components. The mountain-range schema, for instance, can investigate whether a completing (hypothetical) mountain instance makes a Tee junction with another previously found mountain. As mentioned before, if this method succeeds then the mountain interpretation becomes non-hypothetical and all hypothesis trees involved are collapsed (Fig. 14).

4.7 Discussion

Mapsee-2 has successfully interpreted eleven sketch maps. Seven out of eleven examples represent real maps (e.g. the Lower Mainland of B.C. in Fig. 1), the others were made up. Mapsee-2 overcomes the inadequacies of Mapsee-1.

Descriptive adequacy:

1. The cue/model structure has many levels. The composition hierarchy serves as cue/model hierarchy. In this way we overcome the problem of relying on low level image features to invoke high-level object models directly [2].
2. Scene interpretations are represented as schema instances with internal structure and their own procedures for guiding the recognition process. For propagation purposes, on the other hand, each instance can still be treated as a label.
3. Real world scene domain constraints are improved. Scene constraints are distributed over schemata represented at multiple levels of composition and specialization.

Procedural adequacy:

1. The interpretation process combines a data-driven with a model-driven approach.
2. The SCG is constructed incrementally. Upon each addition of a new instance the graph is made consistent.
3. The schema's methods provide local control.
4. Alternative scene interpretations are maintained by means of hypothesis trees.

Despite these improvements Mapsee-2 has introduced new forms of descriptive and procedural inadequacy.

1. Scene objects are represented in a non-uniform way. Scene objects embedded in a composition hierarchy are all represented as schemata. However, most objects in a specialization hierarchy are not. Only the objects at the roots of the specialization hierarchy are schemata. All other nodes occur as labels in the schema at the root of the hierarchy. As one result *HAC* had to be implemented in a procedural form, because the constraints of labels in the specialization hierarchy are not explicitly represented in the composition hierarchy. What is desirable is to represent *all* scene objects as schemata.
2. Spatial relationships are not represented in the SCG. Methods search for spatial relations, and affect the completion process. The particular composition structure represented in the SCG is partially the result of the discovery of spatial relations.
3. Although hypothesis trees are a good way of representing hypothetical interpretations and competitiveness, their size can grow explosively. In an under-

constrained situation, the addition of a single hypothetical component to the hypothesis tree of a super-component can double the tree size of the latter. Thus, the number of nodes in the SCG is exponential in the number of primitives in the worst case. What is needed is a representation by means of which we can control the explosive effect of hypothetical interpretations.

All of these problems can be remedied. A solution is provided in the Mapsee-3 system.

5 Mapsee-3

The inadequacies of Mapsee-2 can be overcome. Like Mapsee-2, Mapsee-3 is a schema-based program, but there are two essential differences between Mapsee-2 and Mapsee-3.

1. *All* scene objects are represented as schemata in Mapsee-3. In addition, all relations between scene objects are represented as schemata.
2. Mapsee-3 uses discrimination graphs instead of hypothesis trees. In combination with *HAC* discrimination graphs allow us to represent hypotheses as alternate labels in the domain of a variable, thereby eliminating the need for a separate representation for hypotheses.

5.1 Uniform representation of objects and relations

All scene objects and relations in Mapsee-3 are represented as schemata. This allows us to use the internal structure of schemata if we are tracing the composition hierarchy. At the same time we can treat objects and relations as labels, if we are propagating constraints. Apart from providing uniformity of representation, the knowledge base has become much more declarative compared to that in Mapsee-2. A composition hierarchy now not only exists for schemata at the root of a specialization hierarchy. All other nodes are embedded in such a hierarchy as well. Figure 18 shows the Mapsee-3 composition hierarchy. We will call this the "natural" composition hierarchy for reasons to be clarified later. Figure 18 does not show any relation schemata.

As in Mapsee-2, spatial relations are created by a schema's methods. The explicit presence of spatial relations in both the composition hierarchy of schema classes and the SCG provides more descriptive adequacy. Additionally, spatial relations have positive effects for implementation of *HAC*. In Mapsee-2, the absence of schemata for objects in the specialization hierarchy and schemata for spatial relations forced an *ad hoc* procedural implementation. With all objects and relations explicitly represented in the SCG, this is no longer necessary. The Mapsee-3 implementation is an exact implementation of the *HAC* algorithm as specified in section 2.

5.2 Discrimination Graphs

The problem of the potential explosion of hypothetical interpretations has largely been ignored in Computational Vision. Yet, the problem is a serious one, even in the domain of sketch maps. Specialization hierarchies can somewhat alleviate the problem, because they allow us to replace some elementary labels in the domain of a variable by a smaller number of abstract labels. In Mapsee-2, for instance, a region can be either a lake, ocean, island, or mainland (Fig. 9). The presence of the specialization hierarchy allows us to replace these labels by one abstract label: *geosystem*. Although we can somewhat reduce the label set in this manner it does not really solve the problem. In many "real world" situations there is no specialization hierarchy that can cover the complete label set. For instance, a set of green pixels could be farmland, a golf course, a roof, a forest, or even astroturf in a stadium.

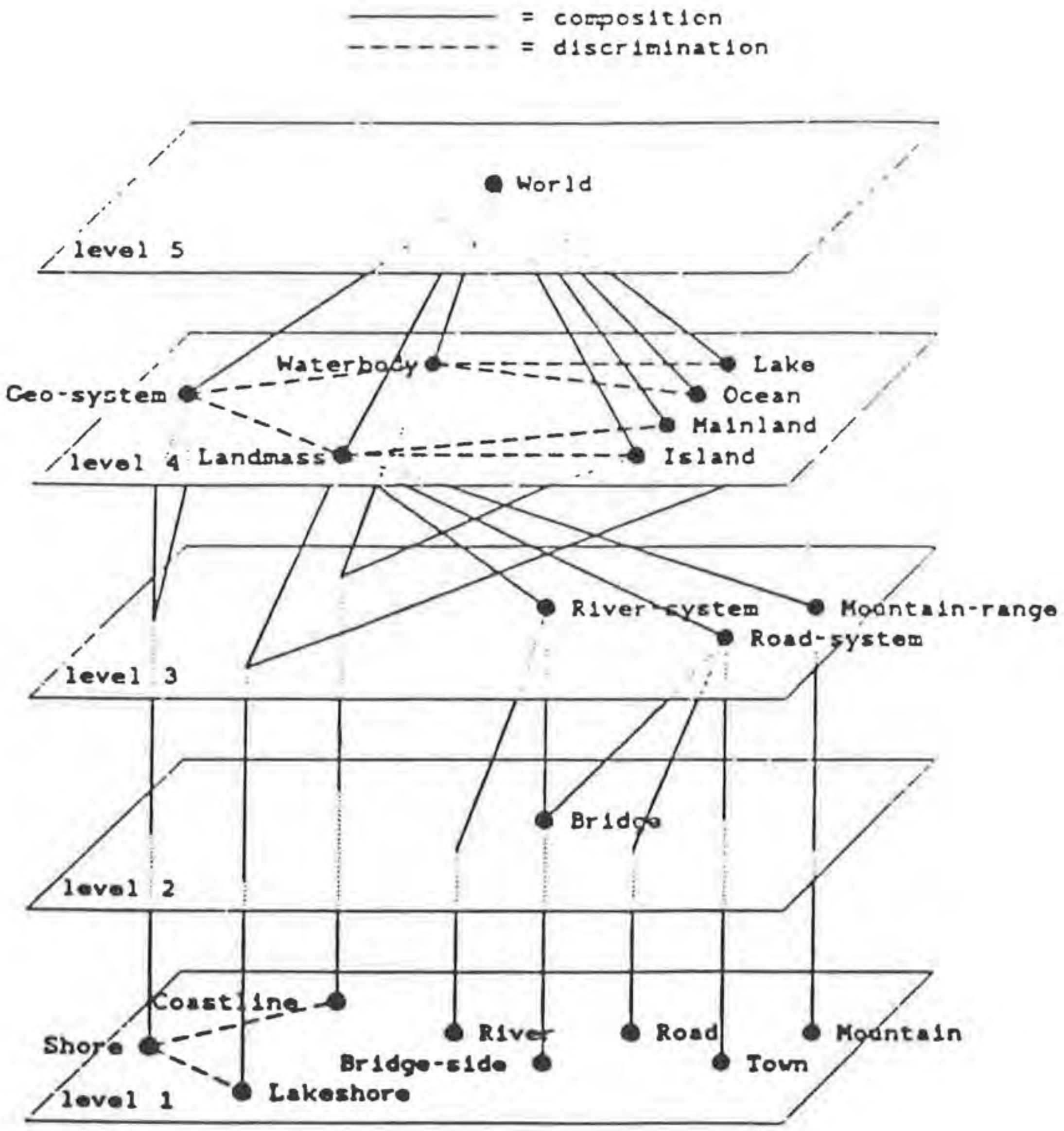

Figure 18: Mapsee-3 composition hierarchy

There is no natural specialization hierarchy to cover these labels. Discrimination graphs offer a solution.

Informally, discrimination graphs (DG's) can be understood as an extension of specialization hierarchies. Figure 19 shows an example. In Mapsee-3, a closed line segment depicts a road, coastline or lakeshore. *Coastline* and *lakeshore* can be naturally combined to be a shore, but that is where the natural abstraction capabilities stop. By means of DG's we extend the abstraction capability to represent closed line segments by one label only: *road/shore*. If a line segment is not closed, it can, among other things, be a road or river. A second extension from *road* and *river* creates the *road/river* label.

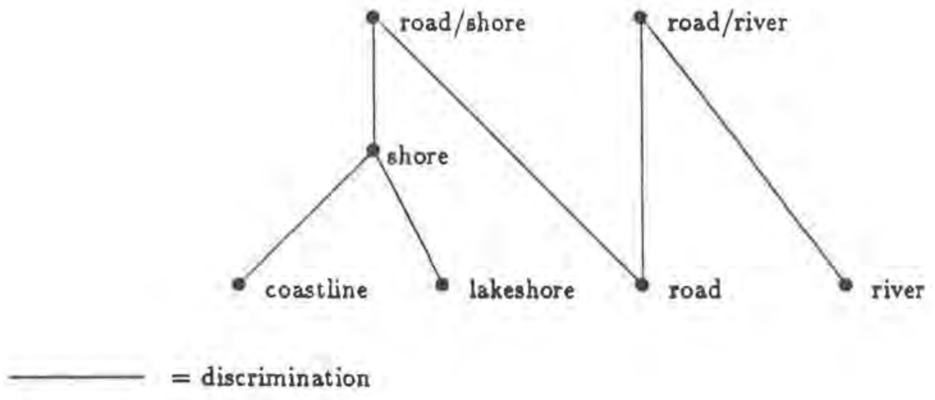

Figure 19: an example of a discrimination graph

More formally, the idea of DG's is based on the assumption that we can classify image features in one or more dimensions (e.g. shape, texture) the result of which is a finite number of categories for each dimension. As well, we assume that there are only a finite number of scene objects whose image appearance falls in a particular category. DG's are based on a categorization of object classes that belong to a particular image feature category. A DG is a directed acyclic graph. A root node of the graph is an abstract object class that intensionally represents all the elementary object classes that belong to a particular image feature category. The leaves of the graph are elementary object classes. Intermediate nodes represent subsets of the set represented by their ancestors. Elementary object classes can belong to more than one image feature category, as the road object in Fig. 19 illustrates. DG's can therefore become tangled hierarchies with multiple root nodes. Specialization hierarchies are strict trees but DG's are not.

All chains in Mapsee-3 are classified with respect to shape. Table 1 illustrates the shape categories used and the interpretations possible. A chain which closes upon itself is visibly closed. It is potentially closed when it runs off the frame on both sides. A chain which does not fit any of the other shape categories is classified as residual. These shape categories are mutually exclusive.

Mapsee-3 uses a composition hierarchy and different DG's. Each scene object is represented at five different levels of composition (Fig. 18). The DG's are constructed orthogonally to the composition hierarchy. Figure 20 shows the DG at the composition leaf level. At this level there is a unique (abstract) scene object for each shape category. For instance, the object *road/shore* intensionally represents all the objects depicted by the visible closure category. DG's such as the one shown in Fig. 20 can be automatically constructed [31,34].

With a unique classification for each chain we can represent this chain by means of the abstract object that uniquely represents this classification. Thus, at the start of the interpretation process each visibly closed chain will be interpreted as a *road/shore*, an interpretation which is ambiguous, but not hypothetical. Like all of its natural descendants, abstract objects such as *road/shore* are also embedded in an abstract composition hierarchy. Thus each abstract object is also represented at each level of composition. At each level of composition a DG exists that connects the abstract object with its natural descendants. Abstract composition hierarchies can also be

1	2	3	4
Road	Road	Road	Mountain
River	River	Coastline	Road
Mountain	Coastline	Lakeshore	River
Coastline	Lakeshore		
Lakeshore	Bridge-side		

5	6	7	8
Bridge-side	Town	Road	Road
Road		River	River
River			Coastline
			Lakeshore

1 = potential closure and mountain shape

2 = potential closure and bridge-side shape

3 = visible closure

4 = mountain shape

5 = bridge-side shape

6 = blob shape

7 = residual

8 = potential closure

Table 1: shape classifications for line segments

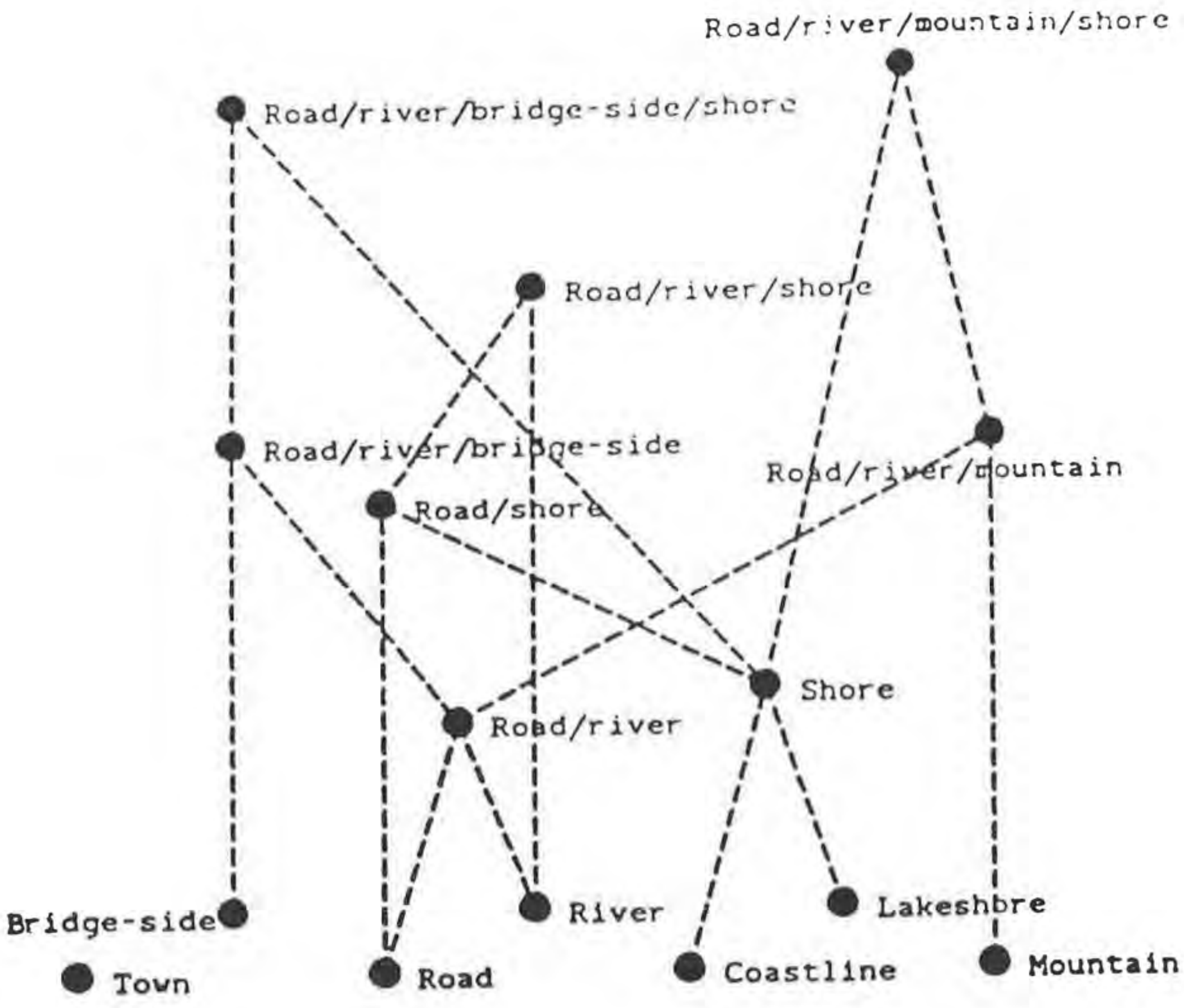

Figure 20: discrimination graph at the composition leaf level

automatically constructed [31].

Both natural and abstract scene objects are represented as schemata. In the Mapsee-3 SCG each variable continues to represent a schema instance. The current interpretation of the instance is represented by means of one or more abstract labels in the variable's domain. The edges in the SCG now represent composition or spatial constraints. Returning to the *road/shore* example, each closed chain is depicted by an instance of the *road/shore* schema. Upon instantiation this instance obtains its own name as label. At any time this label can be replaced by one of its descendants in the DG. For instance, once the region surrounded by the chain is constrained to be a waterbody, then the *road/shore* label must be refined to be a *lakeshore*. This is the case because a road must have land on both sides and a coastline must have water on the outside.

5.3 Constructing a Scene Constraint Graph

DG's eliminate the need for hypothesis trees. Competing interpretations are now represented by one or more labels in the domain of a single variable. Other than that the interpretation process in Mapsee-3 is similar to that of Mapsee-2. The control cycle (Fig. 17) is the same. Its implementation, however, is simpler, more modular, and declarative. The composition process, for one thing, no longer needs to deal with hypothesis trees. This results in a much simpler SCG. For example, figure 21 shows the Mapsee-3 SCG for the same situation as the Mapsee-2 SCG for the Gambier island example (Fig. 13).

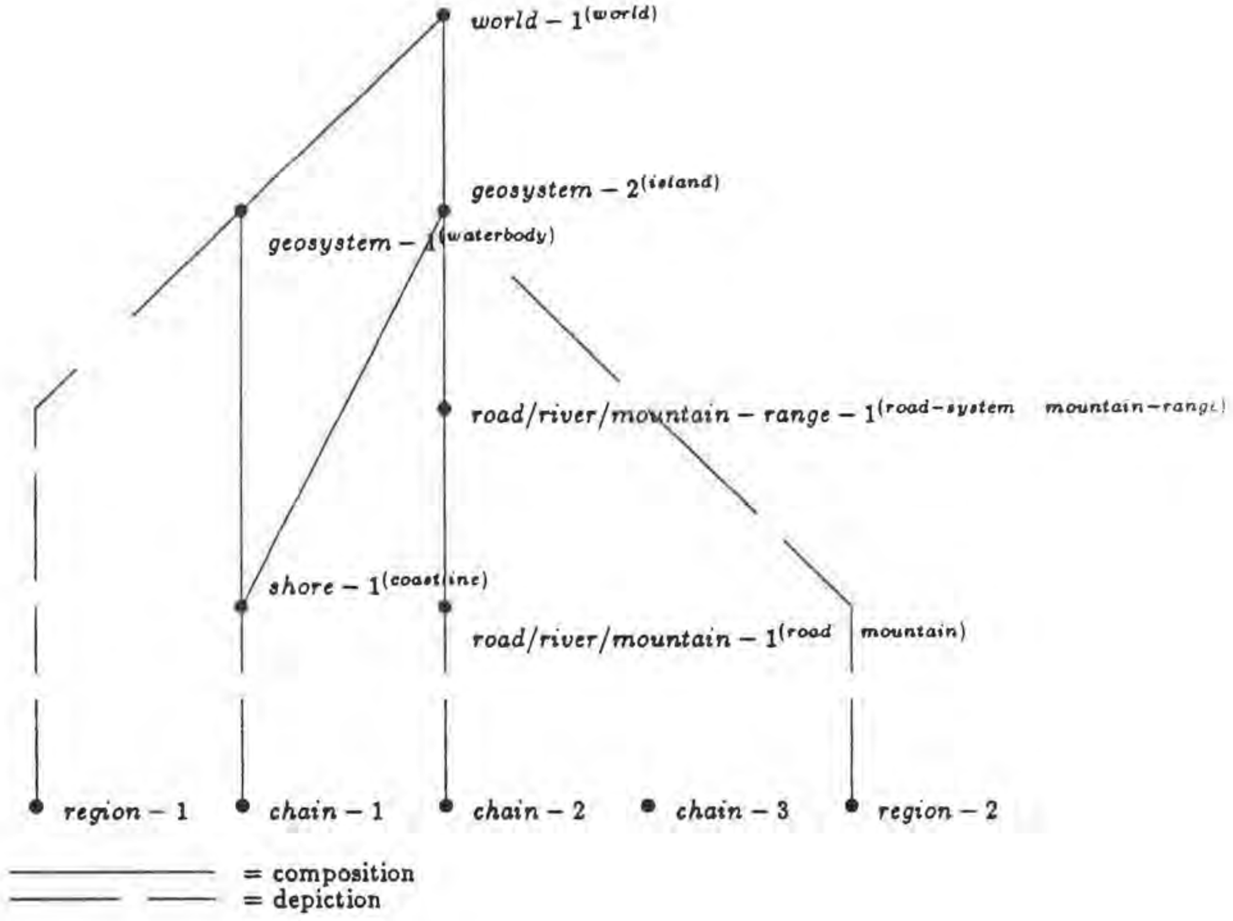

Figure 21: Mapsee-3 scene constraint graph after interpretation of chain-1 and chain-2

Increasing modularity is achieved because *completion* and *HAC* each control their own dimension. *Completion* controls the incremental construction of the SCG, *HAC*, on the other hand, controls the propagation of labels in the domain of each variable. Thus, elimination of competing hypotheses has now become a pure constraint propagation process. The replacement and elimination of competing labels rarely affect the structure of the SCG. In Mapsee-2 the structure of the SCG is always affected by changes in structure of hypothesis trees as a result of label propagation.

The *HAC* implementation is also more declarative then it could ever be in Mapsee-2. Most constraints (i.e. the predicate $Pand_{ij}$ in REVISE-HAC) are now explicitly represented in the combined abstract and natural composition hierarchy. The Por_{ij} predicates can be compiled from the $Pand_{ij}$ predicate [25].

5.4 Discussion

Mapsee-3 solves the problem of explosive growth of the hypothesis trees. Each chain is represented by one schema instance (variable) at the composition leaf level. The total number of variables created for each chain in the SCG has a fixed upper limit which largely depends on the local structure of the composition hierarchy. For example, a blob-shaped chain becomes a town instance at the composition leaf level. Town has a fixed number of super-components in its composition hierarchy (Fig. 18). The number of variables representing objects cannot exceed that limit. In addition, each scene object can be constrained by other objects through relations also represented in the SCG. However, two primitives in the image are generally tied together by at most one spatial relation. The number of nodes in the SCG is therefore linear in the number of image primitives [31].

Theoretically, the number of predicate tests in *HAC* is of $O(ea^3)$ in the worst case [25]. Experiments with Mapsee-3 show results better than that [31]. DG's tend to keep the domain size (a) of each variable very small. For this reason we can take a as a constant. Thus, *HAC* becomes linear in the number of edges. For planar graphs, *HAC* is also linear in the number of nodes in the graph [24]. For most cases the SCG in Mapsee-3 turns out to be planar. With an established linear relationship between image primitives and nodes in the SCG, it follows that there is also a linear (or, at worst quadratic) relationship between the number of predicate tests in *HAC* and the number of image primitives.

The model which underlies the Mapsee-3 design interprets image primitives by means of schema classes whose interpretation is at first extremely generic and relatively domain-independent. As more and more constraints are discovered in the image, this interpretation becomes more specific and domain-dependent. Because of this continuing process of interpretation refinement along a discrimination graph, this approach has been referred to as *Discrimination Vision* [31,33].

Mapsee-3 solves the inadequacies of Mapsee-2 identified earlier:

1. All scene objects are represented as schemata.
2. Spatial relations are also represented as schemata and accordingly appear as variables in the SCG.
3. The number of nodes in the SCG is linear in the number of image primitives. In Mapsee-2 this relationship was exponential.

6 Discussion

The representation of knowledge continues to be a key to progress in computational vision (and artificial intelligence, in general). A schema-based knowledge representation allows us to structure the knowledge base along different dimensions such as composition and specialization. A general-purpose vision system needs the capability to describe an image at different levels of detail and specificity. Composition and specialization hierarchies provide this capability. In addition, a schema-based knowledge representation allows us to treat objects as an atomic entity. This is necessary for constraint propagation purposes.

An expansion of specialization hierarchies into discrimination graphs, combined with a utilization of hierarchical arc consistency provides an approach to the representation and identification of visual knowledge which is modular, efficient, and effective.

Modularity exists in both representation and control. Discrimination graphs can be constructed orthogonally to the composition hierarchy. The completion process operates in the composition dimension only, whereas hierarchical arc consistency operates in the discrimination dimension. Similarly, *completion* and *assembly* construct the scene constraint graph, whereas hierarchical arc consistency propagates constraints over this graph.

This approach is also efficient. The relationship between the number of image primitives and the number of nodes in the scene constraint graph is linear. *HAC* is cubic in the domain size for the worst case. However, experiments with the Mapsee-3 program have shown much better results than predicted by the worst case analysis.

Discrimination graphs in combination with hierarchical arc consistency provide a new approach to image interpretation. The discrimination vision approach embodies an interpretation process in which the image is interpreted using local to global strategy, whereas image primitives depict interpretations which are at first abstract and unspecific. As interpretation progresses, these interpretations are gradually refined until they are concrete and domain-specific. A general-purpose vision system must be capable of transforming an image description in terms of significant features into a domain-specific description correctly, quickly, and flexibly. The combination of discrimination graphs with hierarchical arc consistency provides such a capability.

Discrimination graphs also offer some promise for bridging the gap between *Early Vision* and *Model-based Vision*. In the root nodes, discrimination graphs can represent knowledge that is relatively domain-independent and valid for a wide variety of scenes. At the leaves, on the other hand, the graphs represent knowledge that is highly domain-dependent.

No serious effort has been made in this paper to review related work in Computational Vision. This is not to say that such work does not exist. The Acronym system [5] uses a schema-based representation with composition and specialization hierarchies. Constraint propagation techniques are also utilized. The same holds for the Alven system [44], a system for left ventricular performance assessment from X-ray images. Neither of these systems, however, use discrimination graphs. Alven's notion of similarity links is related to discrimination graphs, but they are used for a different purpose.

Thus far the Mapsee project has focussed on the model-based aspect of visual knowledge representation, and has paid little attention to early vision. The segmentation process in the Mapsee programs is simple. One category of image primitives, the line segments, are provided in the input, and they are assumed correct. The other category, the regions, can be found in a fairly straightforward manner, once the line segments are known. The interaction between early vision and model-based vision is a subject of ongoing research. In particular, the question of how to deal with images that are (potentially) incorrectly segmented remains an open issue. Nevertheless, the success in integrating a schema-based knowledge representation with constraint satisfaction techniques has indicated its utility and the desirability of further exploring this method.

7 Conclusion

This paper has explored the utility of integrating a schema-based representation for visual knowledge with constraint satisfaction techniques. This integration has been studied in a progression of three sketch map interpretation programs. These programs have been evaluated by means of the criteria of descriptive and procedural adequacy. The evaluation indicates that a schema-based representation in combination with a hierarchical arc consistency algorithm constitutes a modular, efficient, and effective scheme for representing visual knowledge.

8 Acknowledgements

The research reported in this paper has been supported by the Natural Sciences and Engineering Research Council of Canada through operating grants A0948, A9281, and A5502, and the Canadian Institute for Advanced Research. Alan Mackworth is a Fellow of the Canadian Institute for Advanced Research.

References

[1] H. H. Baker and T. O. Binford, "Depth from Edge and Intensity based Stereo", *Proc. of the seventh Int. Joint Conference on Artificial Intelligence*, Vancouver, 1981, pp. 631-636.

[2] H. G. Barrow and J. M. Tenenbaum, "Recovering Intrinsic Scene Characteristics from Images", in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman (eds.), Academic Press, New York, 1978, pp. 3-26.

[3] F. C. Bartlett, *Remembering, A Study in Experimental and Social Psychology*, Cambridge University Press, Cambridge, 1932.

[4] D. G. Bobrow and T. Winograd, "An Overview of KRL, A Knowledge Representation Language", *Cognitive Science*, vol. 1, no 1, pp. 3-46, 1977.

[5] R. A. Brooks, "Symbolic Reasoning among 3-D Models and 2-D Images", *Artificial Intelligence*, vol 17, no 1-3, pp. 285-348, 1981.

[6] M. B. Clowes, "On Seeing Things", *Artificial Intelligence*, vol. 2, no 1, pp. 79-112, 1971.

[7] E. C. Freuder, "Synthesizing Constraint Expressions", *Communications ACM*, vol. 21, no 11, pp. 958-966, 1978.

[8] J. Glicksman, "Using Multiple Information Sources in a Computational Vision System", *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence,* Karlsruhe, 1983, pp. 1078-1080.

[9] W. E. L. Grimson, *From Images to Surfaces*, M.I.T. Press, Cambridge, Massachusetts, 1981.

[10] W. S. Havens, "A Procedural Model of Recognition for Machine Perception", TR-78-3, Computer Science Department, University of British Columbia, Vancouver, 1978.

[11] W. S. Havens and A. K. Mackworth, "Structuring Domain Knowledge for Visual Perception", *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence*, Vancouver, B.C., 1981, pp. 625-627.

[12] W. S. Havens and A. K. Mackworth, "Representing Knowledge of the Visual World", *IEEE Computer*, vol. 16, no 10, pp. 90-98, 1983.

[13] W. S. Havens, "A Theory of Schema Labelling", *Computational Intelligence*, vol. 1, no 3-4, pp. 127-139, 1985.

[14] B. Horn, "Obtaining Shape from Shading Information", in *The Psychology of Computer Vision*, P. H. Winston (ed.), McGraw-Hill, New York, 1975, pp. 115-156.

[15] D. A. Huffman, "Impossible Objects as Nonsense Sentences", in *Machine Intelligence*, vol. 6, B. Meltzer and D. Michie (eds.), American Elsevier, New York, 1971, pp. 295-323.

[16] T. Kanade, "Recovery of the Three-Dimensional Shape of an Object from a Single View", *Artificial Intelligence* vol. 17, pp. 409-460, 1981.

[17] H. Levesque and J. Mylopoulos, "A Procedural Semantics for Semantic Networks", in N. V. Findler (ed.), *Associative Networks*, Academic Press, New York, 1979, pp. 93-121.

[18] H. C. Longuet-Higgins and K. Prazdny, "The Interpretation of Moving Retinal Image", *Proc. of the Royal Society B*, 208, 1980, pp. 385-387.

[19] A. K. Mackworth, "Interpreting Pictures of Polyhedral Scenes", *Artificial Intelligence*, vol. 4, no 2, pp. 121-137, 1973.

[20] A. K. Mackworth, "Consistency in Networks of Relations", *Artificial Intelligence*, vol. 8, no 1, pp. 99-118, 1977.

[21] A. K. Mackworth, "On Reading Sketch Maps", *Proc. of the 5th Int. Joint Conf. on Artificial Intelligence*, Cambridge, 1977, pp. 598-606.

[22] A. K. Mackworth, "Vision Research Strategy: Black Magic, Metaphors, Mechanisms, Miniworlds, and Maps", in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman (eds.), Academic Press, New York, 1978, pp. 53-60.

[23] A. K. Mackworth, "Constraints, Descriptions and Domain Mappings in Computational Vision", in *Physical and Biological Processing of Images*, O. J. Braddick and A. C. Sleigh (eds.), Springer Verlag, Berlin, West Germany, 1983, pp. 33-40.

[24] A. K. Mackworth and E. C. Freuder, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems", *Artificial Intelligence*, vol. 25, pp. 65-74, 1985.

[25] A. K. Mackworth, J. A. Mulder, and W. S. Havens, "Hierarchical Arc Consistency: Exploiting Structured Domains in Constraint Satisfaction Problems", *Computational Intelligence*, vol. 1, no 3-4, pp. 118-126, 1985.

[26] A. K. Mackworth, "Constraint Satisfaction", in *Encyclopedia of Artificial Intelligence*, S. C. Shapiro (ed.), John Wiley and Sons Inc., New York, 1987, pp.

205-211.

[27] A. K. Mackworth, "Adequacy Criteria for Visual Knowledge Representation", Technical Report TR-87-4, Department of Computer Science, University of British Columbia, Vancouver, Canada, 1987. To appear in *Computational Processes in Human Vision*, Z. W. Pylyshyn (ed.), Ablex Press, Norwood, N.J. (in press)

[28] D. Marr, *Vision*, W. H. Freeman, San Francisco, CA, 1982.

[29] M. Minsky, "A Framework for Representing Knowledge", in *The Psychology of Computer Vision*, P. H. Winston (ed.), McGraw Hill, New York, 1975, pp. 211-277.

[30] J. A. Mulder and A. K. Mackworth, , "Using Multi-level Semantics to Understand Sketches of Houses and Other Polyhedral Objects", *Proc. of the 2nd national conference of the Can. society for Computational Studies of Intelligence (CSCSI)*, Toronto, 1978, pp. 244-253.

[31] J. A. Mulder, "Using Discrimination Graphs to Represent Visual Knowledge", TR-85-14, Department of Computer Science, University of British Columbia, Vancouver, 1985.

[32] J. A. Mulder, "Using Discrimination Graphs to Represent Visual Interpretations that are Hypothetical and Ambiguous", *Proc. of the 9th Int. Joint Conf. on Artificial Intelligence*, Los Angeles, 1985, pp. 905-907.

[33] J. A. Mulder, "Discrimination Vision", Technical Report 1987CS-3, Dalhousie University, Halifax, N.S., 1987.

[34] J. A. Mulder, "An Algorithm which Automatically Constructs Discrimination Graphs in a Visual Knowledge Base", *Proc. of the 10th Int. Joint Conf. on Artificial Intelligence*, Milan, 1987.

[35] R. Nevatia, *Machine Perception*, Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[36] L. G. Roberts, "Machine Perception of Three-Dimensional Solids", in *Optical and Electro-Optical Information Processing*, J. T. Tippett, D. Berkowitz, L. Clapp, C. Koester, and A. Vanderburgh (eds.), M.I.T. Press, Cambridge, Massachusetts, 1965, pp. 159-197.

[37] R. B. Roberts and I. P. Goldstein, "The FRL Manual", Memo 409, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1977.

[38] R. B. Roberts and I. P. Goldstein, "The FRL Primer", Memo 408, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1977.

[39] R. C. Shank, "Conceptual Dependency: A Theory of Natural Language Understanding", *Cognitive Psychology*, vol. 3, no 4, 1972.

[40] R. G. Smith and P. Friedland, "UNIT Package User's Guide", Technical Memorandum 80/L, DREA, Dartmouth N.S., Canada, 1980.

[41] M. J. Stefik, "An Examination of Frame-structured Representation Systems", *Proc. of the 6th Int. Joint Conf. on Artificial Intelligence*, Tokyo, 1979, pp. 845-852.

[42] P. H. Swain and S. M. Davis (eds.), *Remote Sensing: The Quantitative Approach*, McGraw-Hill, N.Y., 1978.

[43] J. M. Tenenbaum, M. A. Fischler, and H. C. Wolf, "A Scene Analysis Approach to Remote Sensing", Technical Report TN 173, AI center, S.R.I., 1978.

[44] J. K. Tsotsos, "Knowledge Organization and its Role in Representation and Interpretation for Time-varying data: the ALVEN System", *Computational Intelligence*, vol. 1, no 1, pp. 16-32, 1985.

[45] S. Ullman, *The Interpretation of Visual Motion*, M.I.T. Press, Cambridge, Massachusetts, 1979.

[46] D. Waltz, "Understanding Line Drawings of Scenes with Shadows", in *The Psychology of Computer Vision*, P. H. Winston (ed.), McGraw Hill, New York, 1975, pp. 19-91.

[47] T. Winograd, "Frame Representations and the Declarative/Procedural Controversy", in *Representation and Understanding: Studies in Cognitive Science*, D. G. Bobrow and A. Collins (eds.), Academic Press, New York, 1975, pp. 185-210.

[48] R. J. Woodham, "Photometric Method for Determining Surface Orientation from Multiple Images", *Optical Engineering*, vol. 19, no 1, pp. 139-144, 1980.