Knowledge Structuring and Constraint Satisfaction: The Mapsee Approach

JAN A. MULDER, ALAN K. MACKWORTH, AND WILLIAM S. HAVENS, MEMBER, IEEE

Abstract—**Schema-based representations for visual knowledge are integrated with constraint satisfaction techniques. 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.**

Index Terms—**Constraint satisfaction, discrimination graphs, hierarchical arc consistency, model-based vision, recognition, schema representations, sketch maps.**

I. INTRODUCTION

THIS paper describes 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 [25].

The prospect of a general-purpose vision system remains a goal for the not-so-near future. Such a 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 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*) [13].

Research in computational vision has successfully exploited assumptions about surface descriptions, scene illumination and surface continuity, viewing position, lighting position, surface reflectance, motion, surface topology in scene analysis, and surface orientation [13]. Unfortunately, these constraints, while apparently necessary for producing scene descriptions, are frequently embedded in the particular theory being implemented and not explicitly represented. 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) [18]. Since, in its full generality, CSP is NP-hard the existence of algorithms that solve this problem in less than exponential time

Manuscript received November 2, 1987; revised August 3, 1988. This work was supported by the Natural Sciences and Engineering Research Council of Canada under Grants A0948, A9281, and A5502, and by the Canadian Institute for Advanced Research.

J. A. Mulder is with the Department of Mathematics, Statistics, and Computing Science, Dalhousie University, Halifax, N.S. B3H 3J5, Canada.

A. K. Mackworth is a fellow of the Canadian Institute for Advanced Research and is with the Department of Computer Science, University of British Columbia, Vancouver, B.C. V6T 1W5, Canada.

W. S. Havens is with Tektronix Research Laboratories, MS-50-662, P.O. Box 500, Beaverton, OR 97077.

IEEE Log Number 8823851.

0162-8828/88/1100-0866$01.00 © 1988 IEEE

is unlikely. Depth-first backtracking, for instance, is of $O(ea^n)$ in its worst case performance [22]. 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 [24]. In particular, since the vision CSP is NP-hard various polynomial constraint satisfaction *approximation* algorithms, also referred to as *network consistency* algorithms, are attractive. In the next section we briefly discuss this approach.

This paper focusses on the integration of constraint satisfaction techniques with schema-based representations. We discuss this interaction in a progression of three sketch map interpretation programs. 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. The discussion of sketch map interpretation programs is linked to related work in computational vision, but no effort has been made to provide an in-depth overview of such work. For recent reviews of research in model-based vision, see Binford [4] and Tsotsos [42].

Sketch maps capture in a simple form fundamental problems of representing and applying knowledge. Some problems of segmenting and interpreting real imagery are isolated while others are ignored. Furthermore, techniques for understanding sketch maps have application in interpreting real imagery [9], [40], [6]. 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.

II. Constraint Satisfaction

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 *network consistency* 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. Network consistency algorithms evolved from research in the polyhedral domain [43], [18], but have a much wider applicability [24]. Most of these algorithms are characterized by the fact that they test constraints in a local neighborhood only. Node consistency [18], for example, tests for unary constraints only, arc consistency [18], [22] tests unary and binary constraints, whereas k-consistency [8] 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 network consistency 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* [18] illustrates this operation which is characteristic for network consistency algorithms. Various derivatives of this algorithm were used in the Mapsee programs to be described later. The description of *AC-3* provided here is an informal one. For a more formal treatment of *AC-3*, see [18] and [24].

AC-3 enforces node and arc consistency in a constraint graph. Node consistency is established by defining a unary predicate P_i which tests $P_i(k)$ for each label k in the domain D_i of each vertex i in the graph. Inconsistent labels are removed from D_i. After the establishment of node consistency all arcs A_{ij} are stored on a queue. Arc consistency is tested for each arc A_{ij} on this queue. A_{ij} is consistent if for each label k in D_i there is at least one label m in D_j for which $P_{ij}(k, m)$ is true. Any k that is inconsistent is removed from D_i. If A_{ij} was inconsistent, then all arcs pointing at vertex i (except A_{ji}) are merged with the queue.

AC-3 does not maintain a record of label compatibility. The consistency test is terminated at the first m in D_j that is consistent with k. 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 [22]. 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 preprocessor 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 network consistency 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. Effective constraint satisfaction, however, is not the only problem model-based vision systems face. The construction of a constraint graph over which the constraints are propagated is a problem in itself. We will see that combining a schema-based representation with constraint

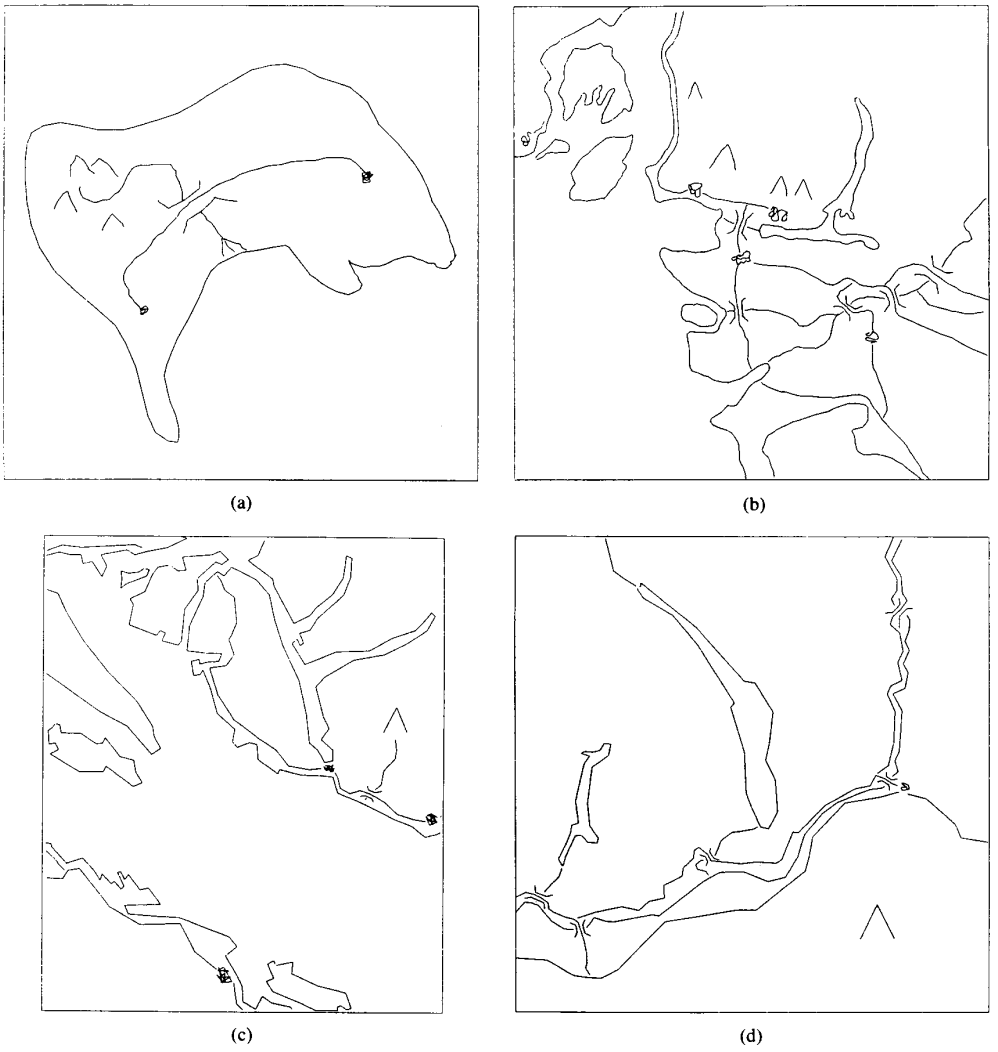


Fig. 1. Four sketch maps. (a) Porpoise Island. (b) Lower mainland of British Columbia. (c) Georgia Strait, British Columbia. (d) Fraser Valley, British Columbia.

propagation techniques can provide a modular and efficient integration of graph construction and constraint satisfaction. 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 in which we study and evaluate various schemes for representing 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 full description of Mapsee-1 appeared in [19]. 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 [12], [13]. Some aspects of the Mapsee-3 program have also been reported elsewhere [29]–[32].

III. 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 [Fig. 1(a)], or, as is the case in the Figs. 1(b)–(d), they can be created by tracing over 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 in interpretation. Line segments, for instance, can be roads, rivers, shores, bridges, town,

and mountains, whereas the "empty" regions in between can be land or water.

All Mapsee programs go through a more or less uniform segmentation stage first. The objective of segmentation is the construction of primitives and features. The primitives are the elements in the image. They are considered to represent a scene object or parts of it. The features are the image properties that constrain the possible interpretations for the primitives of which they are a part. Features also serve as cues for interpretations. Regions and line segments (called *chains*) are the primitives in Mapsee. Junctions, such as Tee, L, and Free-ends constitute the features. Each feature constrains the interpretation 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 sea, and the other two land. Another possibility is a mountain interpretation for both the stem and the bar, in which case all the regions must be land.

The chains are given as input. The junctions are computed from the chains, either during segmentation (Mapsee-1), or during interpretation when required by scene constraints (Mapsee-2 and -3). Regions are created by subdividing and connecting empty spaces by means of a quadtree representation. Their formation is guided by a conservative segmentation process. That is, subdivision of empty spaces stops well before any "leakage" through (nonintended) gaps between chains occur.

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. A constraint satisfaction algorithm, called *network consistency* (*NC*) closely interacts with this table. This interaction takes the form of a cycle of perception [20]. 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. The cycle of perception is a useful metaphor for describing and comparing control structures in computer vision programs [20], [42].

NC is a generalization of *AC-3*. *NC* goes beyond *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 graph solutions. 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 small number of sketches tested, the domain splitting operation did not need to be invoked.

A. 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 as follows [12], [13].

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 [28] has designed and implemented a program that interprets line sketches of houses. This program has a multilevel cue/model structure. As a result, more real world scene domain constraints can be represented. The multilevel 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. Model elaboration at one level provides the cues for the next level up. 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.

IV. Mapsee-2

Mapsee-2 explores schemata [3] as a remedy for the descriptive and procedural inadequacies mentioned above. Schemata and related representations [5], [11], [27], [34], [37], [38] have received considerable theoretical interest in artificial intelligence. Schema-based representations can be formalized. Havens [14] described a schema labeling theory in which a schema represents a class by specifying its membership as a small number of subclasses that sat-

isfy a set of constraints. The constraints are based on two orthogonal relationships in knowledge organization: composition and specialization. Complex objects have internal structure which can be represented as composition constraints among their parts. Likewise, specialization constraints segment classes into subclasses by type. The Mapsee-2 knowledge base is a partial implementation of this theory.

Mapsee-2 is but one example of the use of schema-based representations in model-based vision. Other examples are: Visions [10], [44], Acronym [7], Alven [41], Sigma [26], Goldie [16], and systems designed by Levine [17] and Ballard *et al.* [1].

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 supercomponents for the class.
- A set of constraints defined over component classes. Each constraint is a k-ary relation over the label set of the class and the label sets of some 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 optional. By means of a method a schema can assume local control of the search for an instance of a constraint among its component classes. Methods implement procedural attachment [45].

Fig. 2 shows the properties of a geosystem, a generic interpretation for regions. A geosystem can be either a landmass, waterbody, lake, ocean, mainland, or island. Component constraints are represented by a set of tuples. The first element in the tuple is a component of *geosystem*, the second element is the label to which *geosystem* is constrained by this component. 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 constraint), or forms a closed chain inside (inner-shore constraint). 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 }
Component constraints:	{(mountain-range landmass)(mountain-range mainland) (mountain-range island)(road-system landmass)(road-system mainland) (road-system island)(river-system landmass)(river-system mainland) (river-system island)}
Inner-shore constraints:	{(shore geosystem)(coastline waterbody)(coastline lake) (coastline ocean)(lakeshore landmass)(lakeshore mainland) (lakeshore island)}
Outer-shore constraints:	{(shore geosystem)(coastline island)(lakeshore lake)}

Fig. 2. Properties of the schema class geosystem.

A. Composition

Schema representations exploit composition to represent complex objects and their configurations. Primitive image objects (chains and regions) depict low-level scene objects. Complex scene objects are represented as compositions of simpler schemata. The resulting hiearchical 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. Fig. 3 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.

B. Specialization

For any schema label set that has more than one label there is a strict hierarchical organization which is used by a hierarchical network consistency algorithm to be discussed later. A label in such a hierarchy intensionally represents the labels that descend from it. In Mapsee-2 two schema classes have nontrivial label hierarchies: *geosystem* and *shore*. Fig. 4 illustrates these hierarchies. Hence, the *shore* label implicitly represents the fact that the current interpretation is either a *coastline* or *lakeshore*.

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. Several well known vision systems such as Alven [41], Acronym [7], and Visions [10], [44] have used composition and specialization hierarchies for similar reasons. In Mapsee-2 composition and specialization hierarchies are limited to the scene domain. A hierarchical organization of models can also be useful in the image domain, for example, solving problems of spatial scale [39]. Other researchers have used hierarchical model organizations that cross the image/scene boundary, e.g., [2], [33].

C. 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, Fig. 5 shows an instance of the class *geosystem*. The instance, named *geosystem-2*, represents the island part of the Gambier Island sketch (Fig. 6). This island is an enlargement of one of the islands in Howe

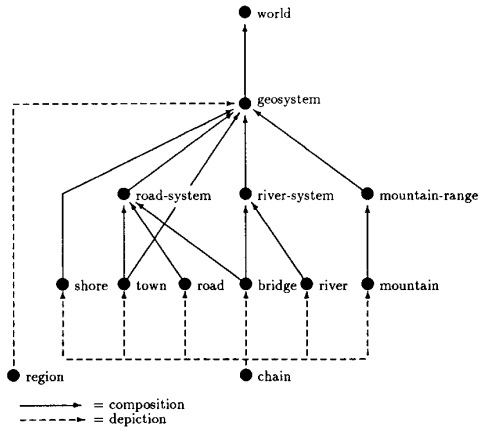


Fig. 3. Mapsee-2 composition hierarchy and depiction relations.

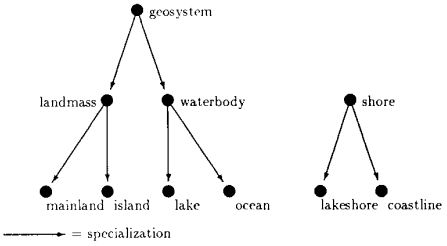


Fig. 4. Mapsee-2 specialization hierarchies.

Class:	geosystem
Name:	geosystem-2
Label set:	{ island }
Super-components:	{ world-1 }
Components:	{ mountain-range-1, shore-1 }
Component constraints:	{(mountain-range-1 geosystem-1)}
Outer-shore constraints:	{(shore-1 geosystem-1)}

Fig. 5. A geosystem instance.

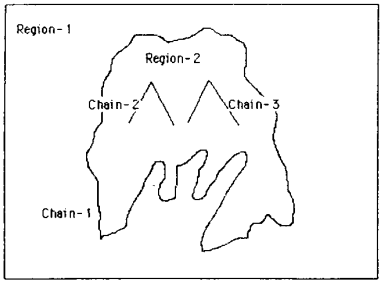


Fig. 6. Gambier Island.

Sound [Fig. 1(b)]. 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. Fig. 9 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 the brackets in Fig. 9), and the edges represent the binary composition constraints.

D. Hypothesis Trees

Fig. 9 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, instantiations of competing schemata have to be maintained as hypotheses. One has to be able to answer the question about competition between different hypotheses. 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*, which is defined as follows.

Each node in this tree represents the same schema instance under different constraints. The root node represents the instance under its nonhypothetical 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, 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 nonhypothetical, then it is linked directly to the source node. Any hypothetical components that are competing with a nonhypothetical component are subsequently removed from the tree.

Examples of hypothesis trees for the Gambier Island sketch are shown in Figs. 7 and 8. In all figures containing scene constraint graphs arcs have been replaced by edges. Each edge hereby represents two arcs, one in each direction. The situation in Fig. 7 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 *geosystem-2* representing the outer and inner region respectively. In accordance with the Mapsee-2 composition hierarchy (Fig. 3) 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 Fig. 8. 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 geosys-

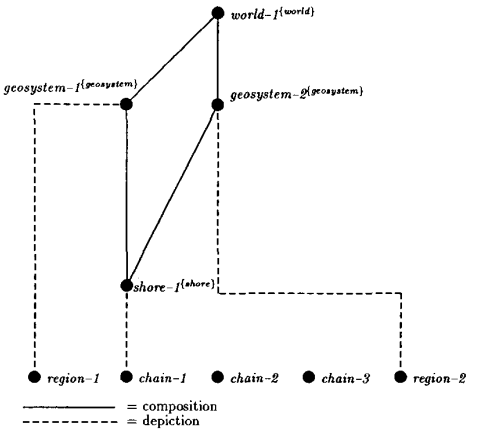


Fig. 7. Gambier Island scene constraint graph after interpretation of chain-1.

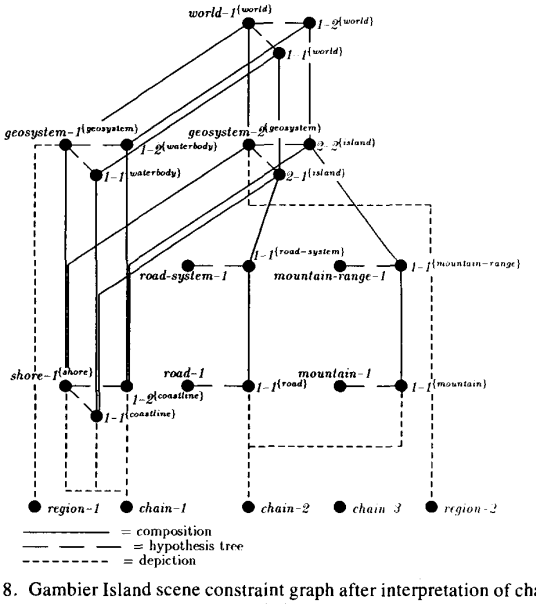


Fig. 8. Gambier Island scene constraint graph after interpretation of chain-1 and chain-2.

tem 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 *world-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 nonhypothetical. Accordingly the constraint moves up to the root node of the instance. The same happens at its super-components *mountain-range-1*, *geosystem-2*, and *world-1*.

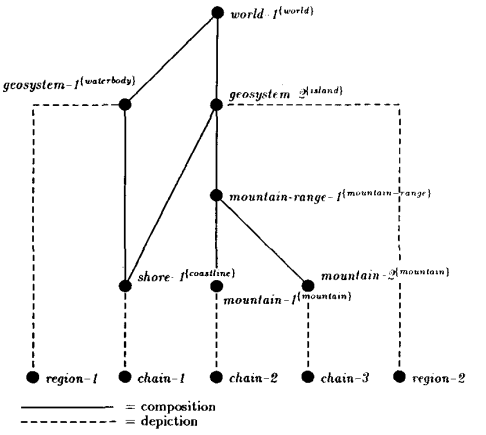


Fig. 9. Final scene constraint graph for Gambier Island.

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 Fig. 9.

E. Hierarchical Arc Consistency

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. 4). The label *geosystem*, for instance, now implicitly represents the fact that any of its descendants (mainland, island, ocean, lake) could be a consistent label. *AC-3* cannot handle such a situation. A new algorithm *hierarchical arc consistency* (*HAC*) was therefore designed. The discussion of *HAC* provided here is very abbreviated and informal. For a formal extensive treatment see [23].

HAC and *AC-3* are analogous except for one key difference: the predicate $P_{ij}(k, m)$ has been replaced by two new predicates $Pand_{ij}(k, m)$ and $Por_{ij}(k, m)$. $Pand_{ij}(k, m)$ on the labels k and m in D_i and D_j is a hierarchical version of $P_{ij}(k, m)$. $Pand_{ij}(k, m)$ is true, if *all* of k's descendants in the specialization hierarchy are consistent with *at least one* of m's descendants. If k is not consistent, then it is replaced by its descendants and the consistency test is repeated for the descendants. If k is inconsistent and has no descendants either, then it is removed altogether.

$Por_{ij}(k, m)$, on the other hand, is true if *at least one of* k's descendants is consistent with at least one of m's descendants. The *Por* predicate serves to increase the efficiency of the algorithm, because it allows for the elimination of subtrees of inconsistent labels without having to search through these subtrees.

Yet, the time complexity of *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 it is the case that the domain of each variable can always be covered by a single node of the tree, then *HAC* is $O((e + 3n/2) \log a)$, a remarkable improvement [23].

An example of the operation of *HAC* can also be found in the Gambier Island example in Figs. 7–9. Upon creation each schema instance inherits the label set of its parent class. Thus, in Fig. 7 *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*. As mentioned before, constraints are k-ary relations over the label set of a class and the label set of some subsets of its components. As one example, the constraints of a geosystem class with label *geosystem* were illustrated in Fig. 2. From the *geosystem* constraints 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 *geosystem-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. 9). 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. 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.

F. 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 flow diagram in Fig. 10. This control cycle is entered after the image has been completely segmented.

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. 3). Blobs, for instance, form a unique cue for a town, whereas a mountain-shaped chain can be either a road or mountain (Fig. 8).

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.

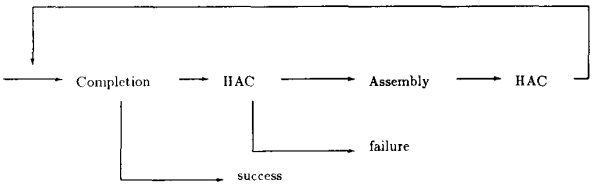


Fig. 10. Control cycle for Mapsee-2 and Mapsee-3.

Although *completion* searches for a suitable super-component, it will not actually establish links between the completing component and a potential supercomponent. First, a label compatibility test is required between the label of the component and the labels of the supercomponent. *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 supercomponents. The supercomponent's labels are adjusted during the second invocation of *HAC* which takes place after *assembly* (Fig. 10). 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 superclass 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 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. 3) before starting on the next chain. Once more, taking Fig. 6 as an example, we first interpret *chain-1* which is ambiguously interpreted as a shore. Because it is the only interpretation possible, it is nonhypothetical. The shore becomes a component of two geosystems (depicted by *regions* 1 and 2), both of which are part-of the world. Next, *chain-2* is pursued, causing the creation of two hypothet-

ical interpretations (Fig. 8). 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. 9).

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 nonhypothetical and all hypothesis trees involved are collapsed (Fig. 9).

G. Discussion

Mapsee-2 has successfully interpreted eleven sketch maps. Seven out of eleven examples were traced over topographic maps. Mapsee-2 overcomes some of 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 as discussed in [2].

2) Scene interpretations are represented as schema instances with internal structure. 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 for guiding the recognition process.

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 nonuniform 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 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 underconstrained situation, the addition of a single hypothetical component to the hypothesis tree of a supercomponent 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.

These problems can be remedied; some solutions are provided in the Mapsee-3 system.

V. Mapsee-3

Mapsee-3, like Mapsee-2, is a schema-based program, but there are two essential differences between them.

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.

A. 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 to trace the composition hierarchy. Objects and relations are also treated as labels, when we are propagating constraints. Apart from providing uniformity of representation, the knowledge base has a more declarative structure compared to that of Mapsee-2. Schemata at all levels of specialization are now embedded in a composition hierarchy. Fig. 11 shows the Mapsee-3 composition hierarchy. We will call this the "natural" composition hierarchy for reasons to be clarified later. Fig. 11 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 a 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.

B. Discrimination Graphs

The problem of the potential explosion of hypothetical interpretations has largely been ignored in computational

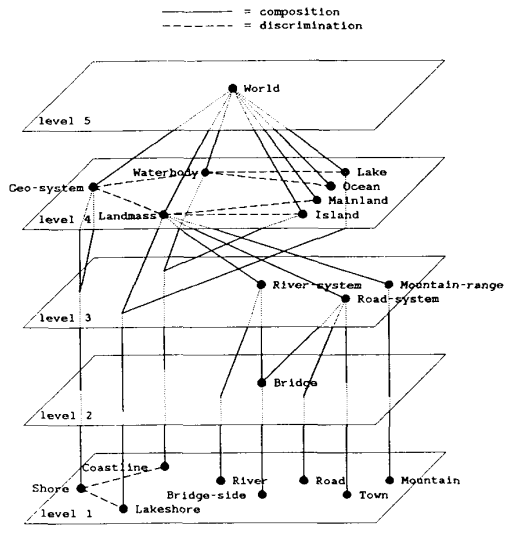


Fig. 11. Mapsee-3 composition hierarchy.

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. 4). 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. 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. Fig. 12 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.

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

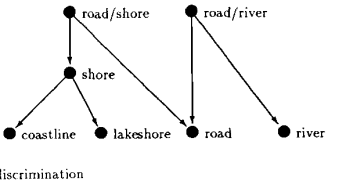


Fig. 12. An example of a discrimination graph.

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. 12 illustrates. A DG can therefore become a lattice with "subset" as partial ordering relation.

DG's have not been employed in other model-based vision systems. Alven [41] uses similarity links, but these links are used to replace rejected hypotheses rather than to prevent hypotheses from occurring as is the case with DG's.

All chains in Mapsee-3 are classified with respect to shape. Table I 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 at both ends. 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. 11). The DG's are constructed orthogonally to the composition hierarchy. Fig. 13 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. 13 can be automatically constructed [29], [32].

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 automatically constructed [29].

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 con-

TABLE I
SHAPE CLASSIFICATION FOR LINE SEGMENTS

1	2	3	4
Road River Mountain Coastline Lakeshore	Road River Coastline Lakeshore Bridge-side	Road Coastline Lakeshore	Mountain Road River
5	**6**	**7**	**8**
Bridge-side Road River	Town	Road River	Road 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

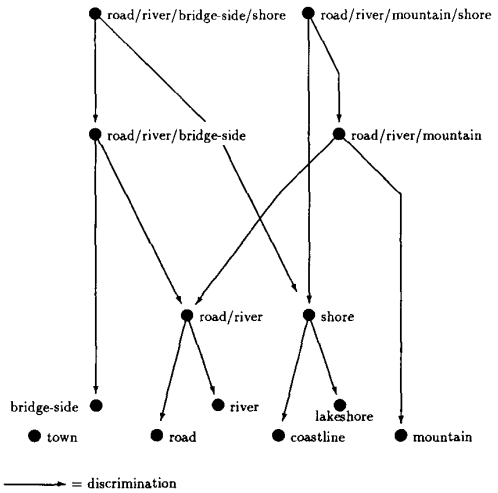


Fig. 13. Discrimination graph at the composition leaf level.

straints. 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 *lake-shore*. This is the case because a road must have land on both sides and a coastline must have water on the outside.

C. 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. 10) 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, Fig. 14 shows the

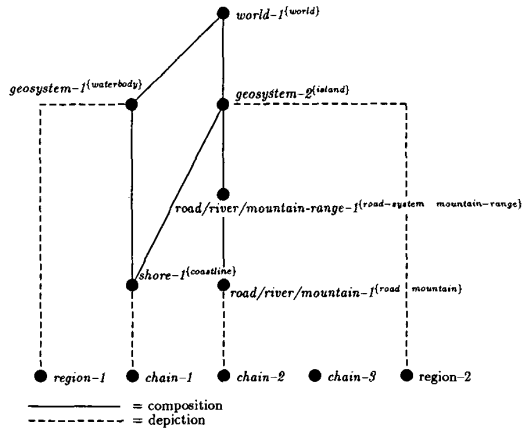


Fig. 14. Mapsee-3 scene constraint graph after interpretation of chain-1 and chain-2.

Mapsee-3 SCG for the same situation as the Mapsee-2 SCG for the Gambier island example (Fig. 8).

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 satisfaction 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 than it could ever be in Mapsee-2. Most constraints are now explicitly represented in the combined abstract and natural composition hierarchy.

D. 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. 11). 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 [29].

Theoretically, the number of predicate tests in *HAC* is of $O(ea^3)$ in the worst case [23]. Experiments with Mapsee-3 show results better than that [29], [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 [22]. 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. This result, however, is specific to the sketch map domain and may not generalize to other domains.

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* [29], [31].

Mapsee-3 solves the particular 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.

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 details 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 satisfaction 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.

By no means do the previous comments imply a level of descriptive and procedural adequacy for Mapsee-3 sufficient for a general-purpose vision system. The Mapsee project has focussed on the model-based aspect of visual knowledge representation, and has paid little attention to early vision. The line segments, for instance, are provided in the input and need not be searched for, as would be the case if the input were a raster image. Thus, the Mapsee project has not seriously addressed the handling of ambiguity in the extraction of features and primitives and its effect on the interpretation process. The problem of finding the most likely interpretation in the light of uncertain features remains an outstanding issue. Several techniques for decision making in the light of uncertain information have been developed in the recent past. Among these are relaxation labeling processes [15], and the Dempster–Shafer formalism [36]. Recent work in neural nets [35] has also indicated some promise towards the solution of this problem. Nevertheless, the success in integrating a schema-based knowledge representation with constraint satisfaction techniques has indicated its utility and the desirability of further explorations of this method.

VI. 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.

REFERENCES

[1] D. Ballard, C. Brown, and J. Feldman, "An approach to knowledge-directed image analysis," in *Computer Vision Systems*, A. Hanson and E. Riseman, Eds. New York: Academic, 1978, pp. 271-282.

[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. New York: Academic, 1978, pp. 3-26.

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

[4] T. Binford, "Survey of model-based image analysis systems," *Int. J. Robotics Res.*, vol. 1, pp. 18-64, 1982.

[5] D. G. Bobrow and T. Winograd, "An overview of KRL, a knowledge representation language," *Cognitive Sci.*, vol. 1, no. 1, pp. 3-46, 1977.

[6] R. C. Bolles, L. H. Quam, M. A. Fischler, and H. C. Wolf, "Automatic determination of image-to-database correspondences," in *Proc. Sixth Int. Joint Conf. Artificial Intelligence*, Tokyo, 1979, pp. 73-78.

[7] R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intell.*, vol. 17, no. 1-3, pp. 285-348, 1981.

[8] E. C. Freuder, "Synthesizing constraint expressions," *Commun. ACM*, vol. 21, no. 11, pp. 958-966, 1978.

[9] J. Glicksman, "Using multiple information sources in a computational vision system," in *Proc. Eighth Int. Joint Conf. Artificial Intelligence*, Karlsruhe, 1983, pp. 1078-1080.

[10] A. R. Hanson and E. M. Riseman, "Visions: A computer system for interpreting scenes," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. New York: Academic, 1978, pp. 303-334.

[11] W. S. Havens, "A procedural model of recognition for machine perception," Dep. Comput. Sci., Univ. British Columbia, Vancouver, Tech. Rep. TR-78-3, 1978.

[12] W. S. Havens and A. K. Mackworth, "Structuring domain knowledge for visual perception," in *Proc. Seventh Int. Joint Conf. Artificial Intelligence*, Vancouver, B.C., 1981, pp. 625-627.

[13] ——, "Representing knowledge of the visual world," *Computer*, vol. 16, no. 10, pp. 90-98, 1983.

[14] W. S. Havens, "A theory of schema labelling," *Computational Intell.*, vol. 1, no. 3-4, pp. 127-139, 1985.

[15] R. A. Hummel and S. W. Zucker, "On the foundations of relaxation labeling processes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 3, pp. 267-287, 1983.

[16] C. A. Kohl, A. Hanson, and E. Riseman, "A goal-directed intermediate level executive for image interpretation," in *Proc. Tenth Int. Joint Conf. Artificial Intelligence*, Milan, Italy, 1987, pp. 811-814.

[17] M. Levine, "A knowledge-based computer vision system," in *Computer Vision Systems*, A. Hanson and E. Riseman, Eds. New York: Academic, 1978, pp. 335-352.

[18] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intell.*, vol. 8, no. 1, pp. 99-118, 1977.

[19] ——, "On reading sketch maps," in *Proc. Fifth Int. Joint Conf. Artificial Intelligence*, Cambridge, 1977, pp. 598-606.

[20] ——, "Vision research strategy: Black magic, metaphors, mechanisms, miniworlds, and maps," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. New York: Academic, 1978, pp. 53-60.

[21] ——, "Constraints, descriptions and domain mappings in computational vision," in *Physical and Biological Processing of Images*, O. J. Braddick and A. C. Sleigh, Eds. Berlin, West Germany: Springer-Verlag, 1983, pp. 33-40.

[22] A. K. Mackworth and E. C. Freuder, "The complexity of some polynomial network consistency algorithms for constraint satisfaction problems," *Artificial Intell.*, vol. 25, pp. 65-74, 1985.

[23] A. K. Mackworth, J. A. Mulder, and W. S. Havens, "Hierarchical arc consistency: Exploiting structured domains in constraint satisfaction problems," *Computational Intell.*, vol. 1, no. 3-4, pp. 118-126, 1985.

[24] A. K. Mackworth, "Constraint satisfaction," in *Encyclopedia of Artificial Intelligence*, S. C. Shapiro, Ed. New York: Wiley, 1987, pp. 205-211.

[25] ——, "Adequacy criteria for visual knowledge representation," Technical Report TR-87-4, Dep. Comput. Sci., Univ. British Columbia, Vancouver, Canada, Tech. Rep. TR-87-4, 1987; to appear in *Computational Processes in Human Vision*, Z. W. Pylyshyn, Ed. Norwood, NJ: Ablex, to be published.

[26] T. Matsuyama and V. Hwang, "Sigma: A framework for image understanding: Integration of bottom-up and top-down analysis," in *Proc. Ninth Int. Joint Conf. Artificial Intelligence*, Los Angeles, CA, 1985, pp. 908-915.

[27] M. Minsky, "A framework for representing knowledge," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York, McGraw-Hill, 1975, pp. 211-277.

[28] J. A. Mulder and A. K. Mackworth, "Using multi-level semantics to understand sketches of houses and other polyhedral objects," in *Proc. Second Nat. Conf. Canadian Society for Computational Studies of Intelligence (CSCSI)*, Toronto, 1978, pp. 244-253.

[29] J. A. Mulder, "Using discrimination graphs to represent visual knowledge," TR-85-14, Dep. Comput. Sci., Univ. British Columbia, Vancouver, 1985.

[30] ——, "Using discrimination graphs to represent visual interpretations that are hypothetical and ambiguous," in *Proc. Ninth Int. Joint Conf. Artificial Intelligence*, Los Angeles, CA, 1985, pp. 905-907.

[31] ——, "Discrimination vision," Technical Report 1987CS-3, Dalhousie Univ., Halifax, N.S., 1987; to appear in *Comput. Vision, Graphics, Image Processing*.

[32] ——, "An algorithm which automatically constructs discrimination graphs in a visual knowledge base," in *Proc. Tenth Int. Joint Conf. Artificial Intelligence*, Milan, Italy, 1987, pp. 855-857.

[33] D. Rosenthal and R. Bajcsy, "Conceptual and visual focussing in the recognition process as induced by queries," in *Proc. Fourth Int. Joint Conf. Pattern Recognition*, Kyoto, 1978, pp. 417-420.

[34] R. B. Roberts and I. P. Goldstein, "The FRL primer," Artificial Intell. Lab., Massachusetts Inst. Technol., Memo 408, 1977.

[35] D. Sabbah, "Computing with connections in visual recognition of origami objects," *Cognitive Sci.*, vol. 9, pp. 25-50, 1985.

[36] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton University Press, 1976.

[37] R. C. Shank, "Conceptual dependency: A theory of natural language understanding," *Cognitive Psychol.*, vol. 3, no. 4, 1972.

[38] M. J. Stefik, "An examination of frame-structured representation systems," in *Proc. 6th Int. Joint Conf. Artificial Intelligence*, Tokyo, 1979, pp. 845-852.

[39] S. Tanimoto and T. Pavlidis, "A hierarchical datastructure for picture processing," *Comput. Graphics Image Processing*, vol. 4, pp. 104-119, 1975.

[40] J. M. Tenenbaum, M. A. Fischler, and H. C. Wolf, "A scene analysis approach to remote sensing," AI Center, S.R.I., Tech. Rep. TN 173, 1978.

[41] J. K. Tsotsos, "Knowledge organization and its role in representation and interpretation for time-varying data: The ALVEN system," *Computational Intell.*, vol. 1, no. 1, pp. 16-32, 1985.

[42] ——, "Image understanding," in *Encyclopedia of Artificial Intelligence*, S. C. Shapiro, Ed. New York: Wiley, 1987, pp. 389-409.

[43] D. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York: McGraw-Hill, 1975, pp. 19-91.

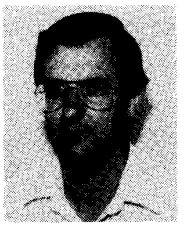
[44] T. E. Weymouth, "Using object descriptions in a schema network for machine vision," Ph.D. dissertation, Dep. Comput. Inform. Sci., Univ. Massachusetts, Amherst, COINS Tech. Rep. 86-24, 1986.

[45] T. Winograd, "Frame representations and the declarative/procedural controversy," in *Representation and Understanding: Studies in Cognitive Science*, D. G. Bobrow and A. Collins, Eds. New York: Academic, 1975, pp. 185-210.

Jan A. Mulder was born in 1948 in The Hague, The Netherlands. He received the "kandidaats" and "doctoraal" degrees in experimental psychology from the University of Leiden in 1974 and 1979, respectively, and the M.Sc. and Ph.D. degrees in computer science from the University of British Columbia Vancouver, B.C., Canada, in 1979 and 1985, respectively.

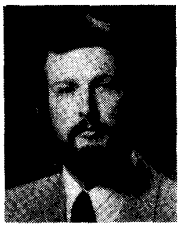
Since 1985 he has been an Assistant Professor in Computing Science at Dalhousie University, Halifax, N.S., Canada. In 1986 he was awarded a Senior Killam Fellowship. His research focuses on knowledge representation in computational vision systems, constraint satisfaction, schema-based representations, and perceptual organization.

Dr. Mulder is a member of the IEEE Computer Society, the American Association for Artificial Intelligence, and the Canadian Society for Computational Studies of Intelligence (CSCSI).



Alan K. Mackworth received the B.A.Sc. degree from the University of Toronto in 1966, the A.M. degree from Harvard University in 1967, and the D.Phil. degree from the University of Sussex in 1974.

Since 1974, he has been at the University of British Columbia. He is currently a Professor of Computer Science at the University of British Columbia, Fellow of the Canadian Institute for Advanced Research, and co-director of the UBC Laboratory for Computational Vision. Since then, he has been at UBC. His research focuses on the representation and use of knowledge in computational vision systems, including surface orientation representations, network consistency algorithms for constraint satisfaction problems, schema-based systems, and logic-based systems. Applications include diagram understanding, telerobotics, and the interpretation of satellite imagery and sketch maps.



William S. Havens (M'79) received the B.Sc. and M.Sc. degrees in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, in 1969 and 1973, respectively, and the Ph.D. degree in computer science from the University of British Columbia, Vancouver, B.C., Canada, in 1978.

He is a Senior Computer Scientist at Tektronix Laboratories in Beaverton, OR, and adjunct Professor in the School of Computing Science at Simon Fraser University in Burnaby, British Columbia. His current research interests focus on architectures for constraint-based reasoning systems and their applications to diagnosis, synthesis, and other recognition tasks. He has worked on computer map understanding, on schema recognition systems, and currently on diagnostic instrumentation.

Dr. Havens is a member of the American Association for Artificial Intelligence, the Canadian Society for the Computational Studies of Intelligence, and the IEEE Computer Society.