Model-Building in the VISIONS High Level System

Thomas D. Williams and John D. Lowrance

COINS TECHNICAL REPORT 77-1

January 1977

ABSTRACT

SYMBOLS, a general purpose model-building tool, has been designed and implemented in order to develop and test the semantic interpretation portion of the VISIONS system. Following the criterion of modularity, the data, processes and search concepts of model-building have been decomposed into units which are natural for the understanding of digitized outdoor scenes. Multiple-leveled structures are described for the representation of the model and processes. The strategy surrounding the control of processes in a huge search space is integrated into the system via a hierarchy of modular substrategies of control.

This research was supported by the National Science Foundation under Grant DCR75-16098.

*Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts 01003

TABLE OF CONTENTS

ACKNOWLEDGMENTS

The work presented in this report has been the result of combined efforts among the members of the VISIONS research team. The core of the paper was presented as the first author's masters project, the continuing efforts of both authors have created SYMBOLS. It could not have proceeded without the tireless persistent help of Ed Riseman and Al Hanson, as well as interactions with Daniel Fishman, Richard Hudson, Kurt Konolige, John Prager and Bryant York.

I. INTRODUCTION

I.1 SYMBOLS

SYMBOLS is a general purpose model-building system. This system provides three structural areas for model-building. They are 1) a layered graph structure for short-term and long-term data storage, 2) a loose hierarchy of processing elements, and 3) a model search space. SYMBOLS stands for StrategY Guided Model Building Operating on Multi-Leveled Structures. It is a tool used for high level VISIONS research.

This report is a description of SYMBOLS and the first pass at implementing the high level interpretation portion of the VISIONS system [HAN75a,75b], a computer system for the segmentation and interpretation of digitized images of natural scenes. SYMBOLS can be used in a variety of ways. However, the examples given concern only VISIONS. As research continues, both SYMBOLS and VISIONS will undergo further refinement.

I.2 MODULARITY

The primary design criteria for SYMBOLS is that it should provide a framework in which different model building strategies and techniques can be easily embodied and understood in terms of their utility.

Visual perception necessitates the integration of many diverse forms of knowledge in a nontrivial way. Because this complexity is inherent in the problem, any model-building system being designed to solve the problem must utilize some powerful methodological weapons to combat it. The traditional methodological weapon against complexity has been decomposition, or modularity [SIM69]. However, to benefit from decomposition, it is essential that both the resulting individual modules and their interaction can be understood. If an isolated module remains overly complex, decomposition can be reapplied directly to that module. But if the complexity remains in the coordination of several modules, further decomposition will not reduce the complexity.

The decomposition of knowledge into "natural" units (i.e., those concepts which a person discussing the domain might find helpful) is a cue that modularity is probably being used in an effective way and that artificial barriers limiting the flexibility of the system are not being created. In addition, it reduces the system to a number of nicely delimited subproblems for independent attack.

I.3 REPRESENTATION

Model-building is a process whereby a system generates an internal representation of a situation. The situations we are concerned with are those presented by the visual environment. Our narrowed definition of model-building then is focussed upon a perceptual system; and the definition of a perceptual system seems to require several necessary parts. It must have an internal representation of the environment, i.e., a model. Continuing a step further, the model must have a representation of the relevant concepts so that elements of the model can be defined. For example, visual models require the identification of objects (the model element) in terms of the class of objects to which they belong (such as "Car", "Tree", etc.) A facility is provided in SYMBOLS for these representations as a multi-leveled graph structure.

I.4 PROCESSES

Another characteristic of model-building systems is that some process(es) must decide what to add or change in the model. The general paradigm for model-building in the VISIONS system is 'hypothesize-test-build'. A number of hypothesize-test-build processes are expressed and a strategy guides their application.

SYMBOLS divides these processes into categories. At the highest level strategic decisions result in changes to the model, while at the lowest levels tactical processing manipulates complex knowledge structures with relative ease. Several levels of processes between the two extremes form a loose hierarchy which is helpful to the researcher in clearly defining the responsibility of various model-building processes in a complex system.

I.5 SEARCH SPACE

The last major structural area that SYMBOLS deals with is the search space. Model-building systems have a wide degree of variation in this respect. For instance, the HEARSAY [ERM75] system does not keep a trace of model-building history. This may be the correct approach in a sequential, time-dependent domain such as speech understanding, but it demands that interpretation follow directly from model-building with no backtracking. VISIONS' initial formulation is a response to the problem of constructing a complex system where the strategies for control of the search are not yet understood. This has led us to the design of the SYMBOLS system where the entire search history is available to the search strategy. This allows the system (and the human user during developmental phases) to examine at any point in the analysis the history of the utilization of processes during model construction.

II. DECLARATIVE KNOWLEDGE: A Multi-Leveled Graph Structure

II.1 Levels of Abstraction

Natural concepts of visual knowledge would seem to include the idea that knowledge relevant to a single scene includes information at many different levels of abstraction [ERM75,HAN76c,TAN75,UHR72]. The grossest division can be made between the 2D information within the image and the 3D information about the world it implies.

Finer levels of abstraction can be found at both the world and image levels. The world can be described in terms of frames of stereotypic scenes [MIN75,HAN76c], the objects involved in those scenes, or the surfaces which delimit those objects. A digitized photographic image can be described by the individual pixel points making up the image or in terms of the more abstract entities resulting from its segmentation (i.e., regions, edges, etc.) [OHL75,ROS71,TEN76]. The world and image meet where visible portions of surfaces are manifest as regions and edges. This is not to say that these are necessarily the correct, best, or complete set of abstraction levels, but only that such levels seem natural and useful.

Once such levels are defined, they naturally constrain the components of visual knowledge and the relationships that can exist between them. Interlevel relationships constrain the levels in a linear fashion resulting in a hierarchy of visual knowledge (see Figure 1). Points collect into regions and edges, region and edges into surfaces, surfaces into objects, and objects into frames. In addition, the components at each level have their own characteristic features and relationships. For example, at the region and edge level these might include 2D shape, 2D size, visual texture

[BAJ73,HAN75b,ROS76], image color, absolute image location, relative image location, and others. While at the object level they might include 3D shape, 3D size, 3D physical texture, color (normalized over lighting conditions), semantic identity (e.g., house, tree, bush, road), absolute 3D location, relative 3D location, and others.

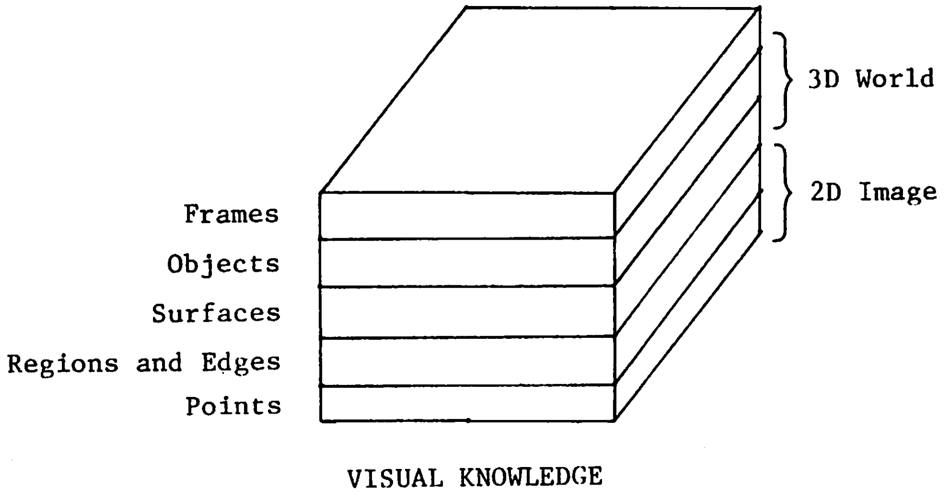


Figure 1 Levels of Abstraction in Visual Knowledge

II.2 Short-Term Model-Specific vs. Long-Term General Visual Knowledge

A <u>model</u> can be viewed as a set of short-term instantiations of long-term general concepts relative to the current environment. This suggests that, in addition to dividing visual knowledge into hierarchically related levels of abstraction, each level should also be subdivided into <u>short-term model-specific</u> and <u>long-term general knowledge sections</u>. <u>Model-</u>

building thus becomes the process by which the short-term model-specific side of the hierarchy is filled out and related to the long-term general knowledge side of the hierarchy based on the environment (see Figure 2).

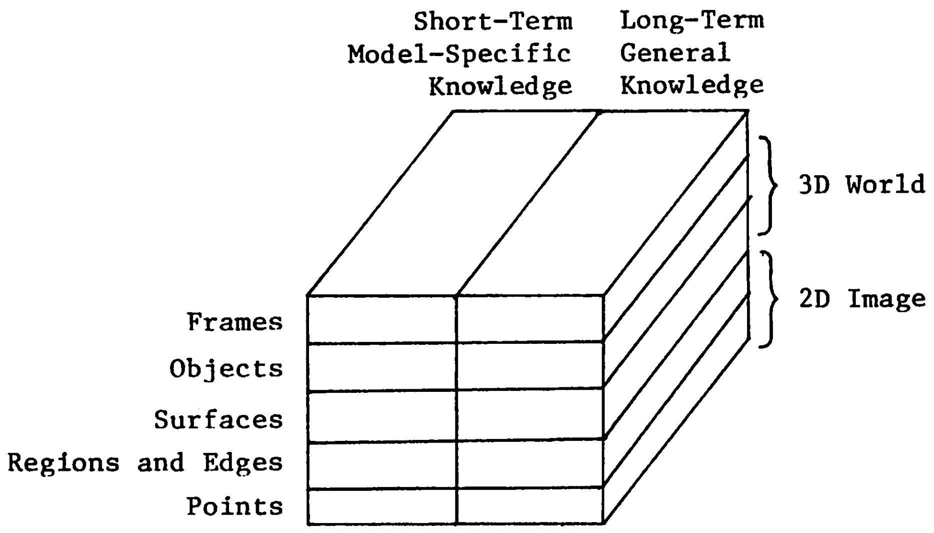

Short-Term Model-Specific Knowledge
Long-Term General Knowledge
3D World
2D Image
Frames
Objects
Surfaces
Regions and Edges
Points

Figure 2 Visual Knowledge Framework

The nature of this framework affords a number of advantages. One is that comprehension of visual knowledge is facilitated since any confusion of a concept at one level might be eliminated by examining the lower and higher level concepts to which the problem concept is related. Another advantage is that incomplete areas of a specific model are readily apparent. Regions not explained as surfaces, surfaces not explained as objects, and surfaces not yet explained as regions and edges are all examples that point to incomplete portions of a model.

II.3 Model-Building and the Knowledge Framework

Model-building techniques can be roughly divided into those which are _generative_ (i.e., top-down) and those which are _predictive_ (i.e., bottom-up) [ERM75]. A generative process makes an addition to a level in the hierarchy based on some model-specific information from a lower level while a predictive process bases its new addition on model-specific information from a higher level. Of course, with either method, general knowledge at those levels plays a role in selecting likely hypotheses.

Notice that _hybrid techniques_ can be just as easily understood in terms of these concepts. For example, a process might hypothesize the existence of an object only if it both covers the features of some instantiated surfaces and is predicted by an instantiated frame.

Further, it can be seen that these levels break up the model-building process into subprocesses that can be carried out independently by a number of different _knowledge sources_ [ERM75]. For example, a purely generative surface builder would only use information at the region and edge level to construct surfaces at the surface level. Similarly, a predictive object builder would use information at the frame level to construct objects at the object level. Such process independence allows knowledge sources to be easily exchanged and their results compared without having to restructure the system. Thus, the assumptions made concerning the structure of the problem space help clarify and delimit the component parts of the model-building process.

II.4 The Data Structure for Declarative Knowledge

These considerations and assumptions have led us to adopt extended hypergraphs as the primitive data structure for declarative knowledge in SYMBOLS. A hypergraph [FRI69,PRA71] is a collection of directed graphs, each residing on a different labelled plane. A directed graph consists of labelled directed arcs between labelled nodes. An extended hypergraph differs from a standard hypergraph by allowing arcs between nodes residing on different planes. Although the arcs are directed, this does not imply that they can only be traversed in a single direction. The directionality of arcs is for increased semantic power and is not meant to imply a one-way pointer structure.

The desired knowledge framework is easily manifest in this structure. Each level of abstraction can be realized as two planes; one for the short-term model-specific knowledge and the other for the long-term general knowledge. Each node on the short-term model-specific side corresponds to one of the primitive elements (i.e., concept) characteristic of that level. Thus, a node on the short-term surface plane corresponds to a surface, a node on the short-term frame plane to a frame, etc.

Nodes on the long-term planes correspond to primitive symbolic classes of characteristic elements at that level. Object class nodes might include classes for trees, bushes, houses, red things, leafy things, etc., whereas region class nodes might include classes for elongated regions, circular regions, smooth textured regions, polygonal regions, etc. Each such class corresponds to some subset of all possible entities that could exist at that level. The particular subset of entities that the class denotes is specified by associated properties that a member of the class must possess. Although an arbitrary number of classes can be defined at each level, in practice only classes which provide useful subsets are utilized. For example, if surfaces which are large, smooth, green spheres are important, a class node corresponding to such surfaces would be placed on the surface level. This node would then provide an immediate means for accessing the information concerning such surfaces including any current instantiations of them, the classes of objects in which such surfaces can participate, the type of regions produced by their projections, and superclasses and subclasses which provide classes to be considered for further information. The utility of a class is proportional to the amount of guidance that its associated information provides the model builder.

Relations between the primitive concepts within the framework can be represented by arcs between the nodes in classic semantic network style [QUI68, SIM73,FAH75,WOO75,RUM75]. Arcs between levels relate concepts at a given level to the concepts residing on planes in the neighboring higher and lower levels. Thus, from an object node, arcs leading up can be followed to nodes at the frame level representing those frames in which the object participates, and arcs leading down can be followed to surface nodes indicating those surfaces which define the object; the same is true for nodes on the object class plane. Arcs between planes at the same level (i.e., those going between nodes on the model-specific and general knowledge planes of a single level) indicate which general classes each model-specific element is an instantiation of.

For example, an instantiation of a green car could be represented by a model-specific node connected by arcs to the green object class node and the car class node. _Intraplanar_ _arcs_ can be used to express additional relationships, including part/whole, subclass/superclass, and other relationships peculiar to particular planes (see Figure 3).

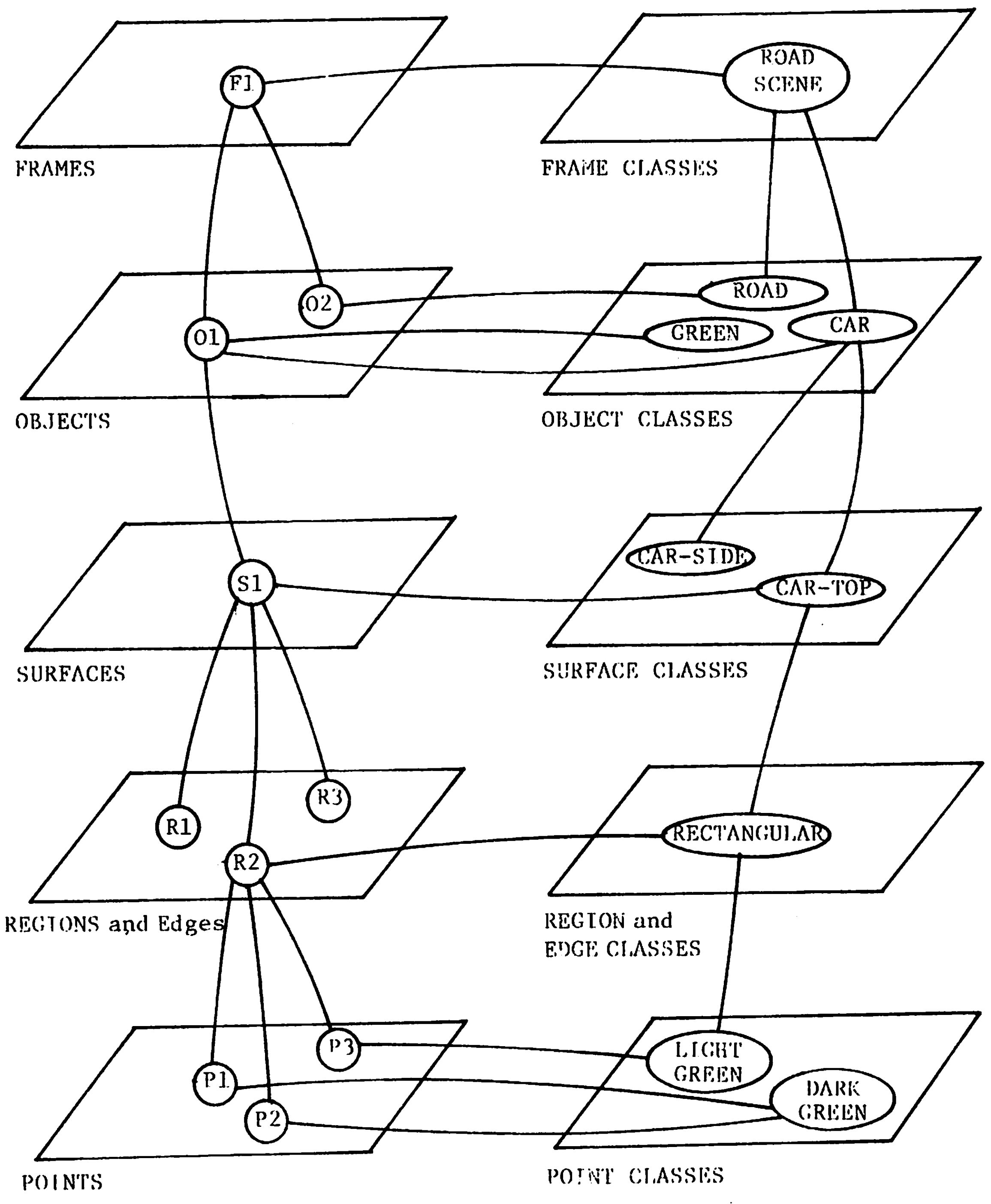


Figure 3 Visual Knowledge Framework as Extended Hypergraph

Two other features of our implementation of extended hypergraphs lend additional flexibility and power. The first is the ability to attach properties to each node, arc, or plane in traditional property-value form [McC62]. This feature provides a convenient means for associating numeric values with all entities indicating the frequency or likelihood of their existence. The second is that arbitrary sub-graphs, partitions, can be defined within the extended hypergraph corresponding to different contexts that might occur. We expect these to be useful in limiting the relevant world knowledge that will be used at any particular moment. For example, if the information concerning the context of the image includes that it was a winter day and 12 noon, all information not included in both the WINTER and MIDDAY partitions can be ignored.

II.5 VISIONS' Utilization of SYMBOLS

Although SYMBOLS could be used to model visual perception through all of the hypothesized levels (from points to frames), VISIONS is only utilizing it at the higher levels. The transformation from the level of points to the level of regions and edges is being handled by another system especially tailored to the needs of this low level segmentation process. This system, the processing cone [HAN74,HAN76b], is a simulation of a hierarchical parallel array computer. It is better adapted to handling the large volume of numeric data entering the system as a two-dimensional array and the application of segmentation algorithms like those historically utilized in scene analysis [OHL75,ROS76].

This divides VISIONS into two major subsystems with corresponding subprocesses; the low-level system responsible for segmenting the image utilizing the processing cone and the high-level system responsible for interpreting the segmented image utilizing SYMBOLS. Although these two subsystems must eventually be integrated, they are currently being developed independently. We are assured that integration will eventually be possible by using the region and edge level as an interfacing structure. The low-level system is concerned with combining the results of various segmentation algorithms to form a consistent structure at the region and edge level. The high-level system is concerned with the interpretation of the resulting structure. This does not preclude the eventual use of feedback between the two subsystems.

A number of factors suggest that the 2D syntactic information characteristic of the region and edge level should have its own hierarchical structure. A great deal of information about a region is directly related to the edges defining its boundary. In fact, such information is often related to only a portion of the entire boundary; part of a boundary may be straight and another part crooked, different portions of a boundary divide the associated region from particular other regions, etc. This suggests that it should be possible to directly associate such information with these segments (i.e., portions of boundaries). Similar arguments support the need to be able to directly associate properties with the endpoints of these segments. It can be seen that there is a natural hierarchy to these elements; regions map down to segments and segments to endpoints. This suggests that the 2D syntactic information should be represented as a three level hierarchical structure with levels for regions, segments and endpoints; and that each level should consist of two planes, one for model-specific knowledge and another for general knowledge. Such a structure has been adopted and is referred to as RSE (<u>R</u>egions, <u>S</u>egments, and <u>E</u>ndpoints). Its structure parallels the structure of the rest of declarative knowledge.

The relationship between the described framework for visual knowledge and VISIONS is shown in figure 4.

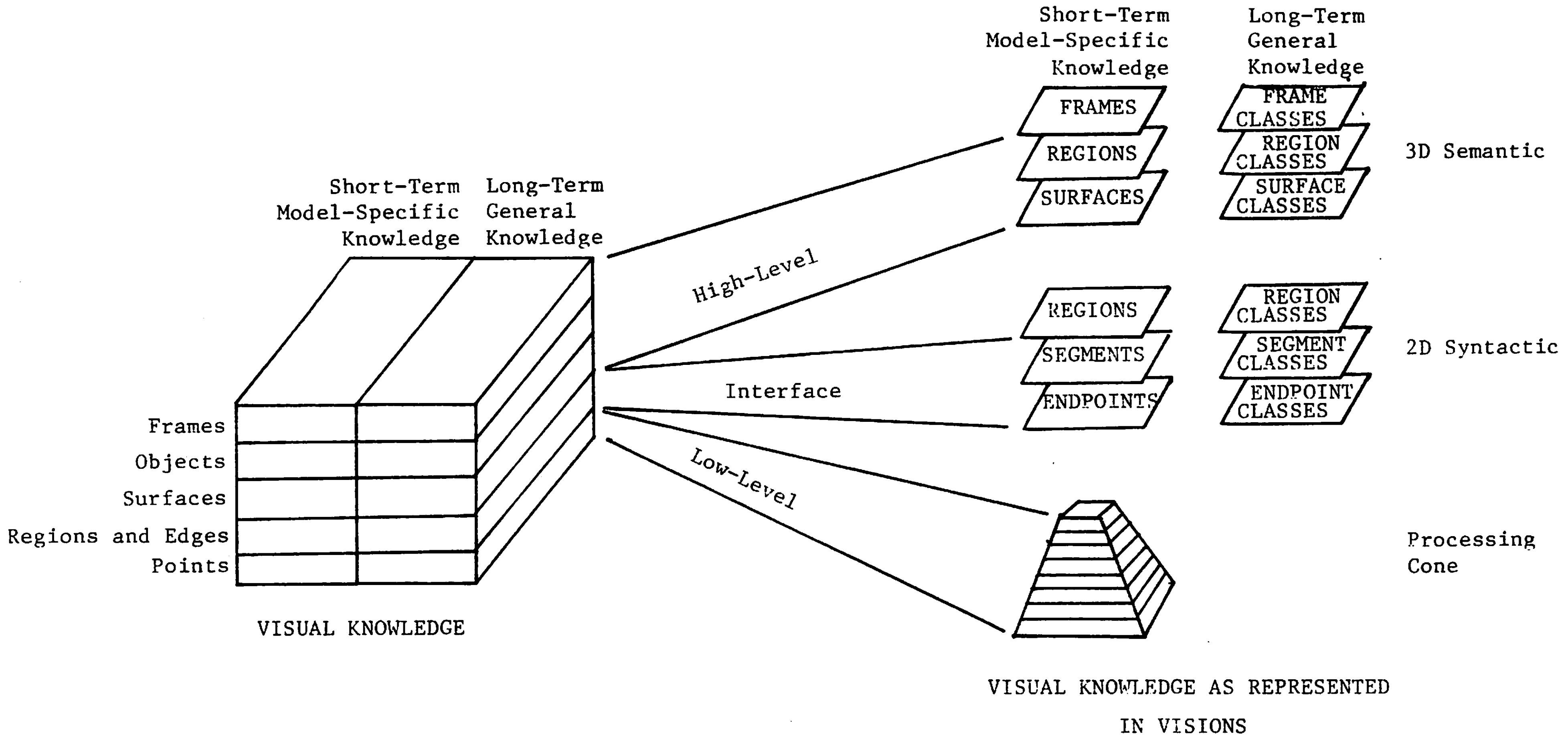


Figure 4 Visual Knowledge and its Representation

II.5.1 The RSE Structure of VISIONS

The RSE structure is designed so that key two-dimensional relationships and features can be easily determined without having to continually reference the vast amounts of data in the low-level systems. It is particularly important that adjacent regions and connected segments be immediately available.

Each region is connected to all the segments participating in its boundary plus any segments which lie in its interior. By definition a segment bounding a region separates it from exactly one other region. During the construction of RSE, if a portion of a boundary separates more than two regions, it is subdivided into segments such that each separates exactly two regions (see Figure 5). A segment may be further subdivided if portions of it have different characteristic properties (e.g., portions are straight and others curved). Nonbounding segments in the interior of a region are distinguishable from bounding segments since they are linked to only a single region. Adjacent regions are available through a region's bounding segments (see Figure 6a).

Endpoints linked to a segment serve to both define those segments to which it is connected and anchor it in two-space. Connected segments have a common endpoint. This allows any connected sequence of segments to be extracted. Associated two-dimensional coordinates of endpoints locate segments in space. Segments which close on themselves are given an arbitrary endpoint so that they too will be fixed in two-space (see Figure 6b).

Additional relationships can be represented as explicitly labelled arcs between primitive visual entities. For example, containment is represented by "C" arcs from the containing region to the contained regions (see Figure 6a).

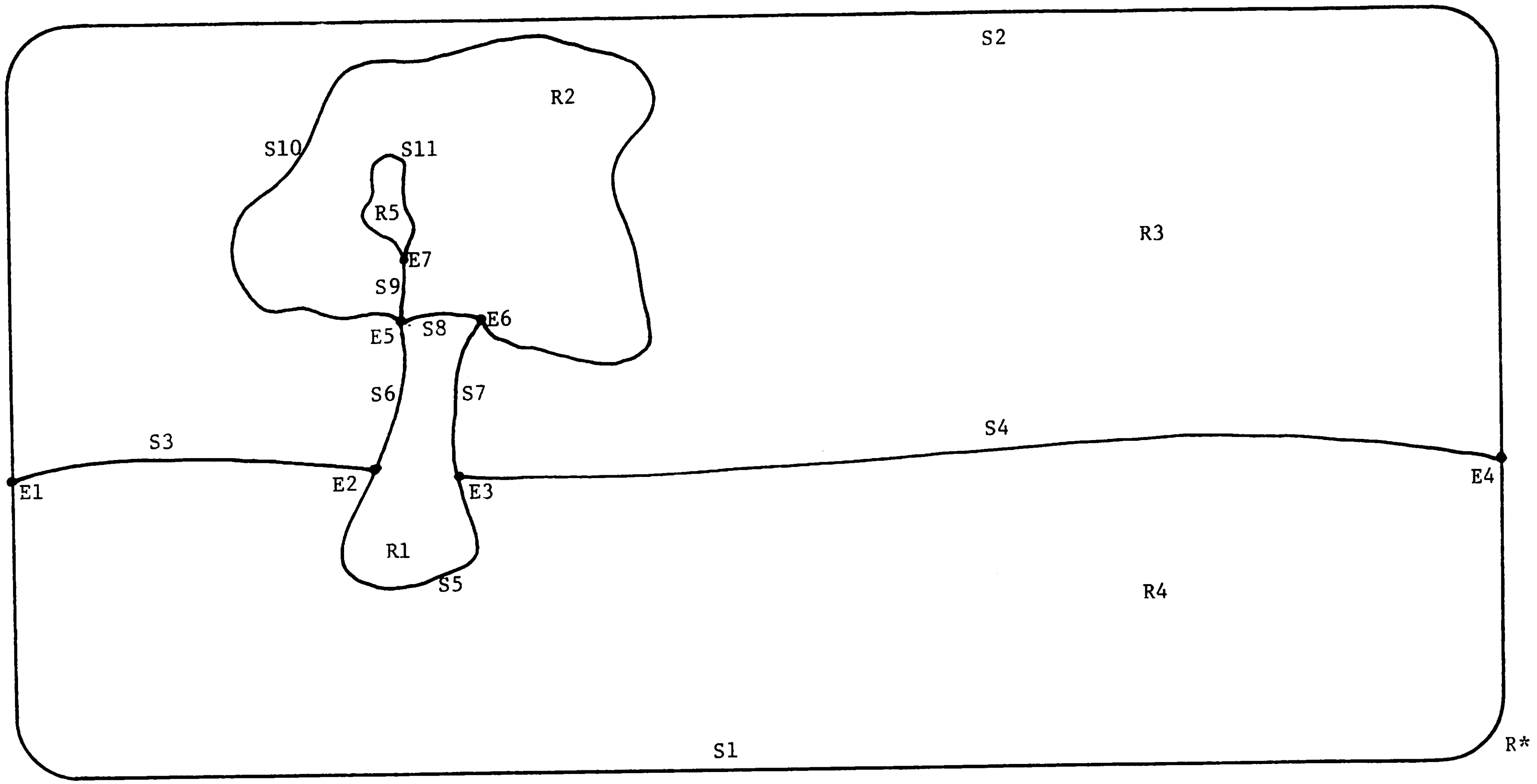


Figure 5 Simplified Segmented Scene

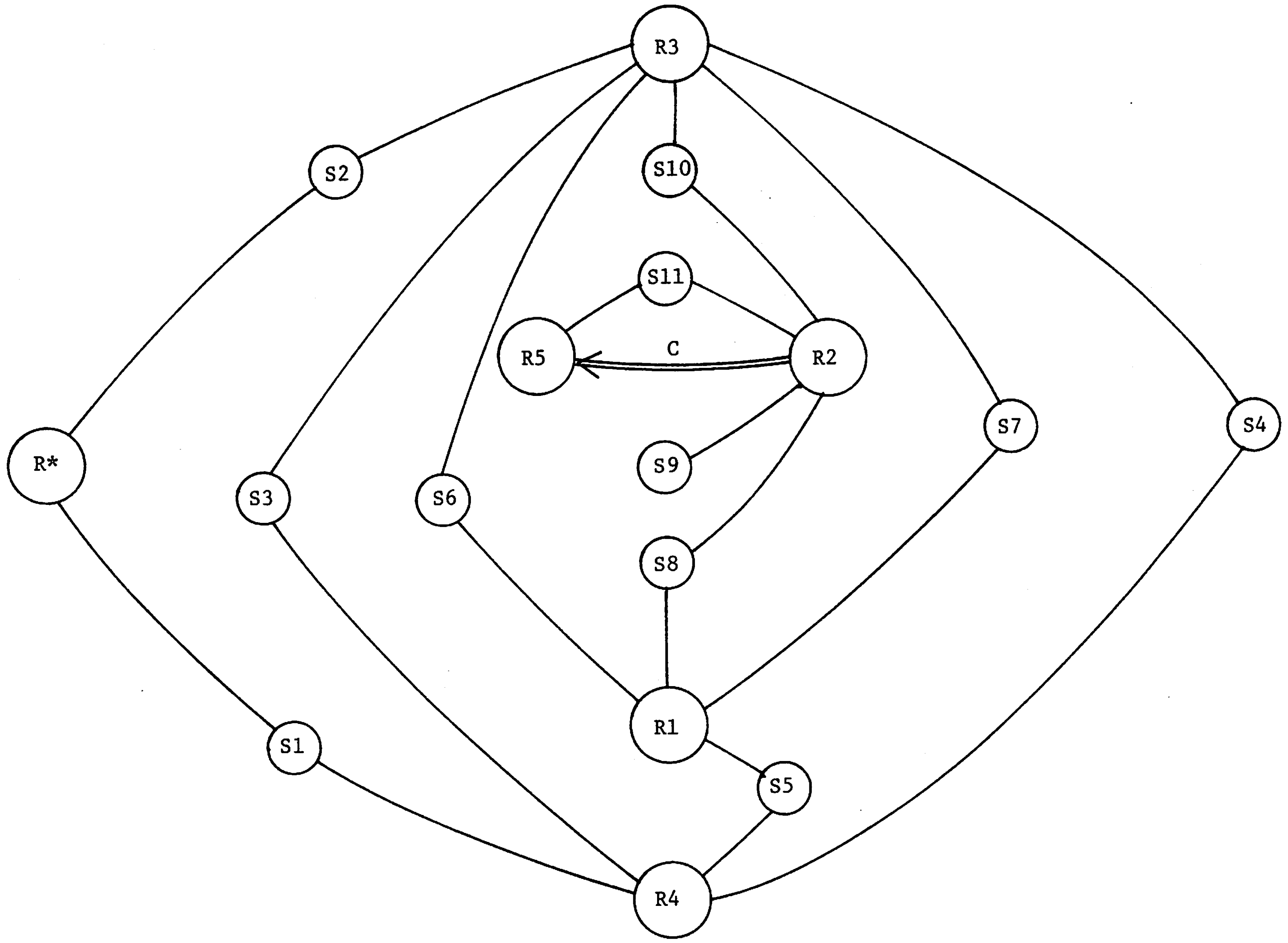


Figure 6a Collapsed Region and Segment Planes of RSE for Figure 5

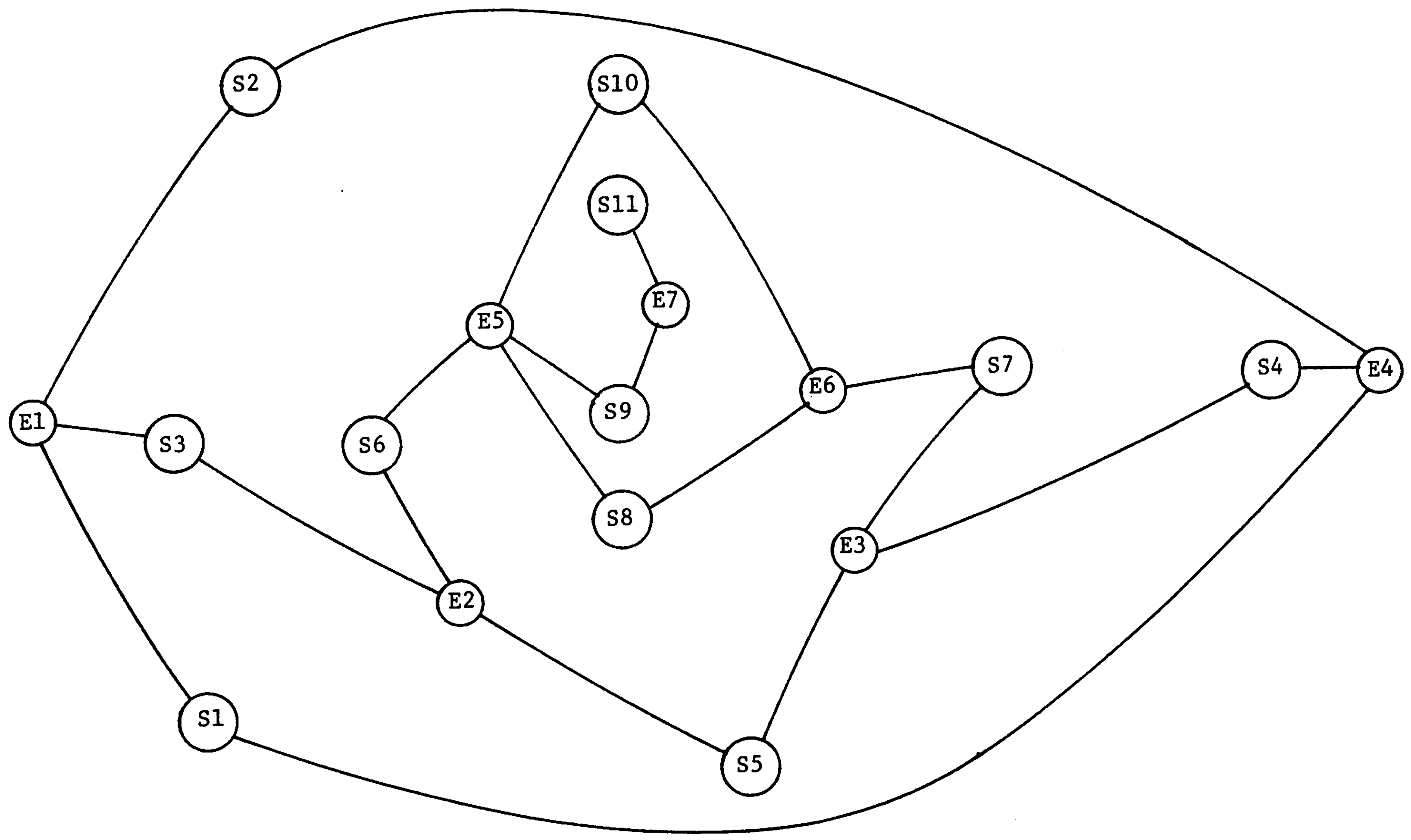


Figure 6b Collapsed Segment and Endpoint Planes of RSE for Figure 5

Examples of features associated with the entities in RSE include the following:

regions --	hue, saturation, intensity, texture, location, size, shape, orientation, etc.;
line segments[1]--	location, quality (straight,curvilinear, or various characteristics of irregularity),width of gradient, orientation, etc.; and
endpoints --	location, type of vertex such as the polyhedral fork, arrow, T, etc.

See [HAN76c] for examples of operations that access information stored in the RSE structure.

II.5.2 The Semantic Levels of VISIONS

The concept of a frame used in VISIONS is similar to that of Minsky [MIN75], although aspects are simplified and tailored particularly to vision. In VISIONS a frame defines stereotypic grouping of objects. It focusses upon the expected relationships between objects, particularly their relative position in three-space. A road scene frame would describe relationships between the road, cars, guard rails, telephone poles, etc.

Another key characteristic of a frame in VISIONS is the rough specification of the importance of the presence of each object to the frame, and vice versa. Frames are expected to provide top-down information predicting

1 The chain encoding [FRE61] of a line segment on a rectangular grid might be stored so that further extraction of properties can be carried out later. This can be done at different resolution levels so that a jagged line which globally is straight would have the local and global properties consistent and accessible. Of course the chain encoding at a coarser resolution can be obtained directly from the chain encoding at a fine resolution.

and confirming the existence of other objects, and possibly, further features based on objects and features already identified.

The relative importance of particular objects and their relationships to a frame are captured to a first approximation as weights. Weights from objects to frames rate the likelihood that the presence of an object implies the presence of a frame, while weights from frames to objects store the likelihood that the presence of a frame implies the presence of an object. The presence of a guard rail strongly implies the road scene frame, while the inverse implication is much weaker since the absence of a guard rail in a road scene is to be expected in many cases.

Of course, this is a crude approximation to the dependencies between objects in a frame. However, it is clear that the accurate estimation of the joint probability distribution of all the subsets of the objects in a road scene is not feasible. Thus, this problem will be approached heuristically with intuitively selected weights. This leads to the problem of tuning the weights; but since these weights are only intended as rough estimates, this problem should remain tractable.

Objects appearing in a frame may also have their own frame specifications. A tree may be part of a road scene frame and be treated as an object with attributes. On the other hand, a more detailed description of the parts of a tree require spatial definition and relationships. Thus, the TREE class node can be treated as an object or a frame during model-building.

Long-term general knowledge at the frame and object levels may then be treated as a network of frame and object classes in which many entities may be dealt with in either way by the system. The primary structure of the network is maintained through part/whole relationships. A tree is designated through specially labelled arcs as a possible part of a road scene and, in turn, as having a trunk and crown as parts of it. Thus, the parts of a frame are specified in the same way as the parts of an object. A sufficiently flexible spatial processor is being developed which will manipulate the spatial properties of both frames and objects in a consistent way [YOR76].

Subclass/superclass structural relationships are also utilized in the network. These can be used for the implicit storage of information [FAH75,QUI68]. For example, if trees have trunks as a part and elms are a subclass of trees, then it can be inferred that elms have trunks. Alternatively, this information could have been explicitly (but redundantly) stored off the class of elms. In addition, the subclass/superclass structure can be used as a decision tree for recognition [MAR75]. Initial identifications can be refined by stepping through more specific subclasses of the initial class; thus an object identified as a tree might be reidentified as an elm.

As of yet, we still do not have a solid understanding of the details of the surface level in our framework. Work on this level is currently being conducted in conjunction with the development of general shape and space processors.

Examples of features associated with entities at these levels include the following:

frames -	relative shape, size, location and orientation of each part in the frame, and the size, location, orientation, and number of each subframe, etc.;
objects -	shape, size, location, orientation, color, texture, etc.;
surfaces -	shape, size, location, orientation, color, texture, etc.

Figure 7 shows a complete model indicating the connections between the semantic planes and the region level of RSE. This does not include many of the particulars just discussed in the interest of clarity.

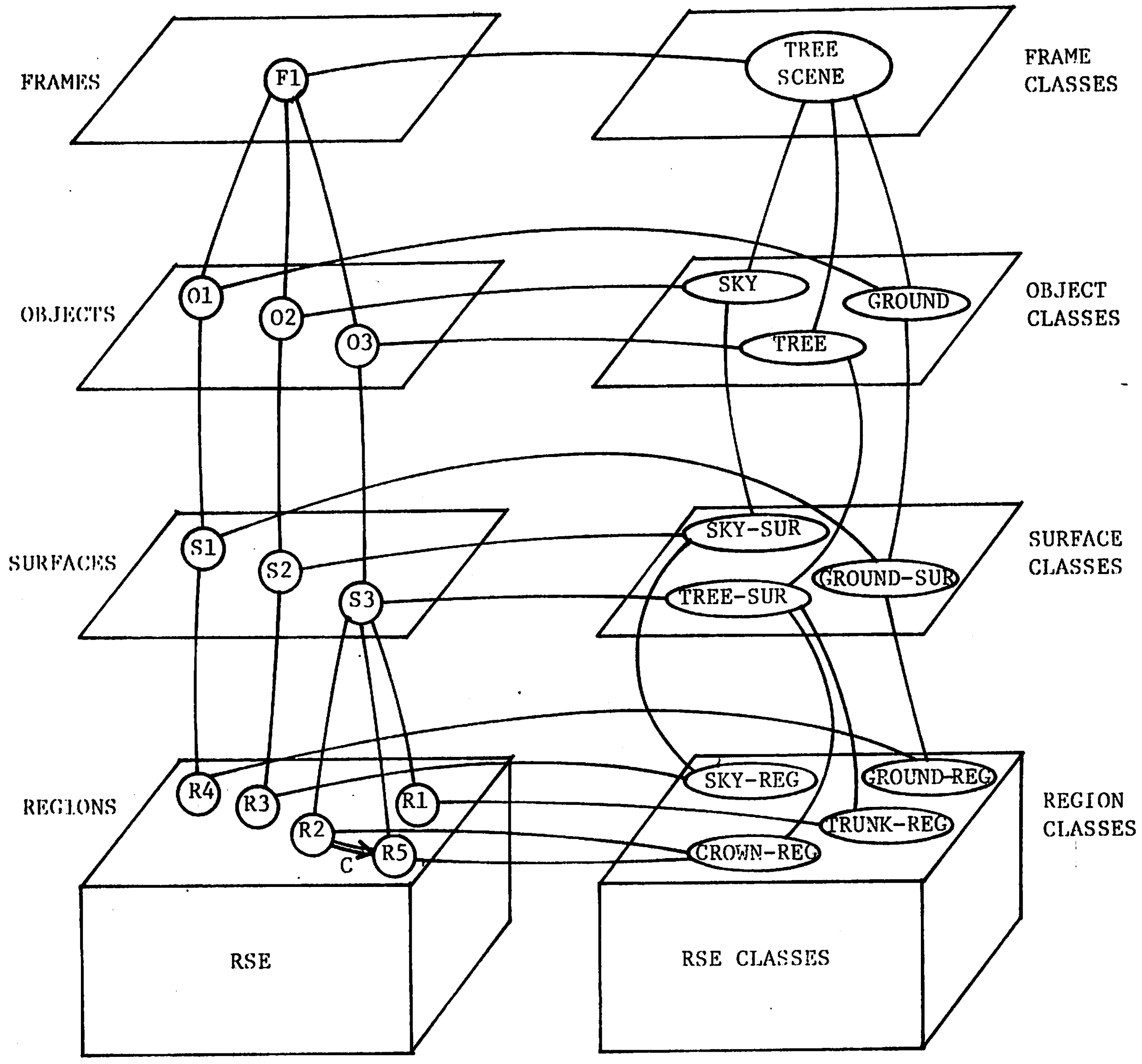


Figure 7 A Complete Model for Figure 5

III. PROCEDURAL KNOWLEDGE: A Loose Hierarchy of Model-Building Processes as a Modular Control Strategy

Procedural knowledge in SYMBOLS is multileveled. Each level contains a model building process, each having a strategic component. Together these strategic components form the strategy or control structure for model-building [HAN76a].

III.1 Organization of Modular Control Strategy

The strategy is divided into levels of abstraction (see Figure 8) according to the type of manipulations required in model-building. SYMBOLS allows an arbitrary description of the strategy hierarchy; however, for the sake of clarity, only the major levels of the organization being used for VISIONS will be given here.

The level #1 strategy is also called the H-L-STRATEGY for Highest Level. This level of strategy is linked (see Figure 9) to the model search space, and therefore is at the highest level of search (for the most satisfactory model). The next major level of strategy, level #2, has components responsible for making some change in a partial model. The discussion about the level of strategy is directed to its principal component, CONTRACTORS. These strategy elements are then divided into level #3 strategy components. These are the processes responsible for hypothesis formation, hypothesis testing, and instantiation following the model-building paradigm of hypothesis-test-build. The instantiation of a hypothesis is equivalent to the generation of a new partial model.

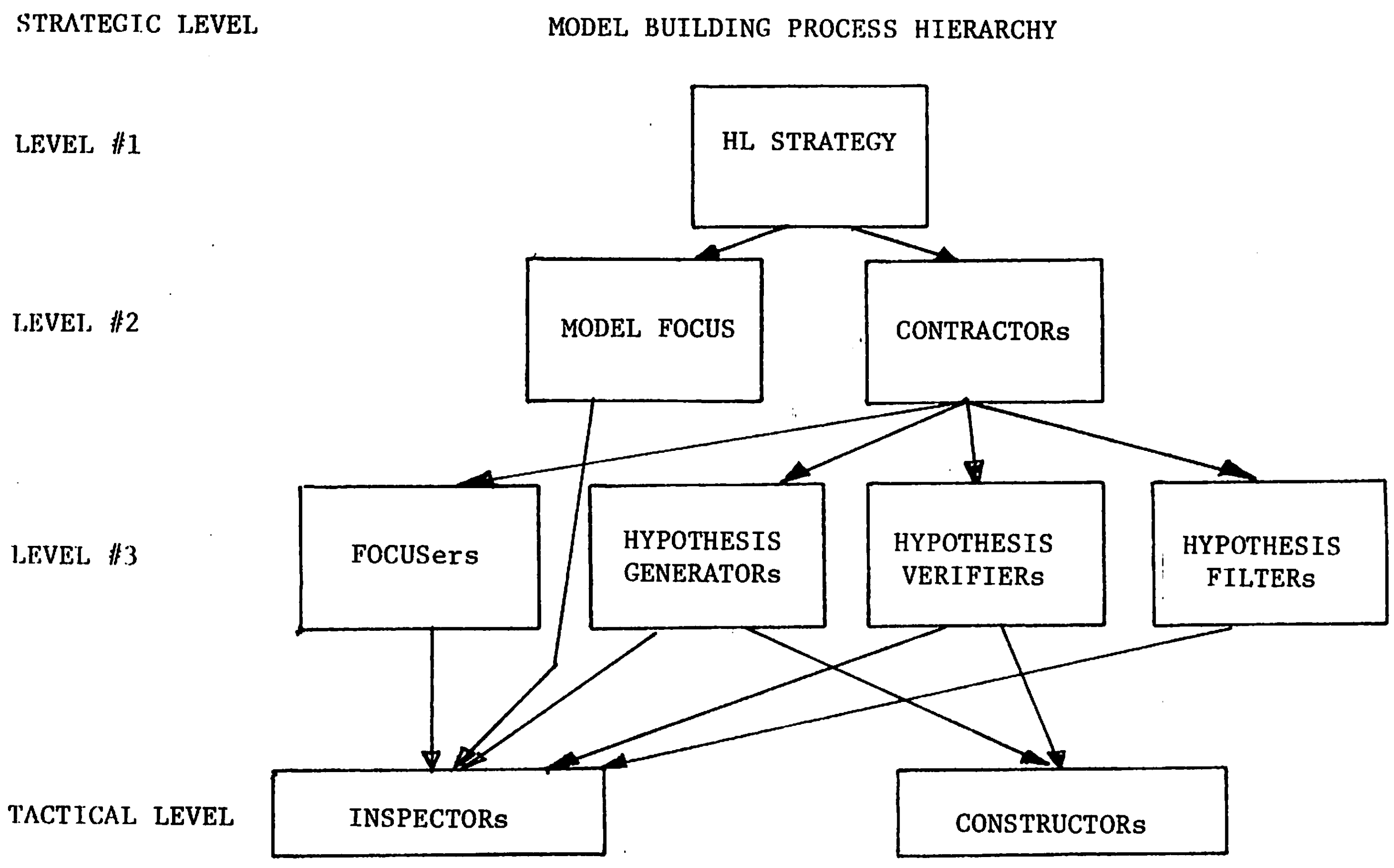


Figure 8 The Major Levels of Strategy Used by VISIONS to Organize the Model Building Process

The actual VISIONS high-level research is being conducted with a finer resolution of levels. This example serves to capture the main organization which has remained constant in a system where details are often being changed.

STRATEGIC LEVEL	TYPICAL PROCESS TYPE	DATA ACTED ON
LEVEL #1	THE SEARCH FOR A SATISFACTORY MODEL AND THE SELECTION OF THE NEXT PARTIAL MODEL TO WORK ON IS THE RESPONSIBILITY OF HL-STRATEGY	
LEVEL #2	CONTRACTOR	
LEVEL #3	VERIFIER	
TACTICAL LEVEL	CONSTRUCTOR	

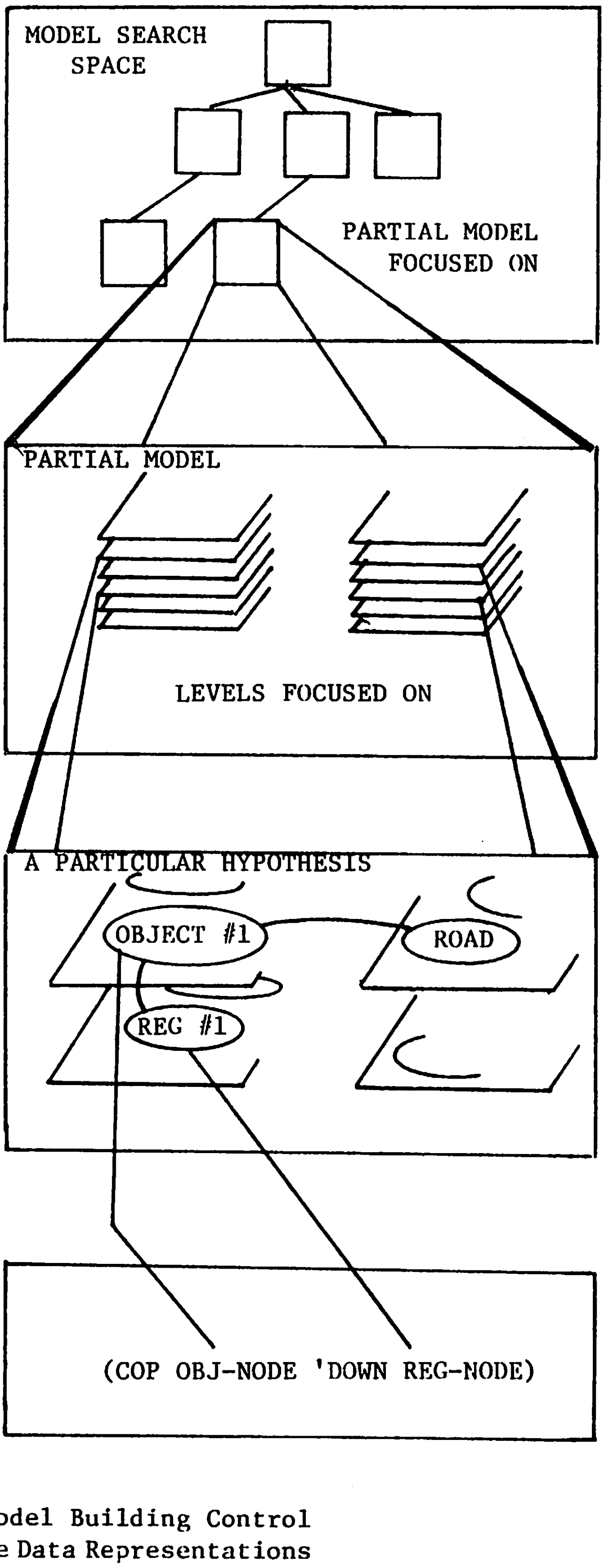


Figure 9 Link Between Model Building Control Strategy and the Data Representations

III.2 The Highest-Level Strategy

In VISIONS, this level is composed of the elements of level #2 of Figure 8. The two primary elements are FOCUS and CONTRACTOR. To some extent the decision of which CONTRACTOR to use should be based on the state of the model search space. Thus, tradeoffs and compromises of efficiency, accuracy, time, computational resources, the bushiness of the search tree, and other related issues are handled by the HL-STRATEGY (see Figure 9). It is at this highest level that access is available to the entire search space. The CONTRACTORs are applied to a particular partial model in that space, and are affected by information local to that partial model. The HL-STRATEGY must specify the procedure by which the CONTRACTOR is selected, much as problem solving A.I. systems select operators to reduce the complexity of a problem [HAY76,KUI75,NEW73].

III.3 CONTRACTORs

The highest level strategy is concerned with applying hypothesize-test-build processes which will produce some improvement in the state of the model, eventually bringing it to a defined satisfactory state of completion. These hypothesize-test-build processes are called CONTRACTORs. Their responsibility is to make some incremental change to the model. This change consists of adding nodes and arcs in the appropriate planes to assert the instantiation of an object, surface, frame or relationship.

This modularization of strategy into the HL-STRATEGY and that of the CONTRACTORs simplifies and clarifies the highest levels of model-building control. Whereas the HL-STRATEGY is a group of commands which is inherently tied to the model search space, the commands themselves are interpreted as contracts, much as a house builder would call upon a number of contractors in a specific order, perhaps changing the order as needed. As long as the contractors are successful and not too costly, the house continues to be built. Because there may be many types of houses to be built to different specifications, and because there are a wide class of scenes to be interpreted by VISIONS, SYMBOLS allows an arbitrary number of CONTRACTORs to be defined.

III.4 CONTRACTORs in VISIONS

It is the CONTRACTOR level of process which specializes in a model-building technique resulting in a new partial model. Each CONTRACTOR is identified by the levels between which it operates in the layered graph structure (Sec. II.3). For instance, a CONTRACTOR which instantiates an object, based on characteristics of a given set of regions would be called an R-to-O CONTRACTOR. This name characterizes it as a bottom-up process (Region-to-Object is 'up' in the layered graph, R-to-O corresponds to bottom-up, O-to-R corresponds to top-down processing).

There may be an arbitrary number of CONTRACTORs of any type, for instance R-to-O#1, R-to-O#2, etc., each with its own special characteristics. One R-to-O CONTRACTOR might generate a hypothesis based on color alone for a high speed crude instantiation, while another R-to-O CONTRACTOR might use all available features including time-costly shape analysis to generate its hypotheses.

III.5 FOCUSers, GENERATORs, FILTERs, and VERIFIERs

CONTRACTORs are not atomic in the taxonomy of model-building procedural knowledge. Four major types of strategical processes and two types of tactical[1] processes are available in SYMBOLS. The four major divisions of a CONTRACTOR are <u>FOCUS</u>, <u>GENERATOR</u>, <u>FILTER</u>, and <u>VERIFIER</u>.

FOCUSers are the process types which select an area to examine on a particular level of the partial model. Its results are given to a

[1] Tactical processes have no decision making elements, hence the distinction between tactical and strategical processes.

GENERATOR. The particular FOCUSer chosen depends on the goal of the CONTRACTOR. For instance, an R-to-O CONTRACTOR might have a FOCUS process which selects according to brightness or hue a region or group of regions to use as a basis for hypotheses.

FOCUS is not restricted to the third level of strategy. It might be applied at any point in a strategy where selection is required. For example, the HL-STRATEGY uses a FOCUSer to select the next partial model to expand.

GENERATORs are used to generate new hypotheses based upon a set of instantiations selected by a FOCUSer. An R-to-O GENERATOR could produce a list of objects which might be obtained by indexing into the Long Term Knowledge based on the attributes of a set of regions. This list of objects could be large, and hence would need to be FILTERed.

FILTERs are designed to weigh the relative merit of a number of hypotheses or tested hypotheses. FILTERs might be applied at any point in a strategy where the number of items in a set need to be reduced. A simple filter might threshold a set of hypotheses according to some visual (syntactic) feature. A more sophisticated filter might select those hypotheses which are of distinct 'classes' of 'types' so that subsequent verification will yield the most valuable information. For instance, the four hypotheses for objects describing some set of regions might be 'HOUSE', 'CAR', 'BUILDING', and 'FACTORY'. This set could be FILTERed to 'CAR' and 'BUILDING'. Eventual verification would then discriminate between widely different classes, giving a greater chance of a near hit with minimum effort.

The process of verification can be viewed simply as pattern recognition where a VERIFIER is a specialized classifier. However, VERIFIERs are expected to be somewhat more powerful than a typical linear pattern classifier because they will have available the entire graph structure, both the model under construction and the a priori knowledge about the domain.

If an Object VERIFIER (previously referred to in VISIONS [HAN75a] as "vision routines") is called upon to determine the validity of the hypothesis that a region R2 is a "TREE", the VERIFIER has at its disposal not only the features of R2 and surrounding regions, but also a definition of "TREE". This definition includes variations of shape, color, and organization of the parts of "TREE". The VERIFIER may inspect the FRAME level of the graph to discover any settings which imply "TREE". It might request a new segmentation from the low-level VISIONS system, or during development stages, we might specify a request for guidance from the user.

The result of applying an OBJECT VERIFIER is the rough probability that the particular object exists, and the naming of regions or surfiaces which it identifies. This information will be very important in the computation of the confidence of any given partial model, which in turn can be used by the HL-STRATEGY when deciding which partial model to expand next. Other regions or surfaces than those specified in the request might be necessary for the identification of the object. If so, the VERIFIER reports them. Additionally it could discover that if some feature were different, the result would be significantly altered. Such a report could aid the CONTRACTOR.

As with CONTRACTORs, any number of VERIFIERs, FILTERs, FOCUSers, and GENERATORs may be defined in SYMBOLS. The choice of which particular ones to apply rests within the CONTRACTOR. There are many types of VERIFIER, FILTER, FOCUS, and GENERATOR strategies which could be specified. Some will be more time-efficient, others more accurate in the analysis and value returned.

III.6 VERIFIERs in VISIONS

VISIONS requires a number of VERIFIERs to carry out the various CONTRACTOR functions. At present these include verification of Frames, Objects, and Surfaces. There will be categories under the heading of surface VERIFIERs for shadow, occlusion, and perspective which will aid a CONTRACTOR in determining the three-dimensional relationships of regions.

Similarly, there will be a number of object VERIFIER types. For instance, a syntactic feature matcher might take a simple logical match between the stored attributes for an object and the features characterizing a region. Shape analysis for objects would perform geometric manipulations for a best fit (rotation and scaling). Other types of VERIFIERs are planned. Because they are independent (except as viewed by a CONTRACTOR) they may be written and developed independently from each other, but in concert with the requirements of the CONTRACTORs in which they are to be used.

III.7 TACTICAL Processes

There are two more process types which complete the model-building process specification. They are called CONSTRUCTORs, and INSPECTORs. These tactical processes change and retrieve information from visual knowledge respectively. They contain no decision making elements, but rather serve other processes, making the complexity of the structure transparent. Because they contain no decision making components, they are considered tactical rather than strategical in nature. A typical INSPECTOR might find the ordered list of segments which constitute a region's boundary; a typical CONSTRUCTOR might instantiate an object which a CONTRACTOR has determined exists. The application of a CONSTRUCTOR has the effect of adding a hypothesis to the model, and hence a new model to the model search space. Thus, CONSTRUCTOR application signifies the completion of a CONTRACT and indicates return of control to the High-Level STRATEGY.

IV. MODEL SEARCH SPACE: A Tree of Partial Models

Search is a basic issue confronting any model-building system. In particular one must decide how best to represent the search space and control the search. First, let us consider alternative ways in which related partial models can be represented. At one extreme there could be an entire structure for each partial model generated during the search. This is not space efficient, considering that each partial model varies only by some small amount from its parent. At the other extreme is the collapsed model where any change replaces old data, thus no partial models are saved and only one updated model exists [SUS72,ERM75]. Another approach to model-building search spaces is that of saving at each node in the search tree only those data which have changed [McD74].

IV.1 Search Issues

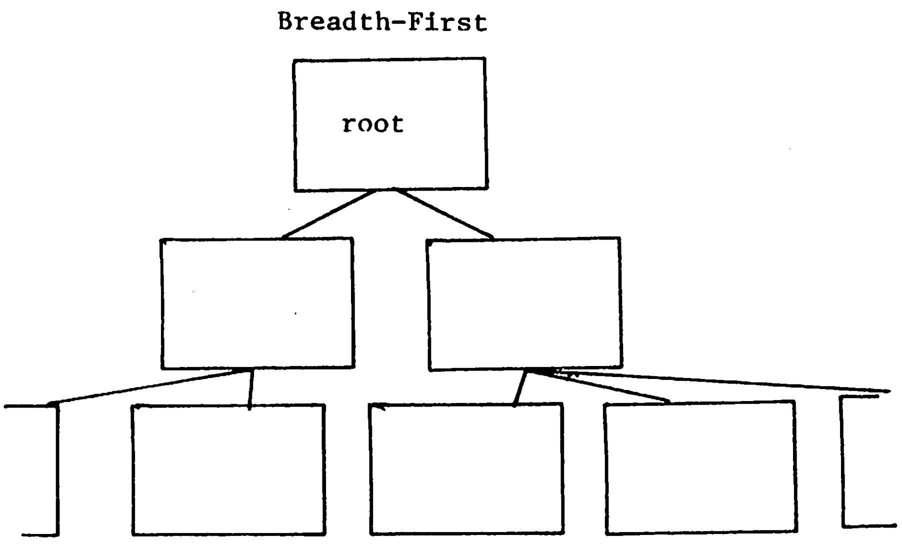
Error recovery and backtracking are two important problems in a search. Several issues arise from these problems. A) Is the process of backtracking easy and efficient? This is related to the implementation of the search space representation; in some cases [ERM74,WAL75,TEN76] search proceeds without backtracking. B) Is the error correctable without backtracking? If provision is made to store the interdependencies of strategic decisions in the search space, then it would be possible to tell by inspection if a previous erroneous instantiation propagated other instantiations. If it did not, then such an error could be corrected by 'un-doing' the error rather than going back and regenerating the search tree from the error point down. C) When returning to the state where an error occurred, how much computation is going to be redundant? If the first hypothesis a CONTRACTOR made was incorrect, but the second one was correct, is it possible to select the next hypothesis or must the HL-STRATEGY evoke the proper CONTRACTOR and redundantly regenerate the list of hypotheses?

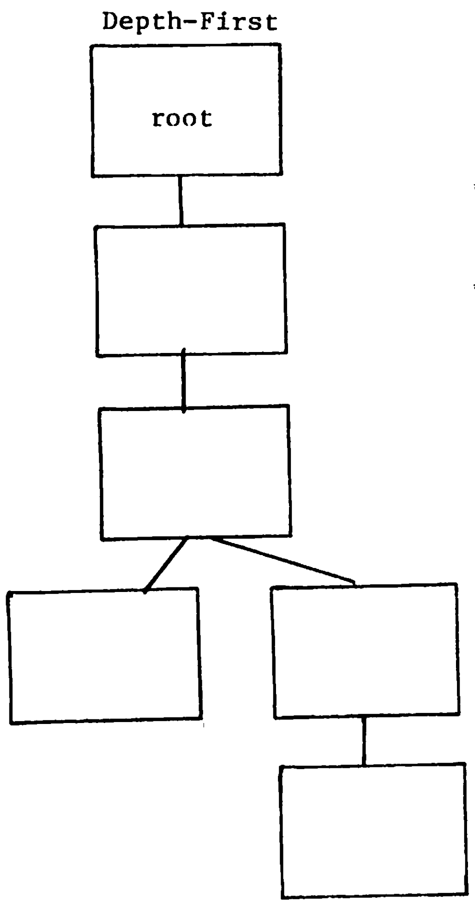
The problems of backtracking have led to a sequence of AI languages. Gross search inefficiency resulted when automatic backtracking was employed [SUS72]; and user control of backtracking has turned out to be a nontrivial task [FAH74]. We address this problem by storing the history of the search process. At any point (partial model) in the model search space, the H-L STRATEGY will be able to access the history of the decisions (what, when, and why) leading to that partial model (refer to Figure 9).

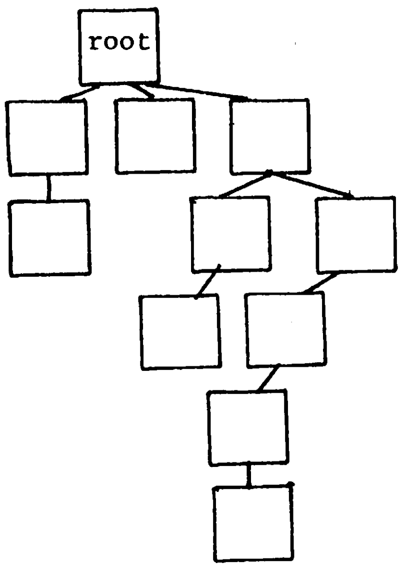
IV.2 Search Space Representation

The search for an acceptable model is conducted as a tree search in SYMBOLS. In this representation each node in the search tree is a partial model, and therefore implicitly is an entire multi-leveled graph (Sec. V.). The complete tree of partial models represents the state of the entire search for a satisfactory model (see Figure 10). The user is permitted the flexibility of controlling the bushiness of the search. For instance, by giving the same value of confidence (explained below) to each partial model, the search will proceed depth-first. If the user demands that all hypotheses be tried at each step of the search it will proceed breadth-first. If the user wishes he may collapse the search tree to a single node, thereby retaining no search history. This degenerate configuration has certain advantages, such as allowing the system to observe the relationship between conflicting hypotheses (of different paths of the search tree), as is evident in HEARSAY's Blackboard [LES75b].

Among other data (see Figure 11), a heuristic value of importance is associated with the partial model at each node. This value is meant to represent four characteristics of the model: 1) Confidences of each instantiation, 2) The







Regardless of type of search, the search space is available to the HL STRATEGY. This allows the bushiness of the search to be controlled at the appropriate strategic level.

Figure 10 The State of the Search for the Best Model is Available to the HL STRATEGY

This allows arbitrary control of bushiness and backtracking for error recovery.

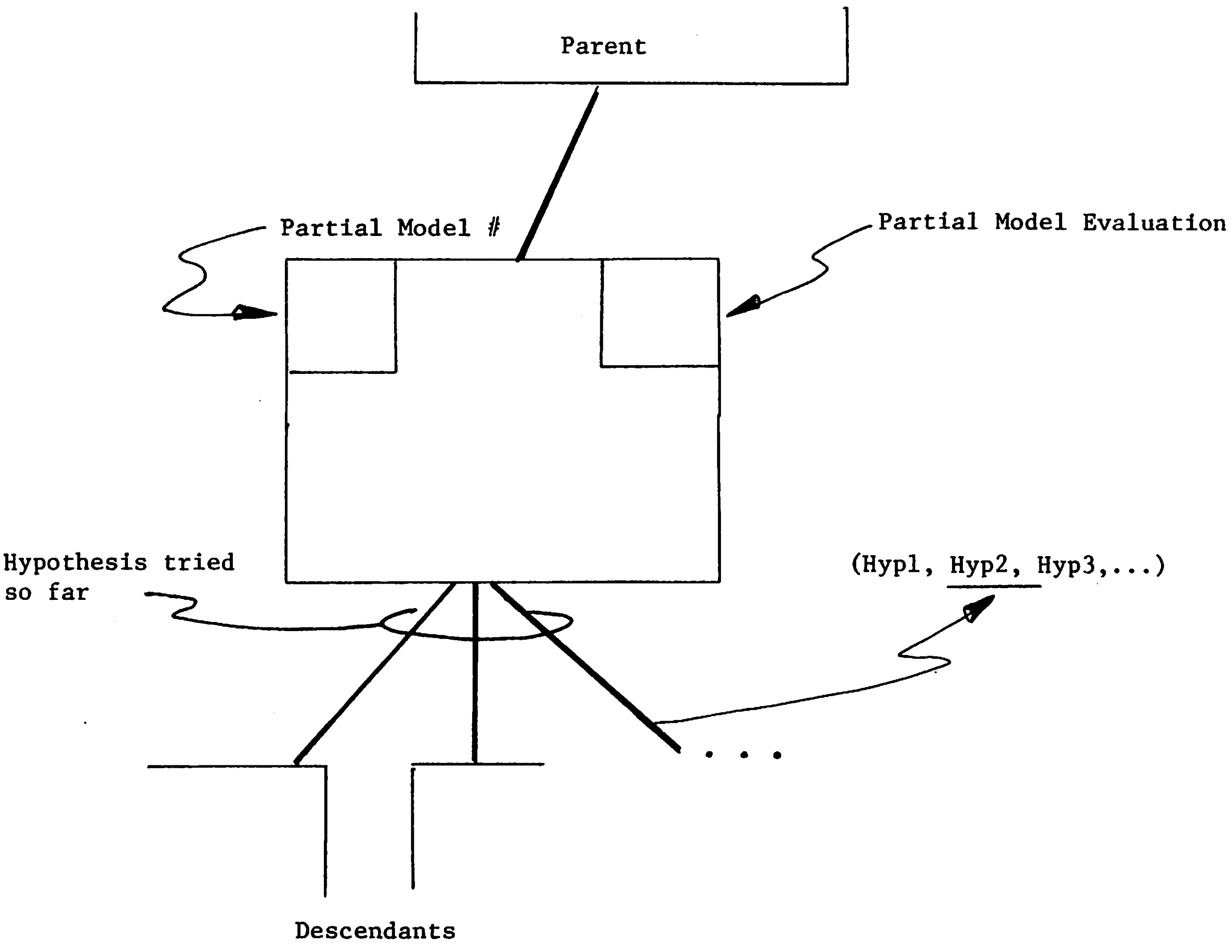


Figure 11 Information Related to a Partial Model Node in Model Search Space

The information shown here is stored explicitly in the search space representation.

consistency of the model, 3) The completeness of the model, and 4) The likelihood that the model can be completed. These values are generated after a CONTRACTOR has successfully completed its task.

This value can act as a heuristic to bias the model-builder toward the more profitable portions of the search. In our initial efforts, nodes will be investigated in order of heuristic value. In the case where more than one node has the greatest value, the most recently generated one will be chosen[1]. This technique gives the default of biasing the strategy to perform a more depth-first search, particularly when the heuristics are conservatively employed in differentiating between partial models. Although it is likely that some model heuristics will, in general, produce bushier trees, it is intended that VISIONS should produce final models without exploring a large number of paths.

IV.3 Strategy Related Data in the Search Space

SYMBOLS provides a storage facility in the search space for information related to the strategic processing which was attempted and performed in the generation of each partial model. These data should be useful both to give the strategy a history and to improve error correction/recovery (see Figures 10 and 11). The history is useful to 1) dynamically control the bushiness of the search, 2) relate inter-dependent instantiations, and 3) decide what level of process to invoke when returning to a partial model.

Although the initial VISIONS system has not made use of these powerful recovery facilities, we expect that eventually it will. This will be

[1] This can be relaxed to "approximately" the greatest node, where approximately means within some constant Δ of the max so that recent strong models will tend to be expanded.

achieved by defining a special FOCUSer which will be allowed to visit (if necessary) all nodes in the search space and return to the STRATEGY an information packet which informs the HL-STRATEGY of its new situation.

These techniques have not been tested and we will only sketch their operation as we envision it to work. Dynamic control of the bushiness of the search is not examined further since this control involves interactions of all strategic process elements and is related to the number of errors made.

Error recovery in a space with propagated errors is examined in the following (see Figure 12). Assume that node #2 is an error, that this node represents the identification of a region as 'ROAD' when it really is the trunk of a tree. Let us also assume that this error is indirectly implied by the low evaluation at node #6 where the 'CROWN' is identified correctly. The low evaluation resulted from the conflict between 'ROAD' and the parts/whole relation 'CROWN','TRUNK'/'TREE'. In this case there were no instantiations based on the error generated at node #2. The HL-STRATEGY could FOCUS on #2 through #6 and return to the HL-STRATEGY the two suggestions: a) to flag node #2 as an error, and b) to undo the effects of the error (shown as $\#2^{-1}$) at node #6.

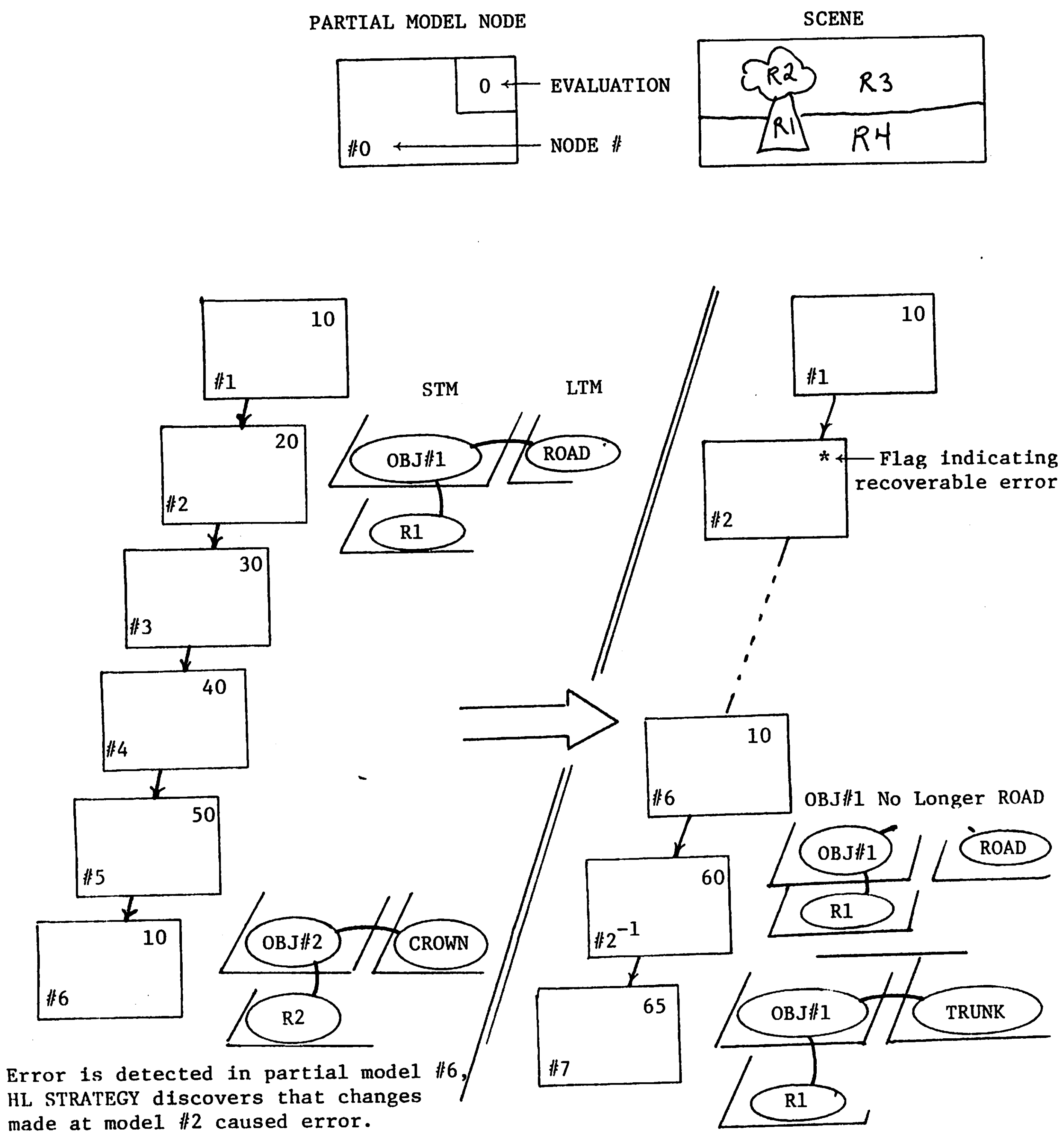


Figure 12 A Recoverable Error is One Which Did Not Propagate Other Errors

Figure 13 shows a different situation, in this case sufficiently many nodes have dependencies to the mode in error, so it would not be efficient to 'un-do' the error. The STRATEGY is presented with the old partial model at node #10. It is not sufficient to place the model-builder at that node because the STRATEGY is likely to do exactly what it did before. Obviously it must have access to the previously generated alternatives of the node at which it is placed. Additionally, it would be helpful to know which CONTRACTOR was applied, and what its hypotheses were. It is the HL-STRATEGY which must decide whether to apply a different CONTRACTOR or merely select a different hypothesis for instantiation.

The initial VISIONS implementation will use the default mechanism of first exhausting the hypotheses before trying a different CONTRACTOR. To this end a packet of data is left at each node in the search space. This data consists of the name of the CONTRACTOR applied, a list of the first N hypotheses, and the name of the item instantiated. Eventually we expect to develop the necessary FOCUSers to provide a packet telling the STRATEGY if it should proceed with the previously generated hypotheses or seek new ones.

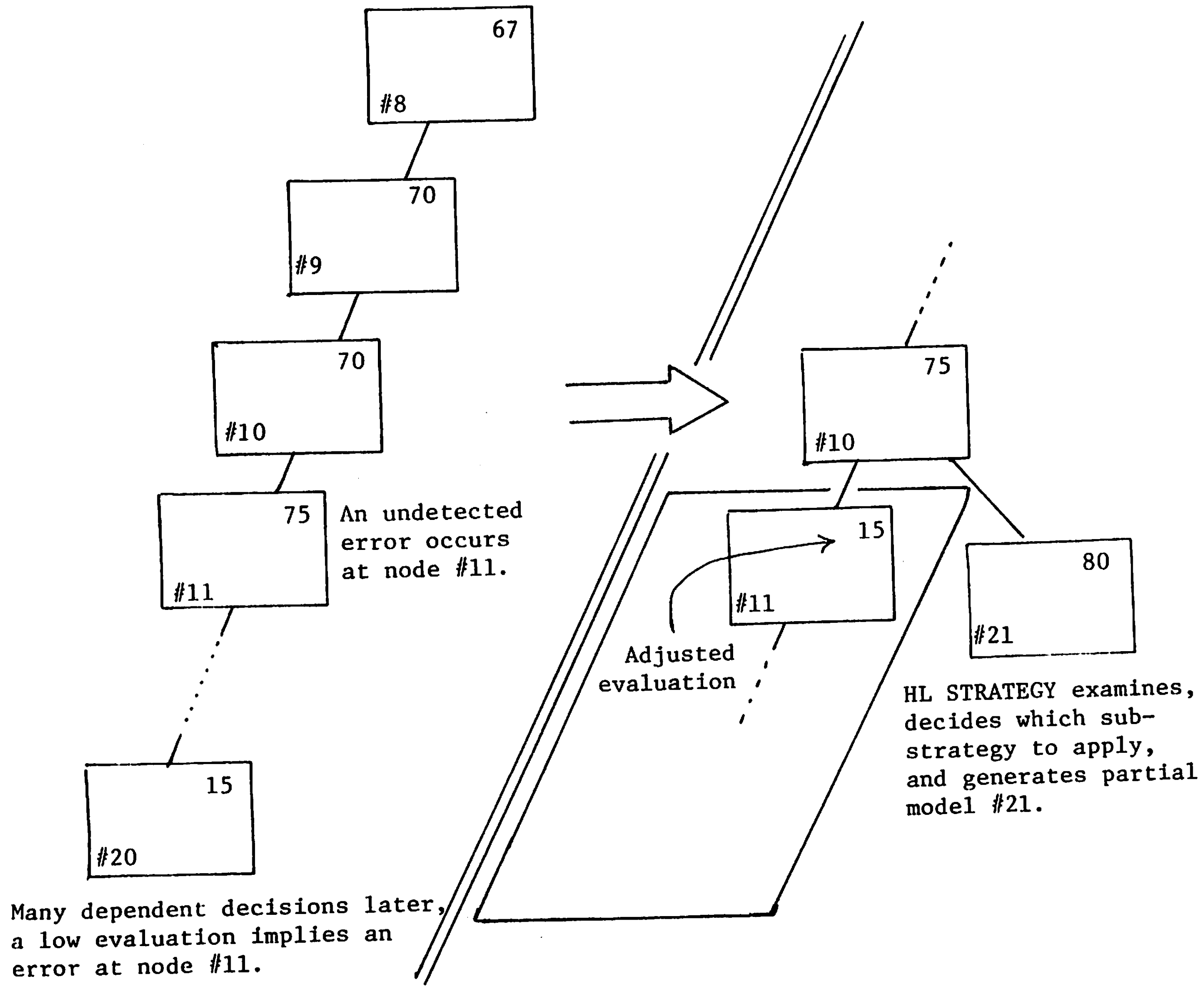

67
#8
70
#9
70
#10
75
#11
An undetected error occurs at node #11.
15
#20
Many dependent decisions later, a low evaluation implies an error at node #11.
75
#10
15
#11
Adjusted evaluation
80
#21
HL STRATEGY examines, decides which sub-strategy to apply, and generates partial model #21.

Figure 13 Recovery When Excessive Propagation of an Error Exists

V. IMPLEMENTATION

The processes of SYMBOLS deal with symbolic data which is best understood as a layered graph when viewed from any node in model search space. These three related structures (processes, layered graphs, and a search space) are implemented in a language developed specifically for SYMBOLS. This language combines characteristics of several AI languages built upon LISP. The interactive nature of LISP puts the user in a situation conducive to experimentation, while the power of SYMBOLS' primitives allows him to alter strategies easily.

SYMBOLS combines GRASPE and portions of CONNIVER making available the data types most convenient for the implementation. GRASPE [FRI69,PRA71], a graph processing language, has been implemented with some additions to support interconnected levels. CONNIVER [McD74] provides a data structure in which the model search space is efficiently implemented. Any node in the search tree (other than the root node which contains the initial state) explicitly contains only those items which differ between it and its parent (Figures 11 and 14) and the strategy-related information. Each node implicitly contains the initial state with all changes along the path from the root.

The tactical commands available to VISIONS through this language are quite powerful. This allows research effort to be concentrated on important strategic problems in scene understanding rather than requiring continuous programming overheads to deal with the complex data structure (see Figure 15).

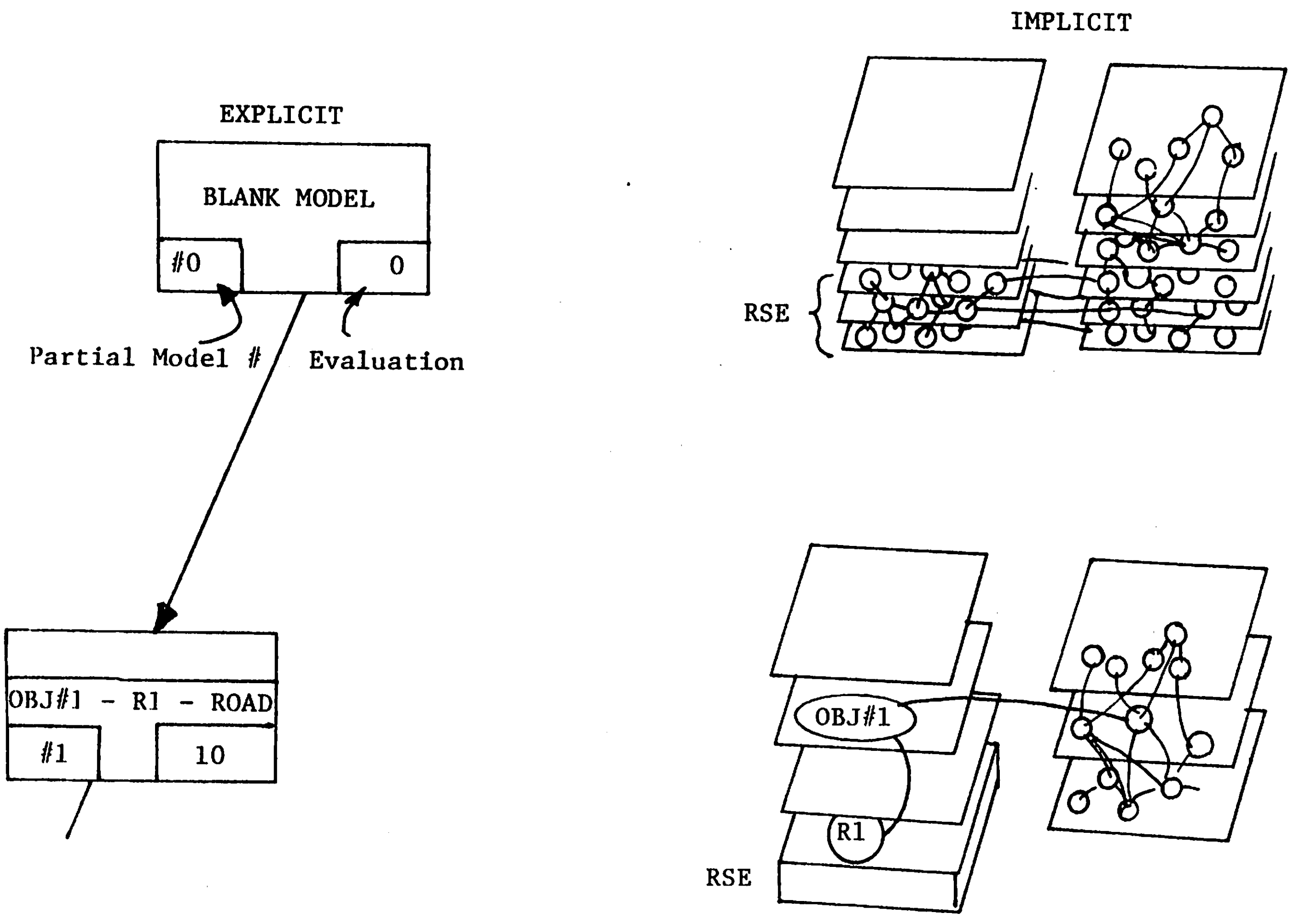


Figure 14 Changes Between the Ancestor and Descendant Nodes in the Model Search Space are Stored Explicitly

The entire partial model is stored implicitly at the partial model node. The path from a node to the root explicitly contains the partial model.

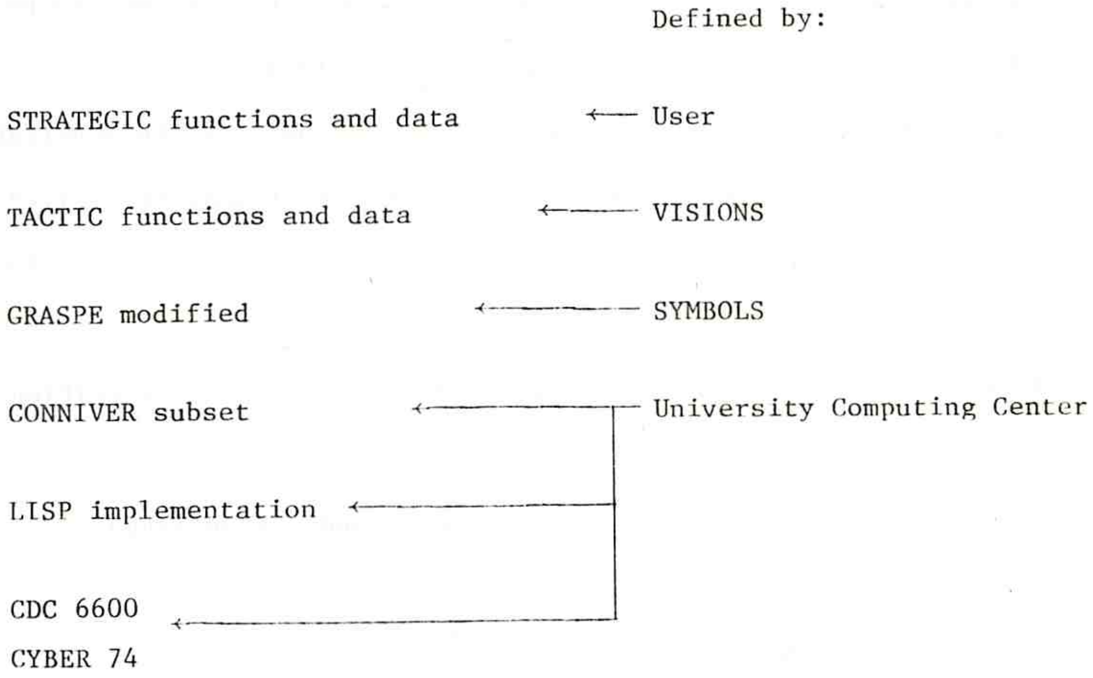


Figure 15 Levels of Implementation for High-Level VISIONS Research

Because the high-level system is implemented in an interactive environment, it is possible to develop the model-building components through the technique of incremental simulation [WOO73]. This allows the user to assume the role of any component whose input/output relationships have been specified. Thus, during development, the user can employ arbitrarily complex procedures limited only by the interface constraints imposed by the system design (criterion of modularity I.2). As the system evolves over time, it will rely less on user-interaction and more on internal computational processes.

SYMBOLS is implemented on a CDC-6600/CYBER 74. As of this writing it includes:

1) a system for the manipulation of the extended hypergraph structure described;
2) an efficient model search tree;
3) a control mechanism for tree searching;
4) a number of CONTRACTORs, FOCUSers, GENERATORs, FILTERs, VERIFIERs, INSPECTORs and CONSTRUCTORs with facility for easily defining more;
5) an oversimplified default model-building HL-STRATEGY in which a user can experiment by augmenting the model-building components.

VI. SUMMARY

SYMBOLS was designed to offer the VISIONS system wide lattitude of expression as a model building tool, while maintaining clear responsibilities of control and process components. The criterion of modularity was applied throughout the initial implementation of VISIONS resulting in the representation of data and processes as multi-leveled structures.

Visual world knowledge is expressed in levels representing visual syntax (2D primitives of endpoints, line segments, and regions), surfaces, objects, and frames. Image-specific information extracted from a particular scene is stored in the same multi-level representation. Arcs between nodes on planes of the world knowledge and image specific knowledge indicate instantiations of stored concepts as elements of the model which has been constructed.

The model building processes are divided according to strategic levels. At the highest level decisions are made to select which partial model (from the search space of all partial models) to expand next. At lower levels of strategy, hypotheses are generated and tested, eventually making some incremental change resulting in a new partial model.

The search space contains information used by the highest level of strategy to aid in the selection of the best partial model to expand. It also should facilitate the use of intelligent backtracking techniques which can be sensitive to the dependencies that one decision might have on others.

REFERENCES

[BAJ73] R. Bajcsy, "Computer Description of Textured Surfaces," Proc. of 3rd IJCAI, August 1973, 572-579.

[ERM74] L. D. Erman, Contributed Papers of the IEEE Symposium on Speech Recognition, April 1974, Pittsburgh, PA, IEEE #74CH0878-9AE.

[ERM75] L. Erman and V. Lesser, "A Multi-Level Organization for Problem-Solving Using Many, Diverse, Cooperating Sources of Knowledges," Proc. 4th Int. Joint Conf. on Artificial Intelligence, September 1975, pp. 483-490.

[FAH74] S. E. Fahlman, "A Planning System for Robot Construction Tasks," Artificial Intelligence, Vol. 5, pp. 1-49, 1974.

[FAH75] S. E. Fahlman, "A System for Representing and Using Real-World Knowledge," Massachusetts Institute of Technology AI-Memo 331, 1975.

[FRE61] H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," IRE Trans. on Electronic Computers, EC-10, June 1961, pp. 260-268.

[FRI69] D. P. Friedman, D. C. Dickson, J. J. Fraser and T. W. Pratt, GRASPE 1.5 A Graph Processor and its Applications, University of Houston, 1969.

[HAN74] A. Hanson and E. Riseman, "Preprocessing Cones: A Computational Structure for Scene Analysis," Computer and Information Science Department Technical Report 74C-7, University of Massachusetts, September 1974.

[HAN75a] A. Hanson and E. Riseman, "The Design of a Semantically Directed Vision Processor (Revised and Updated)," Computer and Information Science Department Technical Report 75C-1, University of Massachusetts, February 1975.

[HAN75b] A. Hanson, E. Riseman and P. Nagin, "Region Growing in Textured Outdoor Scenes," Computer and Information Science Department Technical Report 75C-3, University of Massachusetts, February 1975.

[HAN76a] A. Hanson, E. Riseman and T. Williams, "Constructing Semantic Models in the Visual Analysis of Scenes," Proceedings of the IEEE Milwaukee Symposium on Automatic Computation and Control, April 1976, pp. 97-102.

[Han76b] A. Hanson and E. Riseman, forthcoming paper on Processing Cones, November 1976.

[HAN76c] A. Hanson and E. Riseman, "A Progress Report on VISIONS: Representation and Control in the Construction of Visual Models," Computer and Information Science Department Technical Report 76-9, July 1976.

[HAY76] F. Hayes-Roth and V. R. Lesser, "Focus of Attention in a Distributed-Logic Speech Understanding System," Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, Technical Report, 1976.

[KUI75] B. Kuipers, "A Frame for Frames," Representation and Understanding (D. Bobrow and A. Collins, Eds.), Academic Press, 1975.

[LES75] V. R. Lesser, "Parallel Processing in Speech Understanding: A Survey of Design Problems," in Speech Recognition: Invited Papers of the IEEE Symposium (D. R. Reddy, Ed.), Academic Press, 1975.

[MCC62] J. McCarthy, P. W. Abrahams, D. J. Edwards, T. P. Hart and M. I. Levin, LISP 1.5 Programmer's Manual, MIT Press, Cambridge, MA, 1962.

[McD74] D. McDermott and C. Sussman, "The CONNIVER Reference Manual," Massachusetts Institute of Technology Memo 259a, January 1974.

[MIN75] M. Minsky, "A Framework for Representing Knowledge," in The Psychology of Computer Vision (P. Winston, Ed.), McGraw-Hill, 1975, pp. 211-277.

[NEW73] A. Newell, "Production Systems: Model of Control Structures," in Visual Information Processing (William G. Chase, Ed.), Academic Press, 1973.

[OHL75] R. Ohlander, "Analysis of Natural Scenes," Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA, April 1975.

[PRA71] T. Pratt and D. Friedman, "A Language Extension for Graph Processing and its Formal Semantics," Communications of the ACM, 4, 1971.

[QUI68] R. Quillian, "Semantic Memory," Semantic Information Processing (M. Minsky, Ed.), MIT Press, 1968.

[ROS71] A. Rosenfeld and M. Thurston, "Edge and Curve Detection for Visual Scene Analysis," IEEE Trans. Computers, 1971, pp. 562-569.

[ROS76] A. Rosenfeld and A. C. Kak, Digital Picture Processing, Academic Press, 1976.

[RUM75] D. E. Rumelhart, "Notes on a Schema for Stories," in Representation and Understanding (D. Bobrow and A. Collins, Eds.), Academic Press, 1975.

[SIM69] H. A. Simon, The Sciences of the Artificial, MIT Press, Cambridge, MA, 1969.

[SIM73] R. F. Simmons, "Semantic Networks: Their Computation and Use for Understanding English Sentences," in Computer Models of Thought and Language (R. C. Schank and K. M. Colby, Eds.), H. Freeman & Co., 1973.

[SUS72] G. J. Sussman and D. V. McDermott, "From PLANNER to CONNIVER--A Genetic Approach," Proc. of 1972 FJCC, pp. 1171-1179, 1972.

[TAN75] S. Tanimoto and T. Pavlidis, "A Hierarchical Data Structure for Picture Processing," Computer Graphics and Image Processing, June 1975.

[TEN76] J. M. Tenenbaum and H. G. Barrow, "Experiments in Interpretation-Guided Segmentation," Technical Note 123, Artificial Intelligence Center, Stanford Research Institute, 1976.

[UHR72] L. Uhr, "Layered 'Recognition Cone' Networks that Preprocess, Classify, and Describe," IEEE Trans. Computers, 1972, 758-768.

[WOO73] W. Woods and J. Makhoul, "Mechanical Inference Problems in Continuous Speech Understanding," Third International Joint Conference on Artificial Intelligence, August 1973, pp. 200-202.

[WOO75] W. Woods, "What's in a Link," in Representation and Understanding (D. Bobrow and A. Collins, Eds.), Academic Press, 1975.

[YOR76] B. W. York, private communications concerning research in progress, December 1976.