NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Memorandum 33-773

DABI—A Data Base for Image Analysis With Nondeterministic Inference Capability

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

May 15, 1976

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 33-773	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle DABI--A DATA BASE FOR IMAGE ANALYSIS WITH NONDETERMINISTIC INFERENCE CAPABILITY		5. Report Date May 15, 1976 6. Performing Organization Code
7. Author(s) Y. Yakimovsky, R. Cunningham		8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91103		10. Work Unit No. 11. Contract or Grant No. NAS 7-100 13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A description is given of the data base used in the perception subsystem of the Mars robot vehicle prototype being implemented at the Jet Propulsion Laboratory. This data base contains two types of information. The first is generic (uninstantiated, abstract) information that specifies the general rules of perception of objects in the expected environments. The second kind of information is a specific (instantiated) description of a structure, i.e., the properties and relations of objects in the specific case being analyzed. The generic knowledge can be used by the approximate reasoning subsystem to obtain information on the specific structures which is not directly measurable by the sensory instruments. Raw measurements are input either from the sensory instruments or a human operator using a CRT or a TTY.

The generic rules of perception are substantially a model representing the statistics of the environment relating the properties of objects, and relations between objects. The description of a specific structure is also nondeterministic in the sense that all properties and relations may take a range of values with an associated probability distribution.

17. Key Words (Selected by Author(s))	18. Distribution Statement
Computer Programming and Software Computer Systems Cybernetics Statistics and Probability	Unclassified -- Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	41	

HOW TO FILL OUT THE TECHNICAL REPORT STANDARD TITLE PAGE

Make items 1, 4, 5, 9, 12, and 13 agree with the corresponding information on the report cover. Use all capital letters for title (item 4). Leave items 2, 6, and 14 blank. Complete the remaining items as follows:

3. Recipient's Catalog No. Reserved for use by report recipients.

7. Author(s). Include corresponding information from the report cover. In addition, list the affiliation of an author if it differs from that of the performing organization.

8. Performing Organization Report No. Insert if performing organization wishes to assign this number.

10. Work Unit No. Use the agency-wide code (for example, 923-50-10-06-72), which uniquely identifies the work unit under which the work was authorized. Non-NASA performing organizations will leave this blank.

11. Insert the number of the contract or grant under which the report was prepared.

15. Supplementary Notes. Enter information not included elsewhere but useful, such as: Prepared in cooperation with... Translation of (or by)... Presented at conference of... To be published in...

16. Abstract. Include a brief (not to exceed 200 words) factual summary of the most significant information contained in the report. If possible, the abstract of a classified report should be unclassified. If the report contains a significant bibliography or literature survey, mention it here.

17. Key Words. Insert terms or short phrases selected by the author that identify the principal subjects covered in the report, and that are sufficiently specific and precise to be used for cataloging.

18. Distribution Statement. Enter one of the authorized statements used to denote releasability to the public or a limitation on dissemination for reasons other than security of defense information. Authorized statements are "Unclassified—Unlimited," "U. S. Government and Contractors only," "U. S. Government Agencies only," and "NASA and NASA Contractors only."

19. Security Classification (of report). NOTE: Reports carrying a security classification will require additional markings giving security and downgrading information as specified by the Security Requirements Checklist and the DoD Industrial Security Manual (DoD 5220.22-M).

20. Security Classification (of this page). NOTE: Because this page may be used in preparing announcements, bibliographies, and data banks, it should be unclassified if possible. If a classification is required, indicate separately the classification of the title and the abstract by following these items with either "(U)" for unclassified, or "(C)" or "(S)" as applicable for classified items.

21. No. of Pages. Insert the number of pages.

22. Price. Insert the price set by the Clearinghouse for Federal Scientific and Technical Information or the Government Printing Office, if known.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Memorandum 33-773

DABI—A Data Base for Image Analysis With Nondeterministic Inference Capability

Y. Yakimovsky
R. Cunningham

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

May 15, 1976

PREFACE

The work described in this report was performed by the Science Data Analysis Section of the Jet Propulsion Laboratory.

CONTENTS

APPENDIXES

TABLES

FIGURES

ABSTRACT

A description is given of the data base used in the perception subsystem of the Mars robot vehicle prototype being implemented at the Jet Propulsion Laboratory. This data base contains two types of information. The first is generic (uninstantiated, abstract) information that specifies the general rules of perception of objects in the expected environments. The second kind of information is a specific (instantiated) description of a structure, i.e., the properties and relations of objects in the specific case being analyzed. The generic knowledge can be used by the approximate reasoning subsystem to obtain information on the specific structures which is not directly measurable by the sensory instruments. Raw measurements are input either from the sensory instruments or a human operator using a CRT or a TTY.

The generic rules of perception are substantially a model representing the statistics of the environment relating the properties of objects, and relations between objects. The description of a specific structure is also nondeterministic in the sense that all properties and relations may take a range of values with an associated probability distribution.

I. INTRODUCTION

The modeling of systems is the goal of most scientific disciplines and endeavors. A good generic (abstract or uninstantiated) model is compact and concise. It can be used to analyze the meaning of a particular state of the environment and to anticipate its behavior. The traditional method of modeling in the mathematical physical sciences is the use of a set of deterministic axioms expressed in mathematical logic. This approach was used successfully in generation and analysis of mathematical and physical models and theories. Unfortunately, attempts to use this method of representation by computer programs for automatic reasoning (Ref. 1; Ref. 2, Chaps. 6, 7, 8) ran into the problems of computational impracticability as was foreseeable in hindsight from complexity theory (Ref. 3). In addition to the inherent computational impracticability of the use of nontrivial axiom systems, most rules in the real world are nondeterministic. That is, for almost all rules, exceptions can be found.

Since logic-based (deterministic) rule systems were found to be impractical, attempts to implement practical nondeterministic inference systems were made (Refs. 4, 5).

This article presents our approach to the representation of nondeterministic generic rules, the representation of nondeterministic facts about the specific cases being analyzed, and the approximate reasoning process which applies the generic information to obtain more information on the specific cases. In some sense, we integrated semantic net approach with sequential decision theory. The semantic net approach contributed flexible hierarchical organization, while the sequential decision theory contributed flexibility by use of probabilities and systematic control structure.

Researchers in scene analysis used or suggested use of the data base approach (Ref. 6), the semantic base approach (Ref. 4) or deterministic hierarchical structures (Ref. 7, 8, 9) before. However, it appears that our system is a step forward in providing for powerful control and rule-based probabilistic inference under time constraints in a way which was not attempted before and which seems to be potentially and practically useful.

The data base is implemented in SAIL, an algol-based associative data base language for the DEC PDP-10 computer. The machine being used at the present is the Caltech PDP-10, but implementation started on the Stanford Artificial Intelligence Lab. PDP-10. The sensory instrument, e.g., the stereo pair of TV cameras

and laser range finder (Fig. 1) are controlled and interfaced to the real time minicomputer GA SPC-16/85 (see Fig. 2). The Caltech PDP-10 can control the SPC-16 via telephone line, and many of the preprogrammed attributes (feature extractors) are implemented as low-level image analysis routine on the SPC-16. The SAIL language is described in Refs. 10 and 11.

The sensory system itself and low-level routines for feature extraction out of images are described in Refs. 12 - 15. Appendixes A and B contain a self-contained description of a simple application of the data base in the analysis of the traversibility (safety) of area for the vehicle. It may be helpful to refer to these appendixes at this point.

II. ENVIRONMENT DESCRIPTION - SPECIFIC STRUCTURE REPRESENTATION

In the abstract, a specific structure (an analyzed case) can be represented as a set of objects, properties of those objects and relations between objects. Our domain of specializations is the representation and the analysis of pictorial information. Appendix 1 explains a computer printout of a generic model, and Appendix 2 describes a computer printout of a specific model. Both of these models are dedicated for the analysis of traversibity of areas by a robot vehicle. Typically, when modeling the VISUAL WORLD in an attempt to understand pictures there are the following classes of objects: (1) scenes (picture frames), (2) three-dimensional bodies, (3) two-dimensional regions (pictorial images of three-dimensional surfaces), (4) one-dimensional lines, (5) one-dimensional boundaries between regions, (6) vertices, and (7) 3-D surface sample point, etc.

In each specific instance of the environment (in vision research it will be a specific picture frame), there appear a few objects of those types. Each one of these objects has properties and is related to other objects. Figure 3 gives a partial data structure describing an image.

A. OBJECTS AND CLASSES

The list of possible "classes" of objects is predetermined by the repertoire of the generic model. When representing an instance structure, each "class" is a "set" data structure. Such a "set" is a data structure containing an unsorted list of nonrecurring pointers to the data structures representing the known objects of that class in the specific structure. For instance, the set

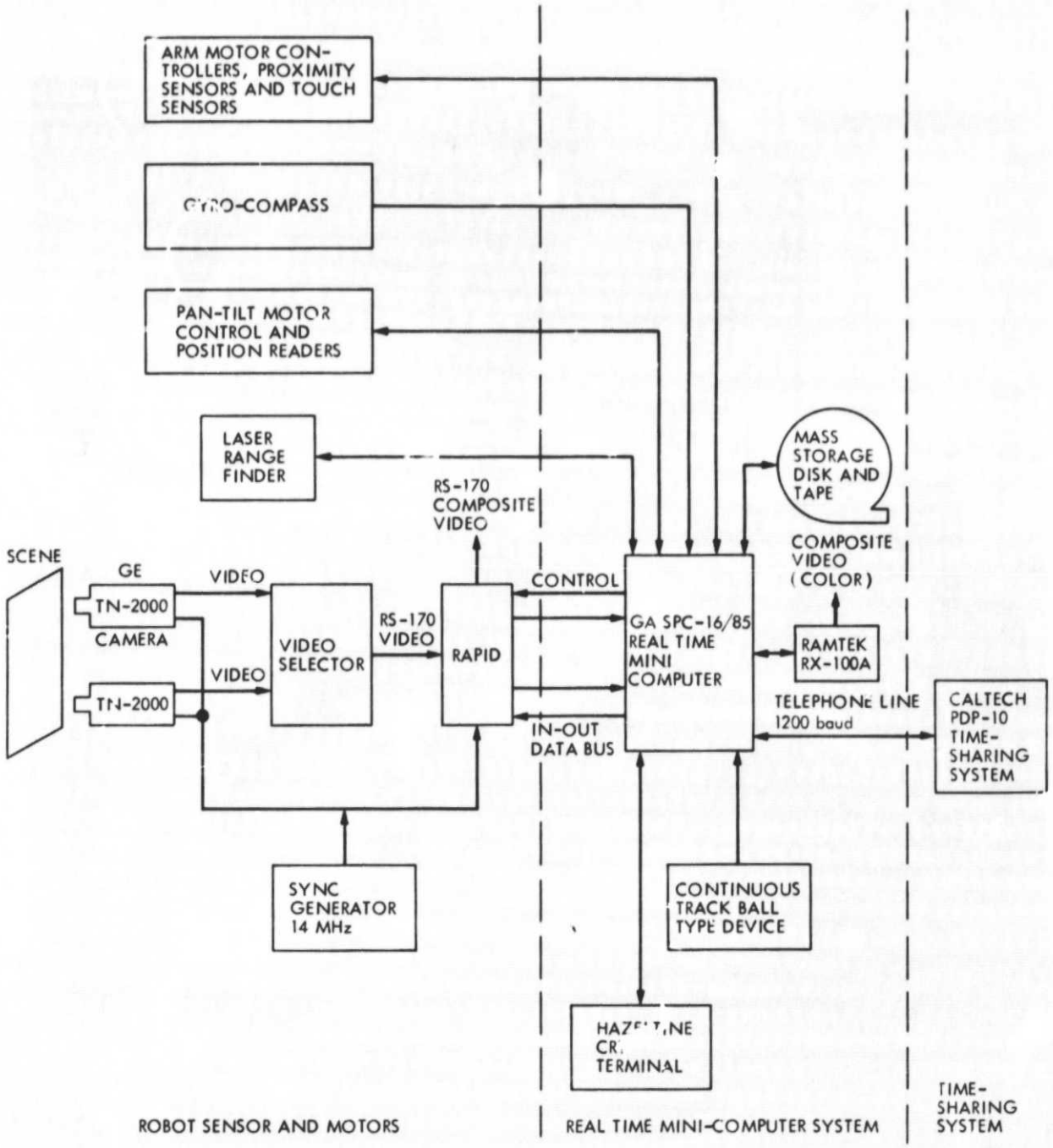


Fig. 1. Robot system configuration

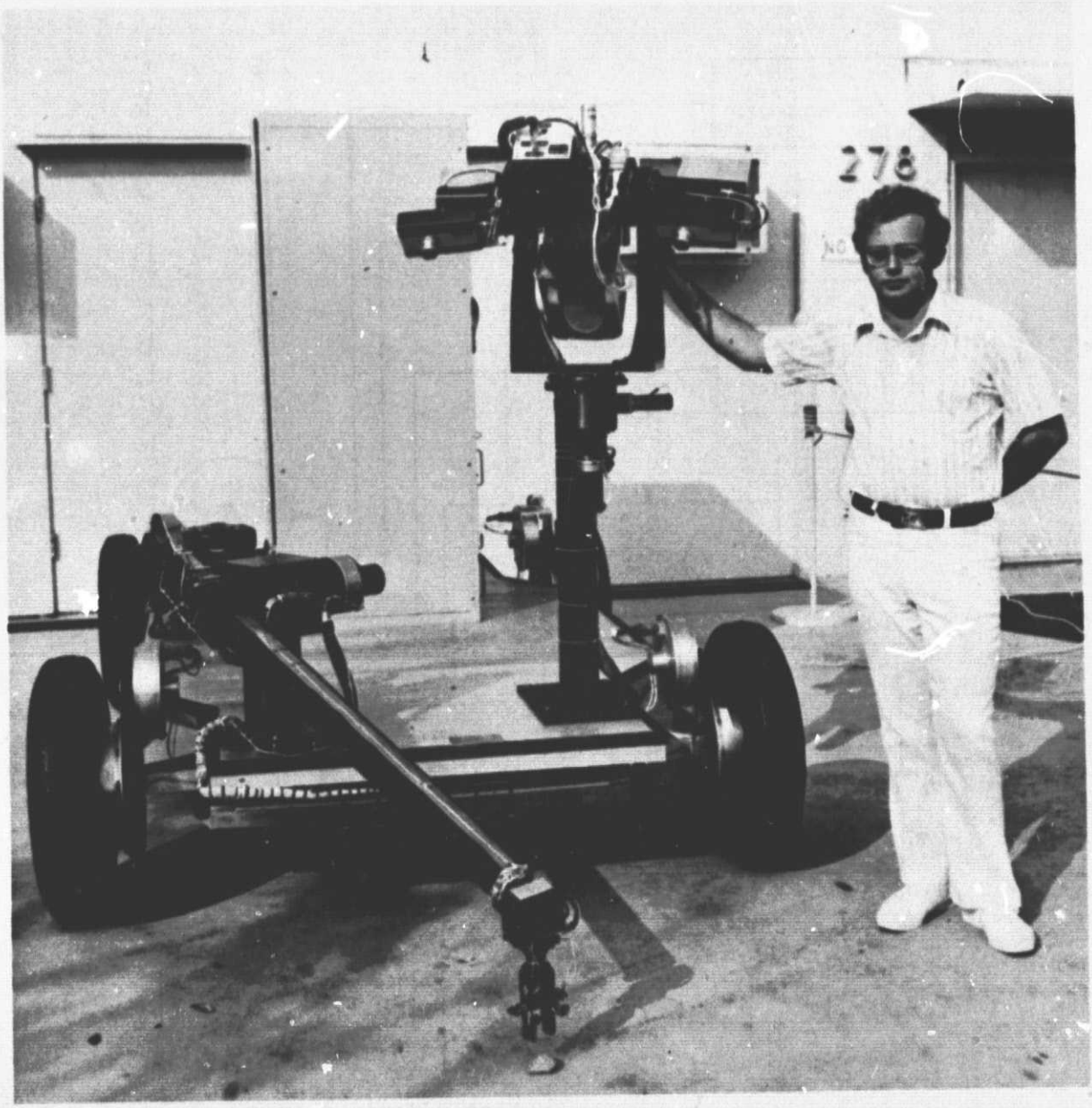
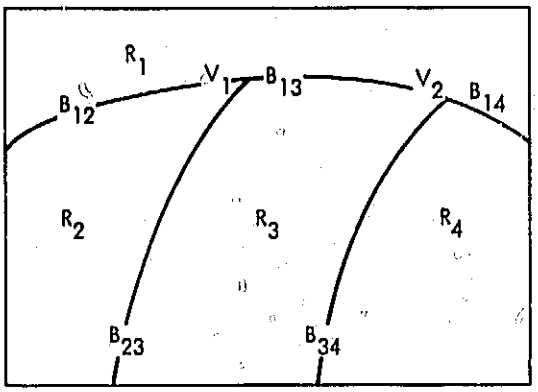


Fig. 2. Hardware configuration: two cameras and arm



CLASSES OF OBJECTS: SCENES, REGIONS, BOUNDARIES, VERTICES

OBJECTS

OF CLASS SCENES: P_1

OF CLASS REGIONS: R_1, R_2, R_3, R_4

OF CLASS BOUNDARIES: B_{12}, B_{13}, B_{14}, B_{23}, B_{34}

OF CLASS VERTICES: V_1, V_2

FEATURES

OF REGIONS: SIZE, GRAY LEVEL, LABEL
OF BOUNDARIES: LENGTH, AVERAGE DIFFERENCES

RELATIONS

REGIONS IN:	SCENES	→ REGIONS
BOUNDARY OF:	REGION	→ BOUNDARY
ADJACENT:	REGION	→ REGION
COMPOSES:	VERTEX	→ BOUNDARY

PROPERTIES:

$$\text{SIZE} \otimes R_1 \equiv (10 \;\; 1.0)$$

$$\text{LABEL} \otimes R_1 \equiv \begin{pmatrix} \text{"SKY"} & .5 \\ \text{"CLOUD"} & .5 \end{pmatrix}$$

RELATION - OBJECT LIST:

$$\text{BOUNDARY OF} \otimes R_1 \equiv \begin{pmatrix} B_{12} & 1.0 \\ B_{13} & 1.0 \\ B_{14} & 1.0 \end{pmatrix}$$

$$\text{COMPOSES} \otimes V_1 \equiv \begin{pmatrix} B_{13} & 1.0 \\ B_{23} & 1.0 \\ B_{12} & 1.0 \end{pmatrix}$$

$$\text{REGIONS IN} \otimes P_1 \equiv \begin{pmatrix} R_1 & 1.0 \\ R_2 & 1.0 \\ R_3 & 1.0 \\ R_4 & 1.0 \end{pmatrix}$$

Fig. 3. Partial representation of a segmented scene

associated with the class "regions" when representing the model in Fig. 3 will contain pointers to regions R_1, R_2, R_3, and R_4. The object data structure itself is an item containing the object's name (for man-machine interaction), the name of its class and some basic preprogrammed properties. Usually each object in a structure is in relation "part of" with a global object (the "scene" in the visual pictures case). The global object allows the definition of global properties like lens iris setting, camera position, and external lighting conditions which affects the analysis of all substructures.

B. REPRESENTATION OF PROPERTIES

The generic model of the environment defines a set of attributes that are functions that act on elements of a specified class and map into the integer domain. For example, the following are some of the attributes which apply to objects of class regions: "area," "color," "average light intensity," "shape." Attributes which apply to objects of type boundary include "length," "difference in light intensity across the boundary line," etc.

The value of an attribute when applied to an object is a property of the object. For instance, application of the attribute "area" to region R_1 may result in the value 7, which is a property of R_1.

The representation of properties is made a bit more complicated, because measurements may be unreliable and take a range of possible values. For instance, estimating the "label" (meaning) of the body imaged on R_1 may be speculative. To represent ambiguity, properties are allowed to take range of values with an associated probability estimate of the validity of that value.

The association of attribute, object, and property is represented by the associative data structure of SAIL, and may look like this:

$$\text{Area} \textcircled{X} R_3 = \left\{ \begin{matrix} 16 & 0.05 \\ 15 & 0.9 \\ 14 & 0.05 \end{matrix} \right\}$$

This will mean that the current estimate of the area of R_3 is 16 with probability 0.05, 15 with probability 0.9, and 14 with probability 0.05. Note, that the probabilities add up to one. This is so because the property values

are assumed to be mutually exclusive (e.g., one and only one value is actually true even though we cannot tell which one).

The properties data structure is further complicated by two factors. First, a property may change in time; hence, some estimate on the period of validity of that property must be given. Secondly, often the reliability of the estimate of a property depends on the resources spent in measurement and analysis. Hence, an indication of: (1) the amount of resources (compute time) allocated to obtaining the specific property, (2) the amount of resources actually used to obtain that estimate, and (3) the real time when the estimate was obtained is associated with the property estimate. As a result, the property data structure (the item's content, which is a linear array here) contains the following information:

(1) Start of validity period date.
(2) End of validity period date.
(3) Date when the estimate was computed.
(4) Resources allocated to obtain the property's estimate.
(5) Resources used to obtain the estimate.
(6) Last access date - used by the "forget" mechanism to release storage taken by unused information.
(7) Estimated values list.

$$\begin{bmatrix} val_1 & P_1 \\ val_2 & P_2 \\ . & . \\ . & . \\ . & . \\ val_n & P_n \end{bmatrix}$$

(more compact representation of the probability distribution of values is used when practical)

All dates are measured in absolute time, and stored in standard PDP-10 format in a 36-bit word specifying the date and time of day down to about a third of a second accuracy. If a property is assumed constant in the effective time, the validity dates may be set to $-\infty$ or to $+\infty$ to indicate permanency in one or two time directions in the scope of the analysis.

Resources are measured in the number of milliseconds of compute time that were used or were allocated to be used to obtain that estimate.

There can be more than one property associated with a pair of an attribute and a specific object. For instance, the color of an object may be blue from 06^{00} to 17^{35}, red from 17^{35} to 18^{30}, and black afterwards in the period of interest. The associative storage mechanism of SAIL provides for that type of multiplicity of associations of few properties with a single pair of an attribute and an object.

Attributes values (properties) and relation values (object lists) can be put in manually in the specific model data base as training samples for a learning process where the generic model is being improved or analyzed. Such properties are marked as "special" knowledge for training purposes. The learning process is not described at all within the scope of this article, and no further reference to that point will be made.

C. REPRESENTATION OF VALUES OF RELATIONS (OBJECTS LIST)

A number of relations are defined for each class of objects in the generic model. The value of a relation applied to an object of the appropriate class will be a list of objects all of one class (possibly different from the class of the object operated on) which satisfy the relation. For instance, "boundary of" will be a relation which operates on object of class "regions" and will come back with a list of objects of class "boundary lines" which are boundaries of that region.

The limitation that relations apply to one object at the time was imposed to simplify implementation.

In general, relations are n-ary, not binary, as in the present implementation; that is, the relation is between more than two objects. The limitation may be bypassed by defining "combination" objects; that is, define an object class whose elements stand for an ordered list of a fixed number of simpler objects.

Typically, application of a relation to an object results in the following association being put in the data base:

$$\text{Relation} \textcircled{X} \text{object}_0 = \begin{bmatrix} \text{object}_1 & W_1 \\ \text{object}_2 & W_2 \\ \text{object}_3 & W_3 \\ \vdots & \\ \text{object}_N & W_N \end{bmatrix}$$

The relation is the name as defined in the generic model. Object_0 is an instance of an object of the class of objects upon which the relation works. Each one of object_i $(1 \leq i \leq N)$ is estimated to satisfy the relations with the corresponding probability W_i. In fact, the relations values define fuzzy sets (Ref. 16). All of these objects belong to the same class of objects which is the range of the relation. Clearly, N varies with different applications of the relation. We also imposed the constraint that the W_i must be higher than 0.5; otherwise, the corresponding object is not included in the list to save storage. Note that the W_i's do not have to add up to one. There may be cases where more than one object satisfies the relation, e.g., the objects which satisfy a relation are not mutually exclusive. In other cases the list is empty; that is, no object appears to satisfy the relation. The distinction between operators which have mutually exclusive values (attributes) and others (relations) is important. Failure to account for these two types of operators results in mathematical logical distortions when the modeling process is bent to fit reality (Ref. 5).

As is the case with properties, the "list" of objects which satisfy a certain relation is time and resource dependent. The more the resources are expanded for searching for such objects, the more of them are likely to be found. Similarly, the period of time during which the relation will be valid may be limited. As a result, the data structure of the object list also contains the following information:

(1) Start of validity period date.

(2) End of validity period date.

(3) Approximated date when list was computed.

(4) Type of computation.

(5) Resources (compute time) allocated to obtaining the list.

(6) Resources used in obtaining the list.

(7) Last access date (used for storage cleanup).

There are two types of evaluations of relations. The first corresponds to the existential quantifier ($\exists$) in logic. This existential search tries to find an object which satisfies the relation with high enough probability as specified in the call, and it terminates successfully as soon as one is found. The second type of evaluation is an exhaustive search. Its aim is to find as many objects that satisfy the relation with high probability within the bound of the resources allocated. The object list contains also a mark as to which type of search was used to obtain it.

There is an indirect call on relations which return a property-like value. This is a filter call. It operates on a pair of objects ($object_0$, $object_1$) and it takes only two values, 0 or 1. It takes the value 1 if $object_1$ appears to satisfy the corresponding relatior to $object_0$ and 0 otherwise,

Example:

$$\text{Filter } (\text{Relation}_1, \text{object}_0, \text{object}_1, \text{res}) = \begin{bmatrix} 0 & 0.4 \\ 1 & 0.6 \end{bmatrix}$$

is equivalent to

$$\text{Relation}_1 \textcircled{X} \text{Object}_0 = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \text{Object}_1 & 0.6 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$$

III. THE GENERIC MODEL

The generic model does not describe an instance of the environment (a specific structure) but contains information on general rules that objects and structures in the environment will generally satisfy. The generic model is designed so as to allow direct use of the rules to compute properties and find objects which satisfy certain relations when this kind of information is requested by a user.

The basis of the knowledge representation in the generic model is, of course, the repertoire of: (1) classes of objects, (2) attributes of those objects, and (3) relations. With each attribute and relation, there is an associated algorithm which, when applied to an object of the adequate class, will come back with the value. These algorithms are constructed so that they may use only the resources (compute time) available to get an estimate of the value of the relation or attribute.

At the present the rules are man-made. Using a special interactive editor, experts generate and update the information contained in the rules expressed in the formats described later. The data base system was designed with the intent that it will be expandable to include a statistical learning subsystem. When the learning subsystem will be implemented, the rules will be largely machine generated. The learning subsystem will integrate principles of adaptive learning (Ref. 17) with statistical structure learning (Refs. 18, 19).

A. ATTRIBUTES

The data structure of each attribute contains information defining the class of objects it operates on and the range of (integer) values that it can take.

There is an option to associate a name (an alpha-numeric string) with some or all of the integer values that the attribute can take. These names are intended to facilitate man-machine interaction when properties are transmitted to or received from human operators. Similarly, the attribute names are selected so that they will be self-explanatory as to their semantic meaning.

There are three types of attributes as determined by the type of the unique algorithm that is associated with the attribute and which is used to compute the properties of objects. The first type is programmed. Here, a preprogrammed routine is used to compute the value of the attribute when applied to an object. Many of these properties are actually stored in the data structure of the object itself in which case they are not saved in the associative storage. These routines typically control directly the sensory instruments interfaced to the real time computer (SPC-16/85) and perform what we call low level analysis. We distinguish between preprogrammed attributes implemented on the SPC-16 and the PDP-10. The second type of attributes are those whose values are obtained from a human operator or experts. When the value of one of those attributes for an object (the property) is required, the system issues a console (teletype or CRT) message requesting those values from the operator. The text of the message with blanks to be filled with the object names is stored in the data structure of the attribute. The last type of attribute is that which is evaluated by the inference system using the classification tree (the probabilistic semantic net) associated with the attribute. A unique dedicated classification tree is associated with each such attribute. This tree represents a sequential classification process. When a property of an object (the value of an attribute) of that type is requested, the property is estimated by classifying the object into one of few small categories of objects of that class, using other (hopefully simpler to obtain) properties. For each of the small categories, the generic model has an estimate of the property distribution for objects in that category. That distribution estimate is then taken as the property of that object. The sequential classification tree can be easily edited to obtain finer categories and to update the estimates of the distribution of the property for objects in the category. This facilitates learning of the generic model. More detail on the classification tree data structure is given below.

B. RELATIONS

The data structure of a relation specifies the class of objects it operates on and the class of objects on which it ranges. As is the case with attributes, there are three classes of relations as defined by the way they are computed. The first is the preprogrammed search where the search is done by a purely programmed algorithm. The second kind of relation requires that the list of objects will be provided by the operator. The third kind of relations, which we call inference relations, is computed from other (simpler) relations by union, intersection, and filtering the output of the simpler relations and attributes. The third type of relations is not yet fully developed in the existing system.

The rest of this article describes the representation of the sequential classification process associated with computation of inferred attributes, followed by a description of the driver system, which computes values of attributes and relations.

IV. INFERRED ATTRIBUTES — THE CLASSIFICATION TREE

The essence of the inference process which computes the value of an inferred attribute applied to an object is classification of the object. The object is classified into categories such that the range of values of that attribute for objects in the small category is very limited and, hence, the property can be well estimated.

Example: Consider three-dimensional bodies in an environment where three-dimensional bodies may be labeled only oranges, bananas, or table tops. Then, without any test, the a priori probability distribution of the attribute "label" applied to an object of class "three-dimensional body" selected at random will be something like orange with probability 0.6, table with probability 0.3, and banana with probability 0.1.

If we break the class of "three-dimensional bodies" into two categories, the first category contains those objects which have some planar surfaces, and the second category contains those objects which are purely curved surface. Then, the first category will include almost exclusively objects whose label takes the value tables, while the second category will be almost exclusively

bananas or oranges. Testing the color and shape of objects in the second category will allow further subclassification of objects in that category into finer subcategories, some almost exclusively containing objects labeled bananas and the others almost exclusively containing objects labeled oranges.

The classification tree for an inferred attribute actually represents the classification process. The top node of the tree stands for the category of all objects of the class. With each node, there is an associated category of object and each son node stands for a (finer) subcategory of the category of objects in the parent mode.

Each node contains the following information (to be described in more detail below):

(1) Calling attribute. Each node is part of a unique classification tree dedicated to one attribute. A pointer to that attribute is contained in the node.

(2) Default value. This is the distribution of the property over all objects which belong to the category associated with the node. This probabilistic information can either put in manually or can be collected by going over (manually) analyzed examples and counting the distribution. If the inference algorithm reaches the node without sufficient resources to expand the node this estimate is returned as the answer. If there is not a default estimate (insufficient training set or too wide range of values to be stored practically), a marker to that effect is put in.

(3) Node expansion plan (optional). This information designates either how to get a finer classification of the objects in the node's category so as to get a better estimate of the property or how to get a better estimate of the property directly without finer classification. Finer classification is obtained by selecting sons of the node which stand for subcategories of the category of objects. If there is no such plan associated with the node, the default estimate is the only possible answer.

(4) Integration procedure. This specifies how to integrate estimates returned from the activated sons to get a unique answer to be returned from the current node. This integration type is needed where, because of ambiguity, more than one son is activated (that is, the object may have been estimated to belong with positive probability to more than one subcategory).

(5) Text (optional). Describes the rate of the node in the classification tree. This text is put in by the person or procedure who generated the node.

The node expansion procedure (ANODE-EXPAND) is described in detail later in the paper. However, it will be outlined here to justify the data structure of the son selection plan. The node expansion procedure operates on one node at a time. When a node expansion is completed, the expanding procedure returns an estimated property. This estimate will be the default value associated with the node in the case where the node does not have a node expansion plan or there are not sufficient resources to activate the node expansion plan. When the node expansion plan is activated the estimate returned from the node will be either the weighted average of the estimates returned from recursive application of the node expansion procedure on sons of the node or the value Q (to be defined shortly) if the node expansion plan does not call for son expansion.

The son nodes where they exist correspond to subcategories of the objects in the category associated with the node. This finer classification is made based on other properties of the specific structure analyzed. Those other properties are requested in the node expansion plan stored in the parent node. Since these other properties may be nondeterministic, this selection of the subcategories may be nondeterministic. In such a case, the object will be assumed to belong to different sons (subcategories) with different positive probabilities. The property estimate of the answer returned from the parent node is the average of the estimates obtained from the activated son weighted by the probability that the object belongs to each of the corresponding subcategories. The integration procedure specifies which kinds of averaging are used for the node.

A node expansion plan contains the following information:

(1) An array of R activation records of relations which are used by the procedure RRECORD-EXPAND (see Sect. VI-E) (R is an integer associated with the node). The ith relation activating record ($1 \leq i \leq R$) specifies: (a) tne name of the relation REL [i], (b) the associated index SUBJECT [i] ($0 \leq$ SUBJECT [i] $< i$) which specifies the objects upon which REL [i] operates (SUBJECT [i] = 0 means the original object and OBJECT [i] = j, $1 \leq j < i$ means that REL [i] should be applied to the objects satisfying the jth relation activation record), (c) the portion of the resources which will be allocated to compute the ith relation activating record out of the total balance of the resources available for the node expansion, (d) the son which should be expanded if no object appears to satisfy the relation with probability greater than a specific threshold, (e) RMODE [i] which specify the mode of operation of RRECORD-EXPAND when it will expand that record.

(2) An array of N attributes activating records (N an integer associated with the node). This record is used by ARECORD-EXPAND (see sect. VI-D). The ith attribute activating record specifies: (a) ATT [i] the attribute it activates, (b) an index IND [i] to the objects operated on (IND [i] = 0) corresponds to the original object, $1 \leq$ IND(i) $\leq$ R means ATT [i] should be applied to the objects computed to satisfy IND [i]-th relation), (c) the portion of the balance of the available resources that should be allocated to that evaluation, and (d) AMODE [i] the mode of application of ARECORD-EXPAND the objects satisfying the IND [i]-th relation.

(3) Two arrays: A, which is a N by N real numbers array and B, which is a N real number array). When the node is expanded, each of the N attribute records is activated and it returns a property value. These N values are stored in N element array of values P. If N > 1 (e.g., more than one call on ARECORD-EXPAND is made) P(i) is the mean of the estimated values which ARECORD-EXPAND returned when N = 1 (only one call on ARECORD-EXPAND in that node), P(i) will be the property itself. The A and B are used to reduce the N properties into a single value Q.

$$Q = \vec{p}^{T} \cdot A \cdot \vec{P} + (\vec{B} \cdot \vec{P})$$

(4,a) — Indicator whether Q is the value to be returned by ANODE_EXPAND.

(4,b) The alternative to (4,a) is to have an array of M sons with an associated array of threshold T_i $1 \leq i \leq m$. T_i are such that $T_i < T_{i+1}$ and $T_m = +\infty$. The ith son will be selected to be expanded if Q has positive probability of satisfying

$$T_{i-1} < Q \leq T_i$$

This will mean that there is a positive probability that the object at hand belongs to the ith subcategory which is associated with the ith son. That probability will be the weight given to that son.

Now since Q can take a range of values with different probabilities, it is nondeterministic As a result, the object may have positive probability of being in more than one subcategory, which will require integrating the estimates returned from each son node so as to get a unique answer from the parent node. Typically, we use the arithmetic average (weighted by the probabilities that the object belongs to the son) of the answers coming back from the different activated sons. This implies real nondeterministic son selection. Alternatively, weighted geometrical averaging is used which implies that we treat each son's answer as an independent estimate.

V. INFERRED RELATION - COMBINATION SEARCHES

Combination searches are not implemented in the current system; we anticipate organizing them in a search tree. The nodes of a search tree will contain a node expansion plan which will be used for resource allocation and expansion of the node. The son selection plan data structure organization will be similar to the son selection plan used in the classification tree. However, clearly there will be no default answer associated with the node (the generic model cannot know the specific objects that will satisfy the relation when applied on a specific object). Also, the dominant form of integration of answers from sons will be probably to take the union of the objects lists returned from the different activated sons.

The default answer of the attribute inference node will most likely be replaced in the nodes of the search tree by calls on other (hopefully simpler) relations and filters.

VI. THE CONTROL STRUCTURE OF THE ACCESS OF THE DATA BASE

Typically an initial access to the data base is a request for the value of a relation or an attribute applied to an object. This request comes from an external user, either a human or another subsystem of the robot system. The external user specifies how much compute time he wishes to allocate to that request. This compute time will be allocated for retrieval of the information. This request is immediately translated into the appropriate calls on ATTR_EXPAND and REL_EXPAND (sections A and B). From that point on control and execution progress in the normal way (see below). The initial access may find an empty specific model (all class sets are empty). In this case the object in the initial request of the access is put into the proper objects class in the data base and other objects are found in the natural way by expanding relations. Description of the 6 recursive procedures which compose the data base access and inference process is given immediately below.

A. THE ATTRIBUTE EXPANDING ROUTINE

The first procedure of the data base control is:

ATTR_EXPAND(ATTRIBUTE,OBJECT,RES)

It expands a request for the property which is the value of the ATTRIBUTE (a pointer to an attribute in the generic model) applied to OBJECT (a pointer to an object in the specific model). RES specifies the total resources (machine time) allocated for that task. A pointer to the computed property is returned as the value of the procedure and also the association ATTRIBUTE (X) OBJECT=PROPERTY is inserted to the data base. The three major steps of the procedure are:

(1) Checks the validity of the request. That is, verify that the ATTRIBUTE is in the generic model, the OBJECT is in the specific model and the OBJECT belongs to the class of objects upon which the attribute operates.

(2) Checks whether the data base already contains the association of that attribute with that object. If that is the case, a decision is made whether to use one of the estimates of the property in the data base or to compute a new estimate of that property. The decision is made based on the period of validity of the stored estimates, the time when the stored estimate was obtained, and the amount of resources allocated and used to obtain the existing estimates. The idea is that if one of the existing estimates is recent and was obtained with about the same or more resources as available now there is no point in doing redundant work. In that case, the best existing estimate in the data base is returned as the value of the procedure immediately.

(3) The action here depends on the type of the attribute. For a preprogrammed attributes, a call is made on the library routine which computes the property estimate on the PDP-10 or via the communication line on the SPC-16. The value returned by the library routine is then stored in the data base (in the proper association) and a pointer to it is returned as the value of the procedure. For interactive (console) attributes, a request for the value is issued to the user's terminal. For the inferred attributes, a call on

ANODE_EXPAND (ATTRIBUTE,NODE,OBJECT,ATTR,NRES)

is issued. (ANODE_EXPAND is the third procedure of the driver and is described below.) NODE is the top node of the classification tree of that attribute (which is specified in the attribute description in the generic world model). The OBJECT is the same object as the OBJECT in the original call and NRES is the balance of resources left after proceeding steps 1 and 2. The value returned by ANODE_EXPAND is taken as the property value and is stored in the data base in the proper association and a pointer to it is returned as the value of ATTR_EXPAND in that call.

B. THE RELATION EXPANDING ROUTINE

The second procedure of the inference system driver is

RELAT_EXPAND(RELATION,OBJECT,RES,MODE,THRESHOLD)

This procedure returns as a value a pointer to the object list (relation value). This object list is the list of the objects which appear to satisfy the RELATION (which is a pointer to a relation in the generic model) applied to OBJECT (a pointer to an object in the specific model). RES is the resources allocated for that search. MODE specifies the type of call, e.g., whether the procedure should return immediately as soon as an object which satisfies the relation with probability higher than THRESHOLD is found or whether it should attempt to find as many objects which satisfy the relation exhausting the resources allocated. The procedure operates in three steps corresponding to the three steps of ATTR_EXPAND.

(1) Checks the validity of the call, e.g., RELATION must be defined in the generic model, OBJECT is in the specific model, and belongs to the class of objects upon which the relation operates as specified in the relation data structure.

(2) Checks whether the information requested is already substantially stored in the data base, in which case it returns the stored information immediately.

(3) Depends on which kind of relation is being evaluated. If the relation is preprogrammed, a call on the computing library routine on the SPC-16 or the PDP-16 is made. Where the relation is interactive, a console message requesting the list of objects is issued to the operator. In the case of inferred relation when it will be implemented, a call on the top node of the relation, inference tree node will be issued RNODE_EXPAND(NODE,OBJ,RELATION,NRES,MODE,THRESHOLD) corresponding to the ANODE_EXPAND in the case of inferred properties. The value returned by RNODE_EXPAND will be taken as the value of RELAT_EXPAND. In either case the object list (the relation value) is also stored in the data base in the association RELATION (X) OBJECT= OBJECT LIST.

C. THE ROUTINE WHICH EXPANDS A NODE IN AN ATTRIBUTE INFERENCE TREE

The third recursive procedure of the driver is

ANODE_EXPAND ATTRIBUTE (NODE,OBJECT,PARAM,RES)

This procedure returns with an estimate of the property values distribution represented in the same form as it is represented in a property (except that validity time, etc. are not included). The expansion of the node is made in the following steps:

(1) Verifies the validity of the call. Check to see that the NODE actually is a valid one and this node is dedicated to the analysis of the specified ATTRIBUTE. This check is made only for debugging purposes.

(2) Checks whether the node should be expanded. If the node is not going to be expanded, the value returned by the routine will be the default answer stored in the node and no further processing will be made. Otherwise, the node expansion plan will be activated. Decision to expand the node is made if two conditions are satisfied. First, the node must have a node expansion plan and, secondly, the resources allocated to this call of the procedure must exceed a certain threshold which is considered minimum for expansion of that node.

(3) Computes the R relations in the relation array in the node (if $R \geq 1$). These relations are evaluated in the order of 1 to R by recursive calls on the RRECORD_EXPAND(REL[i],OBJAR[SUBJECT[I]],NRES,RMODE[I], THRESHOLD[I]). (RRECORD_EXPAND is the 5th data base driver of the inference system.) A pointer to the object list returned by the ith call on RRECORD_EXPAND whether empty or not is stored in OBJAR [i]. OBJAR is an R-element array of pointers to object lists associated with the present expansion of the node. The subject on which the ith relation operates can be any of the objects already pointed at in OBJAR, e.g., SUBJECT[i] must be less than i. Clearly, the first relation must be applied to the original object (e.g., OBJAR[i] = 0) since no other object is pointed to yet. Where there

is call on OBJAR[i] = 0 a direct call on REL_EXPAND with the appropriate variables is issued. If no object appears to satisfy the ith relation the node expansion will branch to $RSON_i$ which corresponds to the subcategory where no object satisfies the ith relation. NRES is computed to be the portion of the compute time left for the node expansion that the plan recommends applying for that call.

(4) Computes the N properties associated with the node. The property is computed either by call on ARECORD_EXPAND (the 4th driver routine) with the proper parameter, subject, type of call and resources; or, if the index is 0, by a call on ATTR_EXPAND directly to apply the current attribute to the original object. The values returned by the property are stored into a vector (array) of property values, P [1:n].

(5) Integrates the N values in P into a single property like quantity Q:

$$Q = P^T A P + B P$$

where A is a two-dimensional array N by N and B a one-dimensional array (vector) of N elements. If N=1, Q looks like a typical property estimate, that is

$$Q = \begin{Bmatrix} val_1 & P_1 \\ val_2 & P_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ val_k & P_k \end{Bmatrix}$$

$$P_1 + P_2 + \ldots + P_k = 1$$

and $Val_{i+1} > Val_i$. If N > 1 for computational complexity reasons the P[i]'s are made deterministic values which is the average of the values in the property, and hence Q is also made deterministic.

(6) If Q is marked as the value to be returned by the expansion plan that value is returned as the value of ANODE_EXPAND.

(7) In the case where the node expansion plan calls for son-nodes expansion the next step is to compare the range of values of Q against a list of threshold and sons

$$\begin{pmatrix} son_1 & son_2 & son_m \\ T_1 & T_2 & T_m \end{pmatrix}.$$

SON_i is expanded if the sum of the probability of Val_j falling into (T_i, T_{i+1}) is positive. The sum of probability is the portion of the available resources for the ith son expansions by recursive call on ANODE_EXPAND out of the remaining resources.

(8) The answer returned from the different expanded sons is integrated by averaging them and returning that value as the value of that call on PNODE_EXPAND. The weight of the answer returned from the node is the same as the weight used to allocate resources to the sons.

D. THE ROUTINE WHICH EXPANDS AN ATTRIBUTE ACTIVATING RECORD IN NODES

The 4th procedure of the driver system is:

ARECORD_EXPAND(ATTR,POBJL,MODE,RES)

POBJL is a pointer of an object list which will be referred to as OBJL. ARECORD_EXPAND starts as the other routine by checking the validity of the call for debugging purposes. The next action depends on the MODE in the call. When the system was implemented originally, there was only one mode of operation (Ref. 20), but unfortunately the complexity of the simplest application forced us to implement plurality of modes. In all cases, it returns a property-like value which is computed in the following ways:

Mode 1: The attribute ATTR is applied by a call on ATT_EXPAND to each object in OBJL with a portion of the resources RES proportional to the probability that the object belongs to OBJL. The values returned by those calls are added together with proper weights to generate a single property-like value returned by ARECORD_EXPAND.

Mode 2: ATTR is applied to all objects in OBJL but the value returned is the property of the object whose mean is the maximum among all other objects.

Mode 3: Same as mode 2 except that the minimum is returned.

Mode 4: The mean of all values computed by application as in mode 1 is returned, e.g., a single deterministic value which is the mean (average) of the property value which would have been returned by similar call with Mode 1.

Mode 5: The variance of the values computed by application as in mode 1 is returned.

Mode 6: ATTR is applied only to the one object which maximizes the probability that it belongs to OBJL among all other objects in the list.

E. THE ROUTINE WHICH EXPANDS RELATION RECORD IN NODES

The 5th procedure of the driver

RRECORD_EXPAND(RELATION,POBJL,RES,MODE,THRESH)

POBJL is a pointer to an object list referred to as OBJL. RRECORD_EXPAND will return a pointer to an object list that it generated.

MODE = 0 means REL_EXPAND is going to be called only once to apply the RELATION to the object which maximizes the belong feature among all objects in the list pointed to by POBJL.

MODE = 1 means REL_EXPAND is going to be applied on all objects in the list. The new object list is generated by taking the object which satisfies each call of REL_EXPAND with highest probability.

MODE = 2 means REL_EXPAND is going to be applied on all objects in the list. All the object's lists generated are going to be united into a single long list.

F. THE ROUTINE WHICH EXPANDS NODES IN RELATION INFERENCE TREE

The sixth routine of the driver is RNODE_EXPAND. This routine when implemented will be similar to PNODE_EXPAND with two differences:

(1) If the search operates in the existential mode, then as soon as one object which appears to satisfy the relation with high enough probability is found, the procedure returns immediately.

(2) There are no default answers. Instead of the default answer, a node may have a call on other relations and a filter property. If there is such a relation in the node, it is always evaluated with a portion of the resources. Then the filter is applied to those objects which satisfy that relation. The filter can have only two values, 0 and 1. The weighted average of the two values is taken as the probability that the object satisfies the original relation. If that probability is high enough, the object is added to the object list.

VII. PERMANENT STORAGE

The associative data base representing a specific structure is meaningless without the associated generic model which defines the classes of objects, the attributes, and the relations. Hence, there are two types of permanent storage options on a magnetic disk or tape file. One is the generic model by itself, and the other is a generic model coupled with a description of a specific structure using the terms defined by the generic model. The system can store files containing either kind of data and accept them so as to continue analysis or editing from the status as saved.

VIII. MEMORY REFRESH

We plan on implementing a memory garbage collection mechanism. This mechanism will be called when the data base runs out of storage. It will erase properties values and object lists giving priority according to the following factors:

(1) The amount of resources used to obtain the estimate.

(2) The length of time left for that information to be valid.

(3) The time lapsed after the last access for that information.

IX. CONCLUDING REMARKS

The foregoing material describes our approach to the representation of perceptual information. This system is an evolving effort, and is expanded as experience is gained in using the inference capabilities of the data base. Control options are added when necessary or when they appear to help convenience of use. We believe this work is an improvement on our previous efforts described in Refs. 4, 17, 20. We consider our major contribution to be the integration of the concepts of (1) inference and search bound by time constraints, (2) nondeterministic hierarchical inferences, and (3) nondeterministic data representation.

Our present goal is to apply the system to image recognition using the low-level segmentation routines described in Refs. 13 and 14 so as to create a more effective system than the one described in Ref. 17. We are also looking for practical ways to save the status of inference processes which exhausted their resources so that if additional resources become later available, the inference process may be resumed from the status it was in when it was terminated. Currently, it has to be started from the beginning again, but it can make use of any properties and values of relations that are already in the data base.

We also consider the theoretical merit of integrating the resources (compute time) available more elaborately into the node expansion plan. Adding such an option may allow more effective and sophisticated use of limited resources in special cases. That is, branching to special options where there are very limited resources.

APPENDIX A

EXAMPLE OF A GENERIC MODEL

I. INTRODUCTION

This section describes a generic model which DABI uses to solve a traversability problem for the JPL Robot Vehicle. The model is used to obtain an estimate of whether or not the vehicle can safely occupy a specified location in the environment. A complete listing of the model is included at the end of this appendix (Table A-1).

For a location to be classified "safe," several criteria must be satisfied. The weight of the vehicle should be nearly equally distributed on all four wheels. This implies that the four points which support the wheels should be "nearly" planar. The vehicle should be supported at a slope which is acceptable within limits imposed by wheel traction, driving and braking power, and the center of gravity. Once it has been determined that the vehicle will be adequately supported, the area that would be covered by the vehicle in that location must be examined for obstacles. The area around each wheel support point is checked first to see that the wheels are free of obstacles, then the entire region is checked to see if there are any obstacles which would interfere with any other part of the vehicle.

As the classification tree is expanded, each of the above criteria is tested as the value of an attribute defined in the model, in the order presented above. If each test is satisfied with probability 1, the location is classified "safe." If any test fails with probability 1, the location is classified "unsafe." Otherwise a value (safe, p_1), (unsafe, p_2) is assigned to the location where $p_1 + p_2 = 1$.

II. DESCRIPTION OF THE MODEL

The generic model consists of four data structure type CLASSES, RELATIONS, ATTRIBUTES, and NODES. A CLASS stands for a set of objects of the same type. RELATIONS and ATTRIBUTES are mappings f: A → B. In either case the domain A is a class defined in the generic model. In the case of RELATIONS, B is also a class defined in the model. For ATTRIBUTES, B is a specified set of integers.

Each NODE of the classification tree is described in terms of the relations and attributes to be expanded by that node, and the manner in which the expansion proceeds from that node.

A 3-dimensional cartesian coordinate system fixed in the environment is defined such that the x- and y-axes determine a horizontal plane and the positive z-axis points up. All references to the coordinates of a point are with this coordinate system in mind. The unit of measurement is inches.

III. CLASSES

A. VEHICLE!LOCATION

An object in this class specifies a vehicle location and is represented by an ordered triple (x_0, y_0, θ_0). The vehicle is at location (x_0, y_0, θ_0) if:

(1) The coordinates of a fixed point P rigidly attached to the vehicle are (x_0, y_0, z);

(2) θ_0 is the angle formed by the positive x-axis and a vector $\vec{V}$ pointing toward the front of the vehicle (see Fig. A-1).

Implicit in this definition is a rectangular region which contains the projection of each point of the vehicle onto the xy-plane. This region will be referred to as the vehicle region.

B. TILE

The vehicle region corresponding to a vehicle location (x_0, y_0, θ_0) is subdivided into 8 by 8 in. squares or tiles. Each of these subregions is an object of class TILE and is specified by the (x,y) coordinates of its center point. The four tiles which contain the support points of the wheels will be referred to as the wheel tiles.

C. SAMPLE!POINT

A real world surface point (x, y, z) is an object of SAMPLE!POINT. The coordinates of a SAMPLE!POINT are obtained by the robot sensory instruments, using either the LRF (laser range finder) or stereo correlation (Ref. 15).

IV. RELATIONS

SUPPORT: VEHICLE!LOCATION → TILE. This relation is preprogrammed to return a list of the four wheel tiles corresponding to a vehicle location (x_0, y_0, θ_0). The list is generated in the following order: right front (wheel), right rear, left rear, and left front.

A. WHEEL!AREA: TILE → TILE

This relation is preprogrammed. This relation is applied to each wheel tile and returns a list of nine tiles forming a square with the wheel tile in the center (see Fig. A-2).

B. GET!REGION: VEHICLE!LOCATION → TILE

This relation is preprogrammed to return a list of all the tiles covering the vehicle region defined by (x_0, y_0, θ_0).

C. SAMPLE: TILE → SAMPLE!POINT

This relation is preprogrammed to drive the sensory instrument to find a real world surface point (x, y, z) whose (x, y) coordinates are in the map tile.

V. ATTRIBUTES

A. SAFETY: VEHICLE!LOCATION → {1,2}

The value of this inferred attribute is taken as the estimate of whether the location (x_0, y_0, θ_0) is safe or unsafe for the vehicle. The values that this attribute can take are 1 and 2, corresponding to safe and unsafe, respectively. Nodes 1, 3, 4, 6, 7, 9, 10, 11, and 12 are controlled by SAFETY. The values computed at each of these nodes are integrated into a single value which node 1 returns as the value of SAFETY.

B. ZCOORD: SAMPLE!POINT → {integers}

This attribute is preprogrammed to return the z-coordinate of a sample point.

C. PLANE!FIT: VEHICLE!LOCATION → {non-negative integers}

This attribute is computed in node 2 using ARECORD_EXPAND. The attribute ZCOORD is applied to the sample point of each wheel tile in the order that they are generated by the relation SUPPORT. If the wheel tiles are labeled in order T_1, T_2, T_3, T_4 with Z_i $(1 \leq i \leq 4)$ the corresponding z-coordinate of T_i then the value of PLANE!FIT can be expressed as

$$d = \left| (z_1 + z_3) - (z_2 + z_4) \right| \quad .$$

If the four points are planar, the value of d is 0. Otherwise d represents the distance in the z direction from any one point to a plane passing through the other three points. Thus d can be thresholded to determine if all four wheels of the vehicle will be adequately supported at the location (x_0, y_0, θ_0).

D. SUPPORT!VAR: VEHICLE!LOCATION → {non-negative integers}

The value of this attribute is computed in node 5 in a manner similar to PLANE!FIT. The value of SUPPORT!VAR is $(\max \{z_i\} - \min \{z_i\})$, where z_i, $1 \leq i \leq 4$, represents the z-coordinates of the sample points of the four wheel tiles as above. This value is a measure of the slope of the vehicle region.

E. SUPPORT!HGT: VEHICLE!LOCATION → {integers}

This attribute is computed in node 14 as the average of the z_i, $1 \leq i \leq 4$ (z_i as above).

F. WHEEL!OBST: TILE → {non-negative integers}

This attribute is computed in node 8. For each wheel tile (obtained by SUPPORT), the relation WHEEL!AREA is expanded. ZCOORD is called for the sample point of each tile ARECORD_EXPAND. The value $w_i = \max \{z\} - \min \{z\}$, $i = 1 \leq i \leq 4$ is computed for each wheel area. The value of WHEEL!OBST is taken as the worst case of w_i over the four wheel areas i.e.,

$$\max_{1 \leq i \leq 4} \left(|w_i| \right)$$

A large value of WHEEL!OBST indicates the presence of an obstacle near one of the wheels.

G. HIGH!POINT: VEHICLE!LOCATION → {non-negative integers}

Using the list of all tiles obtained by the relation GET!REGION (in node 13), this attribute finds the maximum z-coordinate taken over the sample points of all tiles in the vehicle region. The z-coordinates are obtained by successive calls to ZCOORD in ARECORD_EXPAND. The difference between HIGH!POINT and SUPPORT!HGT is taken as an indication of whether or not there are obstacles present in the vehicle region.

H. NODES

The tree structure (Fig. A-3) is designed to compute the attributes described above in the order of increasing amounts of data necessary to compute the desired value.

In node 1, PLANE!FIT is computed. This value is used to determine the selection of nodes 3 and 4. For large values, node 4 is expanded, which returns a default answer UNSAFE.

In node 3 (expanded for small values of PLANE!FIT) the value of SUPPORT!VAR is obtained and used to determine the selection of nodes 6 and 7. For large values, node 7 is expanded which returns a default answer UNSAFE.

In node 6 (expanded for small values of SUPPORT!VAR) the value of WHEEL!OBST is computed, and used to determine the selection of nodes 9 and 10. For large values, node 10 is expanded which returns a default answer UNSAFE.

In node 9 (expanded for small values of WHEEL!OBST) the value of HIGH!POINT and SUPPORT!HGT are computed. The difference between these two values is used to determine the selection of nodes 11 and 12. If difference is large (indicates a vehicle obstacle is present), node 12 is expanded which returns a default answer UNSAFE. If the difference is small, node 11 is expanded and returns a default answer SAFE.

This completes the expansion of the nodes controlled by SAFETY. The values at each node are integrated and returned by node 1 in the form (SAFE, p_1), (UNSAFE, p_2) as the value of SAFETY ($p_1 + p_2 = 1$).

Nodes 2, 5, 8, 13, and 14 are controlled by the attributes PLANE!FIT, SUPPORT!VAR, WHEEL!OBST, HIGH!POINT, and SUPPORT!HGT respectively. Each of these nodes returns the value

$$Q = P \times A \times P^T + P \times B$$

as described in section VI-C (6).

Table A-1. A generic model for vehicle obstacle detection

```
CLASSES:
    SAMPLE!POINT
    TILE
    VEHICLE!LOCATION

RELATIONS:
    NAME:       SUPPORT
    TYPE:       PDP10
    DOMAIN:     VEHICLE!LOCATION
    RANGE:      TILE

    NAME:       WHEEL!AREA
    TYPE:       PDP10
    DOMAIN:     TILE
    RANGE:      TILE

    NAME:       SAMPLE
    TYPE:       CPC16
    DOMAIN:     TILE
    RANGE:      SAMPLE!POINT

    NAME:       GET!REGION
    TYPE:       PDP10
    DOMAIN:     VEHICLE!LOCATION
    RANGE:      TILE

ATTRIBUTES:
    NAME:                   HIGH!POINT
    TYPE:                   TREE
    DOMAIN:                 VEHICLE!LOCATION
    RANGE:                  UNBOUNDED
    DEFAULT VALUE:          NONE

    NAME:                   SUPPORT!HGT
    TYPE:                   TREE
    DOMAIN:                 VEHICLE!LOCATION
    RANGE:                  UNBOUNDED
    DEFAULT VALUE:          NONE

    NAME:                   PLANE!FIT
    TYPE:                   TREE
    DOMAIN:                 VEHICLE!LOCATION
    RANGE:                  UNBOUNDED
    DEFAULT VALUE:          NONE

    NAME:                   ZCOORD
    TYPE:                   PDP10
    DOMAIN:                 SAMPLE!POINT
    RANGE:                  UNBOUNDED
    DEFAULT VALUE:          NONE

    NAME:                   SAFETY
    TYPE:                   TREE
    DOMAIN:                 VEHICLE!LOCATION
    RANGE:                  2
    DEFAULT VALUE:
        SAFE:1
        UNSAFE:1

    NAME:                   WHEEL!OBST
    TYPE:                   TREE
    DOMAIN:                 TILE
    RANGE:                  UNBOUNDED
    DEFAULT VALUE:          NONE

    NAME:                   SUPPORT!VAR
    TYPE:                   TREE
    DOMAIN:                 VEHICLE!LOCATION
    RANGE:                  UNBOUNDED
    DEFAULT VALUE:          NONE

NODES:

1   TEXT:                   SAFETY TOP NODE
    TYPE:                   EXPAND SONS
    ATTRIBUTE:              SAFETY
    DEFAULT VALUE:
        SAFE:1
        UNSAFE:1
    RELATION BLOCK:         EMPTY
    SELECTION BLOCK:

        VECTOR #:           1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:      PLANE!FIT
            OBJECT INDEX:   0
            WEIGHT:         1
            MODE:           0

    SON BLOCK:
        SON:                3     4
        THRESHOLD:          6     INF
    INTEGRATION TYPE:       2

2   TEXT:                   COMPUTE PLANE!FIT
    TYPE:                   TERMINAL--RETURN 0
    ATTRIBUTE:              PLANE!FIT
    DEFAULT VALUE:
        1:1
    RELATION BLOCK:

        RELATION:           SUPPORT
        OBJECT INDEX:       0
        WEIGHT:             1
        THRESHOLD:          0.5
        SON:                2
        MODE:               0

        RELATION:           SAMPLE
        OBJECT INDEX:       1
        WEIGHT:             1
        THRESHOLD:          0.5
        SON:                2
        MODE:               2

    SELECTION BLOCK:

        VECTOR #:           1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:      ZCOORD
            OBJECT INDEX:   2
            WEIGHT:         1
            MODE:           2

    SON BLOCK:              EMPTY
    INTEGRATION TYPE:       2

3   TEXT:                   ALL 4 WHEELS ON THE GROUND
    TYPE:                   EXPAND SONS
    ATTRIBUTE:              SAFETY
    DEFAULT VALUE:
        SAFE:1
        UNSAFE:1
    RELATION BLOCK:         EMPTY
    SELECTION BLOCK:

        VECTOR #:           1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:      SUPPORT!VAR
            OBJECT INDEX:   0
            WEIGHT:         1
            MODE:           0

    SON BLOCK:
        SON:                6     7
        THRESHOLD:          6     INF
    INTEGRATION TYPE:       2

4   TEXT:                   ONE WHEEL OFF THE GROUND
    TYPE:                   TERMINAL--RETURN DEFAULT ANSWER
    ATTRIBUTE:              SAFETY
    DEFAULT VALUE:
        UNSAFE:1
    RELATION BLOCK:         EMPTY
    SELECTION BLOCK:        EMPTY
    SON BLOCK:              EMPTY
    INTEGRATION TYPE:       2

5   TEXT:                   COMPUTE SUPPORT!VAR
    TYPE:                   TERMINAL--RETURN 0
    ATTRIBUTE:              SUPPORT!VAR
    DEFAULT VALUE:
        1:1
    RELATION BLOCK:

        RELATION:           SUPPORT
        OBJECT INDEX:       0
        WEIGHT:             1
        THRESHOLD:          0.5
        SON:                5
        MODE:               0

        RELATION:           SAMPLE
        OBJECT INDEX:       1
        WEIGHT:             1
        THRESHOLD:          0.5
        SON:                5
        MODE:               2
```

Table A-1 (contd)

```
    SELECTION BLOCK:

        VECTOR B:          1  -1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:       ZCOORD
            OBJECT INDEX:    2
            WEIGHT:          1
            MODE:            2

            ATTRIBUTE:       ZCOORD
            OBJECT INDEX:    2
            WEIGHT:          1
            MODE:            4

    SON BLOCK:           EMPTY
    INTEGRATION TYPE:    2

6   TEXT:                LOOK FOR WHEEL!OBSTACLE
    TYPE:                EXPAND SONS
    ATTRIBUTE:           SAFETY
    DEFAULT VALUE:
        SAFE=1
        UNSAFE=1
    RELATION BLOCK:

        RELATION:        SUPPORT
        OBJECT INDEX:    0
        WEIGHT:          1
        THRESHOLD:       0.5
        SON:             9
        MODE:            0

    SELECTION BLOCK:

        VECTOR B:          1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:       WHEEL!OBST
            OBJECT INDEX:    1
            WEIGHT:          1
            MODE:            3

    SON BLOCK:
        SON:               9    10
        THRESHOLD:         1    INF
    INTEGRATION TYPE:    2

7   TEXT:                DANGEROUS SLOPE
    TYPE:                TERMINAL--RETURN DEFAULT ANSWER
    ATTRIBUTE:           SAFETY
    DEFAULT VALUE:
        UNSAFE=1
    RELATION BLOCK:      EMPTY
    SELECTION BLOCK:     EMPTY
    SON BLOCK:           EMPTY
    INTEGRATION TYPE:    2

8   TEXT:                WHEEL!OBST COMPUTE MODE
    TYPE:                TERMINAL--RETURN 0
    ATTRIBUTE:           WHEEL!OBST
    DEFAULT VALUE:
        1=1
    RELATION BLOCK:

        RELATION:        WHEEL!AREA
        OBJECT INDEX:    0
        WEIGHT:          1
        THRESHOLD:       0.5
        SON:             8
        MODE:            0

        RELATION:        SAMPLE
        OBJECT INDEX:    1
        WEIGHT:          1
        THRESHOLD:       0.5
        SON:             8
        MODE:            2

    SELECTION BLOCK:

        VECTOR B:          1  -1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:       ZCOORD
            OBJECT INDEX:    2
            WEIGHT:          1
            MODE:            3

            ATTRIBUTE:       ZCOORD
            OBJECT INDEX:    2
            WEIGHT:          1
            MODE:            4

    SON BLOCK:           EMPTY
    INTEGRATION TYPE:    2

9   TEXT:                SAFE SO FAR . . . LOOK FOR VEHICLE OBSTACLE
    TYPE:                EXPAND SONS
    ATTRIBUTE:           SAFETY
    DEFAULT VALUE:
        SAFE=1
        UNSAFE=1
    RELATION BLOCK:      EMPTY
    SELECTION BLOCK:

        VECTOR B:          1  -1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:       HIGH!POINT
            OBJECT INDEX:    0
            WEIGHT:          2
            MODE:            0

            ATTRIBUTE:       SUPPORT!HGT
            OBJECT INDEX:    0
            WEIGHT:          1
            MODE:            0

    SON BLOCK:
        SON:               11    12
        THRESHOLD:          0   INF
    INTEGRATION TYPE:    2

10  TEXT:                FOUND A WHEEL OBSTACLE
    TYPE:                TERMINAL--RETURN DEFAULT ANSWER
    ATTRIBUTE:           SAFETY
    DEFAULT VALUE:
        UNSAFE=1
    RELATION BLOCK:      EMPTY
    SELECTION BLOCK:     EMPTY
    SON BLOCK:           EMPTY
    INTEGRATION TYPE:    2

11  TEXT:                NO OBSTACLES
    TYPE:                TERMINAL--RETURN DEFAULT ANSWER
    ATTRIBUTE:           SAFETY
    DEFAULT VALUE:
        SAFE=1
    RELATION BLOCK:      EMPTY
    SELECTION BLOCK:     EMPTY
    SON BLOCK:           EMPTY
    INTEGRATION TYPE:    2

12  TEXT:                THERE IS A VEHICLE OBSTACLE
    TYPE:                TERMINAL--RETURN DEFAULT ANSWER
    ATTRIBUTE:           SAFETY
    DEFAULT VALUE:
        UNSAFE=1
    RELATION BLOCK:      EMPTY
    SELECTION BLOCK:     EMPTY
    SON BLOCK:           EMPTY
    INTEGRATION TYPE:    2

13  TEXT:                COMPUTE HIGHEST POINT IN REGION
    TYPE:                TERMINAL--RETURN 0
    ATTRIBUTE:           HIGH!POINT
    DEFAULT VALUE:
        1=1
    RELATION BLOCK:

        RELATION:        GET!REGION
        OBJECT INDEX:    0
        WEIGHT:          1
        THRESHOLD:       0.5
        SON:             13
        MODE:            0

        RELATION:        SAMPLE
        OBJECT INDEX:    1
        WEIGHT:          1
        THRESHOLD:       0.5
        SON:             13
        MODE:            2

    SELECTION BLOCK:

        VECTOR B:          1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:       ZCOORD
            OBJECT INDEX:    2
            WEIGHT:          1
            MODE:            3

    SON BLOCK:           EMPTY
    INTEGRATION TYPE:    2
```

Table A-1 (contd)

```
14  TEXT:               COMPUTE SUPPORTINGT
    TYPE:               TERMINAL--RETURN 0
    ATTRIBUTE:          SUPPORTINGT
    DEFAULT VALUE:
        1.1
    RELATION BLOCK:

        RELATION:           SUPPORT
        OBJECT INDEX:       0
        WEIGHT:             1
        THRESHOLD:          0.5
        SON:                14
        MODE:               0

        RELATION:           SAMPLE
        OBJECT INDEX:       1
        WEIGHT:             1
        THRESHOLD:          0.5
        SON:                14
        MODE:               2

    SELECTION BLOCK:

        VECTOR #:           1

        SELECTION ATTRIBUTES:
            ATTRIBUTE:      ZCCORD
            OBJECT INDEX:   2
            WEIGHT:         1
            MODE:           1

    SON BLOCK:          EMPTY
    INTEGRATION TYPE:   2
```

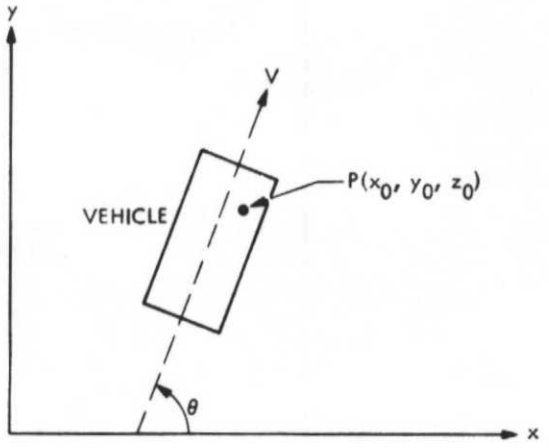


Fig. A-1. A vehicle location

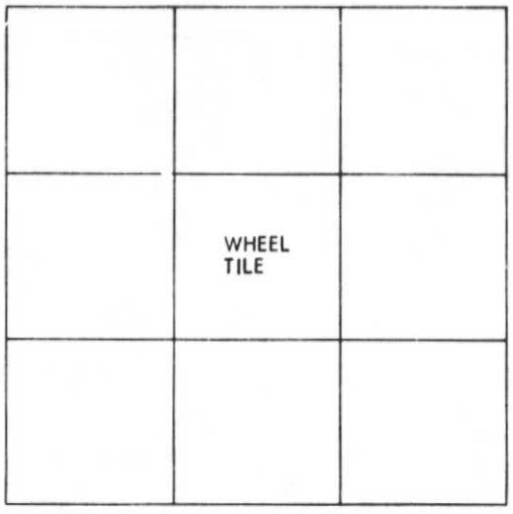


Fig. A-2. Configuration of tiles computed by WHEEL!AREA

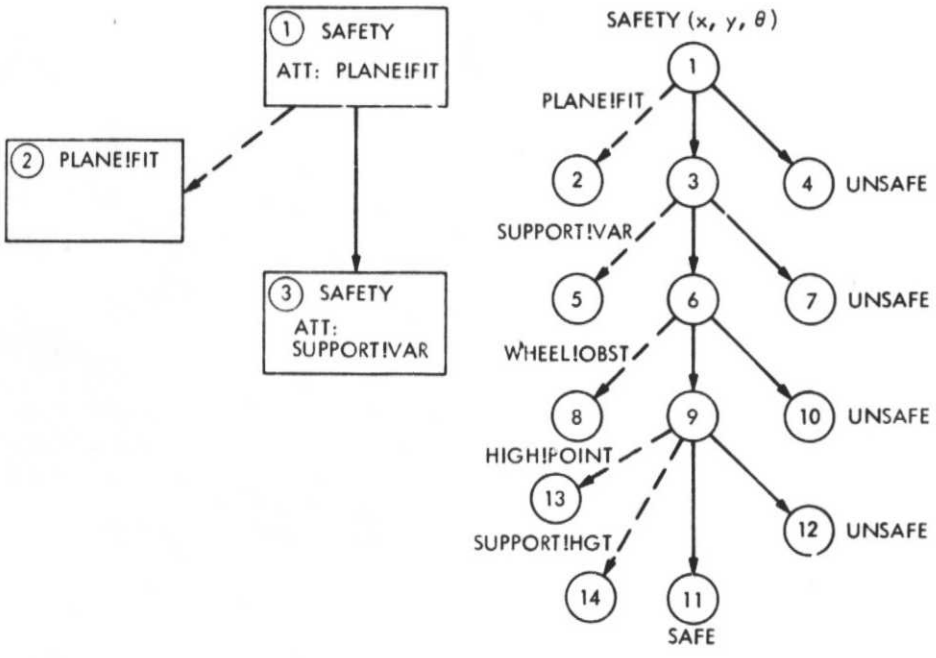


Fig. A-3. Safety inference tree

APPENDIX B

AN EXAMPLE OF A SPECIFIC MODEL

This section presents an application of the generic model described in Appendix 1. Table B-1 lists the contents of the associative data base after the expansion of SAFETY for the location $(x,y,\theta) = (0,0,0)$ is complete. The listing in Table B-1 consists of three parts: OBJECTS, RELATION VALUES, and ATTRIBUTE VALUES. Objects are listed under the class they belong to. The number in parentheses after each class name indicates the number of objects present in that class. Objects names consists of a letter to help identify the class of the object, and the numbers that specify that object to the system. For example, S-5!49!30 (line 300) is the SAMPLE!POINT with (x,y,z) coordinates $(-5,49,30)$; T-5!49 (line 800) is the TILE centered at $(x,y) = (-5,49)$; V0!0!0 (line 1300) is the VEHICLE!LOCATION $(x,y,\theta) = (0,0,0)$. V0!0!0 is defined in the initial request to expand SAFETY of V0!0!0. The objects in classes TILE and SAMPLE!POINT are defined as the result of relation values (SUPPORT and SAMPLE) obtained during the expansion.

Relation values are listed starting at line 1500. Each relation defined in the generic model is listed with a number in parentheses indicating how many objects (subjects) the relation was applied to in this expansion. Each subject of the relation is followed by three quantities in parentheses indicating the resources allocated and the resources used for the expansion on that subject, and the time the value was obtained. For example (line 1700) 672 milliseconds were allocated to obtain the value of SUPPORT (V0!0!0). The expansion required 50 milliseconds and the value was obtained at 90:9:28:46 (day:hour:minute:second, with the days counted from Jan. 1, 1976). Under each subject, the objects satisfying the relation are listed with the corresponding probabilities. Thus the value of SUPPORT (V0!0!0) is a list of four (wheel) TILES - T-5!49, T-101!49, T-101!-7, T-5!-7, each satisfying the relation with probability 1. The relation SAMPLE (line 2300) was applied to four TILES and one SAMPLE!POINT was obtained for each TILE.

The listing of attribute values follows the same format as the relation values with a list of (value, count) pairs under each object instead of (object, probability pairs). In the case of SAFETY, names have been defined for the range values (1=SAFE, 2=UNSAFE) and the value name appears (line 5100).

The expansion in this example involves nodes 1, 3, and 7 (SAFETY); 2 (PLANE!FIT); and 5 (SUPPORT!VAR). The relation values for SUPPORT and SAMPLE are originally obtained in node 2, along with the value of ZCOORD for each SAMPLE!POINT. Node 2 returns $|(30\pm10) - (30+10)| = 0$ as the value of PLANE!FIT (see Appendix A). This value is thresholded in Node 1 and the expansion proceeds to Node 3 which is expanded for small values of PLANE!FIT.

The values of SUPPORT, SAMPLE and ZCOORD are required again in Node 5 to compute SUPPORT!VAR. The existing values are retrieved from the associative data base and the value 30-10 = 20 is returned to node 3 as the value of SUPPORT!VAR (see Appendix A). This value is thresholded and the expansion proceeds to node 7 where it is assumed that the support slope of the vehicle is greater than the allowable slope. Node 7 returns the value (UNSAFE, 1), which is ultimately returned by node 1 as the value of SAFETY (V0!0!0).

Table B-1. A specific model based on the generic model of A-1

```
00100   OBJECTS
00200       SAMPLE!POINT(4)
00300           S-5!49!30
00400           S-101!49!10
00500           S-101!-7!10
00600           S-5!-7!30
00700       TILE(4)
00800           T-5!49
00900           T-101!49
01000           T-101!-7
01100           T-5!-7
01200       VEHICLE!LOCATION(1)
01300           V0!0!0
01400
01500   RELATION VALUES
01600       SUPPORT(1)
01700           V0!0!0(672,50,90:9:28:46)
01800               T-5!49,1.0,
01900               T-101!49,1.0,
02000               T-101!-7,1.0,
02100               T-5!-7,1.0;
02200       WHEEL!AREA(0)
02300       SAMPLE(4)
02400           T-5!49(151,33,90:9:28:48)
02500               S-5!49!30,1.0;
02600           T-101!49(151,33,90:9:28:50)
02700               S-101!49!10,1.0;
02800           T-101!-7(151,50,90:9:28:51)
02900               S-101!-7!10,1.0;
03000           T-5!-7(151,33,90:9:28:53)
03100               S-5!-7!30,1.0;
03200       GET!REGION(0)
03300
03400   ATTRIBUTE VALUES
03500       HIGH!POINT(0)
03600       SUPPORT!HGT(0)
03700       PLANE!FIT(1)
03800           V0!0!0(4500,317,90:9:28:53)
03900               0,1;
04000       ZCOORD(4)
04100           S-5!49!30(380,16,90:9:28:53)
04200               30,1;
04300           S-101!49!10(380,0,90:9:28:53)
04400               10,1;
04500           S-101!-7!10(380,0,90:9:28:53)
04600               10,1;
04700           S-5!-7!30(380,17,90:9:28:53)
04800               30,1;
04900       SAFETY(1)
05000           V0!0!0(10000,500,90:9:28:53)
05100               UNSAFE,1;
05200       WHEEL!DEST(0)
05300       SUPPORT!VAR(1)
05400           V0!0!0(3915,83,90:9:28:53)
05500               20,1;
```

REFERENCES

1. McCarty, J., and Hayes, P.S., "Some Philosophical Problems From the Standpoint of Artificial Intelligence," Stanford University, Artificial Intelligence Lab., A.I.M.-73, Nov. 1968.

2. Nilsson, N., Problem Solving Methods in Artificial Intelligence, McGraw-Hill Book Co., Inc., New York, 1971.

3. Rabin, M.O., "Theoretical Impediments to Artificial Intelligence," Proceedings of I.F.I.P., 1974, pp. 615-619.

4. Feldman, J., and Yakimovsky, Y., "Decision Theory and Artificial Intelligence: A Semantics Based Region Analyzer," A. I. Journal, 5 (1974), 349-371.

5. Shortliffe, E.H., Mycin: A Rule-Based Program for Advising Physicians Regarding Antimicrobal Therapy Selections, Stanford University Technical Report STAN-CS-74-465, 1974.

6. Kunii, T., Weyl, S., Tenenbaum, J., "A Relational Data Base Schema for Describing Complex Pictures With Color and Texture," in Proceedings of the Second Joint International Pattern Recognition Conference, Aug. 1974, p. 310.

7. Barrow, H., Popplestone, R., "Relations Description in Picture Processing," Machine Intelligence, 6, pp. 377-296, B. Metzer (ed.), American Elsevier Pub. Co., N.Y., 1971.

8. Winston, P., "Learning Structural Description From Examples," Mac-TR-76, Massachusetts Institute of Technology, Artificial Intelligence Lab., Sept. 1970.

9. Minsky, M., A Framework for Representation of Knowledge, TR-306, Massachusetts Institute of Technology, Artificial Intelligence Lab., 1974.

10. Van-Lehn, K., et al., "SAIL Manual," Stanford University Technical Report CS-STAN-73-373, July 1973.

11. Feldman, J., and Rovner, P.D., "An Algol-Based Associate Language," C.A.C.M. Vol. 12, No. 8, pp. 439-449, 1969.

12. Yakimovsky, Y., Eskanazi, R., and Rayfield, M., RAPID - A Random Access Picture Digitizer Display and Memory System, Technical Memorandum 33-772, Jet Propulsion Laboratory, Pasadena, Calif., April 1976.

13. Yakimovsky, Y., "Boundary and Object Detection in Real World Images," to appear in JACM. Also appears in Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Russia, Advance Papers. Cambridge, Mass., International Joint Council on Artificial Intelligence, 1975.

14. Yakimovsky, Y., and Cunningham, R., On the Problem of Embedding an Image Point in a Region, Technical Memorandum 33-774, Jet Propulsion Laboratory, Calif., April, 1976.

15. Y. Yakimovsky and R. Cunningham, A System for Extracting 3-Dimensional Measurements from a Stereo Pair of TV Cameras, Technical Memorandum 33-769, Jet Propulsion Laboratory, Pasadena, Calif., April 1976.

16. Zadeh, L.A., "Fuzzy Sets," Information and Control, Vol. 8, pp. 334-353, 1965.

17. Yakimovsky, Y., Scene Analysis Using Sematic Base for Region Growing, Stanford University Artificial Intelligence Lab., A.I.M. -209, June 1973 (Ph.D. Thesis).

18. Michalski, R., "Variable Valued Logic and Its Application to Pattern Recognition and Machine Learning," in Monograph Multiple Valued Logic and Computer Science, American Elsevier Pub. Co., New York, 1975.

19. Fu, K.S., Sequential Methods in Pattern Recognition and Machine Learning, Academic Press, New York, 1967.

20. Yakimovsky, Y., Nondeterminant Data Base for Computerized Visual Perception, Technical Memorandum 33-761, Jet Propulsion Laboratory, Pasadena, Calif., January 1976.

NASA—JPL—Coml., L.A., Calif