UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AI-TR-345	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A Computer System for Visual Recognition Using Active Knowledge		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Eugene C. Freuder		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0643
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, Massachusetts 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd Arlington, Virginia 22209		12. REPORT DATE June 1976 13. NUMBER OF PAGES 278
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* Office of Naval Research Information Systems Arlington, Virginia 22217		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Artificial Intelligence
Domain Directed Processing
Heterarchical Control Structures
Machine Vision
Representation of Knowledge
Scene Analysis
Visual Recognition

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

A system for visual recognition is described, with implications for the general problem of representation of knowledge to assist control. The immediate objective is a computer system that will recognize objects in a visual scene, specifically hammers. The computer receives an array of light intensities from a device like a television camera. It is to locate and identify the hammer if one is present. The computer must produce from the numerical "sensory data" a symbolic description that constitutes its (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

A COMPUTER SYSTEM FOR VISUAL RECOGNITION

USING ACTIVE KNOWLEDGE

by

Eugene C. Freuder

Massachusetts Institute of Technology

June 1976

Revised version of a dissertation submitted to the Department of Electrical Engineering and Computer Science on May 2, 1975 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

A system for visual recognition is described, with implications for the general problem of representation of knowledge to assist control. The immediate objective is a computer system that will recognize objects in a visual scene, specifically hammers. The computer receives an array of light intensities from a device like a television camera. It is to locate and identify the hammer if one is present. The computer must produce from the numerical "sensory data" a symbolic description that constitutes its perception of the scene. Of primary concern is the control of the recognition process. Control decisions should be guided by the partial results obtained on the scene. If a hammer handle is observed this should suggest that the handle is part of a hammer and advise where to look for the hammer head. The particular knowledge that a handle has been found combines with general knowledge about hammers to influence the recognition process. This use of knowledge to direct control is denoted here by the term "active knowledge". A descriptive formalism is presented for visual knowledge which identifies the relationships relevant to the active use of the knowledge. A control structure is provided which can apply knowledge organized in this fashion actively to the processing of a given scene.

Thesis Supervisor: Patrick H. Winston

Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgements

Tomas Lozano-Perez and Tim Finin for moral support.

Patrick Winston, my thesis advisor; Marvin Minsky and B.K.P. Horn, the readers.

Tim Finin, Jeff Hill, John Hollerbach, Pitts Jarvis, Mark Lavin, Jerry Lerman, Tomas Lozano-Perez, Drew McDermott, David Silver, Bob Woodham for theoretical and practical assistance.

Steve Slesinger for programming assistance; Mona Campanella for thesis illustrations and Suzin Jabari for final TR illustrations; Steve Slesinger and Bob Tycast for the photographs.

Rita and Evan Freuder for support, understanding and perspective.

Dedication

To ELF and the 3 R's

Table of Contents

1

INTRODUCTION

AN OVERVIEW

WE ARE CONCERNED WITH VISUAL RECOGNITION AND WITH LARGER ISSUES OF DESCRIPTION AND CONTROL

The immediate objective of this work is a computer system that will recognize objects in a visual scene, specifically hammers. Its "eye" is a device something like a television camera. The work is directed at the visual variety of everyday objects, not prepared or stylized models. The system must produce a symbolic description, that constitutes its "perception" of the scene, from the "sensory data", numbers representing light intensities, that the camera provides.

Of primary concern is the control of the recognition process: what to do, how and when to do it. These decisions are guided by the partial results obtained on the scene. If a hammer handle is observed this suggests that the handle is part of a hammer and advises where to look for the hammer head. The particular knowledge that a handle has been found combines with general knowledge about hammers to influence the recognition process. This use of knowledge to direct control is denoted here by the term active knowledge.

A descriptive formalism is presented for visual knowledge which identifies the relationships relevant to the active use of the knowledge. A control structure is provided which can apply knowledge organized in this fashion actively to the processing of a given scene.

ACTIVE VERSUS PASSIVE KNOWLEDGE

Active knowledge might seem an obvious control principle. However, it is possible to employ control algorithms which depend very little on either partial results or the nature of the objects to be recognized. Early efforts in both pattern recognition and artificial intelligence employed knowledge in a very "passive" role. They built a scene description by basically filling out a checklist of possible

features, and then matched the results against stored models.

In the early Edinburgh system [Barrow and Popplestone 1971], for example, recognition was a graph matching process that compared descriptive networks of a scene and a model. Both hammer head and hammer handle had to have been found independently before the match began. If the head were to be confused with the background in a scene, the match with the hammer model simply would not succeed. The presence of the handle did not direct the system in a search for the head. (In fact, the Edinburgh system, while quite successful for its time, did fail badly on even a simplified toy hammer, largely because the descriptive stage often failed to find both the handle and head regions. However, see [Milner 1970] for a more active approach.)

Such problems of accumulating error are one motivation for the active knowledge approach. One can also argue that it is more efficient for partial results to make suggestions on what to look for and how to look for it, as opposed to a system that merely looks for everything it knows is possible, in a uniform manner. For example, a line finder sensitive enough to find very faint lines is too costly to be employed everywhere in a scene (and has the additional problem that it may find too many unimportant lines). However, it can be directed to verify a specific missing line suggested by previously found lines which partially complete a general model [Minsky and Papert 1967].

Nevertheless, an argument can be made for passive knowledge. We cannot start by asking if there is a hammer handle at point (500, 500) in the scene. As far as efficiency goes, the overhead of clever control structures may threaten to offset any savings over "brute force" uniform procedures.

There are clearly some areas in which active knowledge comes into play, even if it is only at a relatively high level, in terms of concepts like context, expectation, focus of attention. The question is where to use passive and where active knowledge. Also, once passive processing areas are delimited, can there be any flow back into them? Must we simply accept and work with the results of the passive segments, or can other problems motivate a return to modify the results, or to consult directly with the primitive input data for these segments? If three regions are found in a passive pass, can we go back and question if one is really there, or find a fourth one?

The system described here, which is called SEER, has a minimal passive initial stage and does not regard the results of this stage as at all final. An argument is made for extensive use of active

knowledge. However, the issues involved have not been fully resolved, by any means; much more experimental work and theoretical analysis will be required.

Active knowledge is a control principle that may be used to relate chunks of knowledge at any level of processing; it does not apply only to application of very "high level" general knowledge, e.g. models of the objects being viewed. Lower level research, e.g. David Marr's work [Marr 1975], is not incompatible with an active knowledge approach. (However, if we assign visual knowledge positions along an axis from the lowest level raw input to the highest level most specific identifications, Marr would seem to prohibit the filtering downward of constraints over any great distance along this axis.)

Consider, for example, the following line finding algorithms. Both operate on an array of feature points, i.e. suspected edge points; neither involves knowledge of specific models like cubes and wedges. Algorithm A runs a transformation across the array operating on all feature points in parallel. Every feature point has an initial "strength". Each point is assigned a new value by a function which depends on the surrounding points. This process is repeated several times. Lines are then defined as connected sets of feature points whose strength is above a set threshhold. The processing sequence here is fixed, irrespective of the specific scene, the results being obtained, or the model of feature points and lines embodied in the function applied at each point. (See [Zucker, Hummel and Rosenfeld 1975].)

Algorithm B "tracks" lines through the feature point array. Suppose at some point in the processing it has grouped several points into a line and is attempting to extend that line. It looks at the feature points surrounding the endpoint of the line. If one of those feature points "matches" the feature points already in the line, e.g. all suggest the same type of edge, and the feature point has sufficient strength (where the strength threshhold depends on the length of the line found so far), then the point is added to the line and tracking continues. (If several points will do, the "best" is chosen.) If tracking stops, an attempt is made to find the beginning of a new line. Clearly what computations are performed and in what order will vary widely in this algorithm from scene to scene, and is dependent on the results obtained and the general knowledge of line models (e.g. types of edges).

The same knowledge may be <u>used</u> in both algorithms. Only in the second one, however, does it

affect the flow of control; only algorithm B provides an example of active knowledge. One can analyze the various computational tradeoffs here. More generally we can ask how easy or efficient it is to imbed knowledge in active or passive forms. This will depend on the problem domain and on the specific active or passive implementation. (SEER embodies a specific implementation of active knowledge.) At the moment we do not have the tools to answer such questions very well, though some general arguments can be made. A reasonable conjecture at this point might be that many low level computations will prove susceptible to passive implementations while higher level problems will benefit from an active approach.

VISUAL KNOWLEDGE IS ORGANIZED TO ENCOURAGE ITS ACTIVE USE

Given a commitment to active knowledge how do we implement it? When SEER discovers that "region R is bar shaped" how does it use this information? General visual knowledge is organized to respond to these questions.

A data base object exists representing "bar shaped". This object knows how a bar shape is useful for acquiring further knowledge of a region. It knows, for example, that hammer handles are bar shaped. This knowledge is represented by a link between data base objects representing bar shaped and hammer handle:

HANDLE --- BAR-SHAPED

ESTABLISHED RESULTS ARE EXPLOITED TO MAKE FURTHER SUGGESTIONS

When a particular region R is found to be bar shaped, this knowledge is exploited as follows. The bar shaped element of general knowledge is consulted. The link to hammer handle provides the suggestion that R is also a hammer handle. This suggestion becomes an object in the particular knowledge data base. It begins as a "conjecture", but eventually may become a "success" or a "failure". (The fact that R is bar shaped is a successful particular knowledge object.) A link is placed between BAR-SHAPED R and HANDLE R corresponding to the general knowledge link between BAR-SHAPED and HAMMER HANDLE:

HANDLE R --- BAR-SHAPED R

BAR-SHAPED is also linked to HEAD in the general knowledge: the exploitation of BAR-SHAPED R will also suggest HEAD R, and the appropriate particular knowledge conjecture and link will be created:

```
HANDLE R \
          BAR-SHAPED R
HEAD R   /
```

GENERAL VISUAL KNOWLEDGE AND PARTICULAR SCENE RESULTS RESIDE IN INTERACTING NETWORKS

We have then two knowledge structures, which we will abbreviate as the GK (general knowledge) and the PK (particular knowledge) structures. Both are networks: the nodes represent items of visual knowledge, objects, properties, relationships, and the links indicate how these items help establish each other.

The knowledge involved is not very esoteric. It is based on functional definitions: a hammer requires a handle to hold and a head with which to hit; the two should be set at right angles to transfer a swinging motion into a blow; the head requires a striking face at one end; it should be flat to contact the nail easily; and so on. The concern is with the organization of this knowledge. Standard descriptive networks, where nodes represent parts and links represent properties and relationships, are good for passive matching. SEER's networks facilitate active knowledge.

The GK represents potential programs for recognizing specific instances of the GK concepts. We write these programs by representing possible descriptions of the concepts in a network form. These descriptive nets seem a natural formalism for the procedural embedding of knowledge [Hewitt 1972] in the domain of visual recognition (as opposed to theorems, for example). Similarly the PK represents, at any point of processing, the current state of description of the scene, and the state of the specific processes chosen to work on it.

The interactions within and between these structures implement active knowledge, and we have seen a basic example: a PK success consulting GK to suggest further PK conjectures.

CONJECTURES ARE ESTABLISHED BY EXPLORATION

We have seen the basic mechanism for exploiting results, but how do these results get established; once a suggestion is made, how is it pursued? The same link that indicates that a bar

shaped region may be a handle, also says that if we wish to establish a region as a handle, we should see if it is bar shaped. Thus, analogous to exploitation there is an exploration process. When the PK conjecture HANDLE R is explored, it consults the GK object HANDLE, which follows the link to BAR-SHAPED, providing the suggestion BAR-SHAPED R, which enters the PK as a conjecture linked to the HANDLE R conjecture:

HANDLE R --- BAR-SHAPED R

The links have a direction: if B helps to establish A we say that B is "below" A. Exploitation involves looking upward along links, exploration downward.

Exploration of HANDLE R also suggests LONG-AND-THIN R:

```
          / LONG-AND-THIN R
HANDLE R
          \ BAR-SHAPED R
```

Let us say that the two properties, bar shaped and long-and-thin, are sufficient to define a hammer handle; the GK knows this. Now if either of the new conjectures, BAR-SHAPED R or LONG-AND-THIN R, are established, they will be exploited. We did observe earlier that exploitation of BAR-SHAPED R involves creation of a new PK object for HANDLE R. However, obviously this will not be necessary now, as HANDLE R already exists as a PK conjecture. What will happen rather is that the HANDLE R conjecture will be found and told that one of the properties it needs to succeed has been established. The GK has told HANDLE R to wait for two results (it is an "AND gate" if you like); it will now count one and wait for the other. When that comes, HANDLE R will in turn succeed and be exploited.

That is how HANDLE R gets established; but what about BAR-SHAPED R and LONG-AND-THIN R, which must be established first? Clearly we can repeat the exploration process, but it must terminate somewhere. Exploration of LONG-AND-THIN R leads to three further conjectures:

```
                              / LENGTH R
            / LONG-AND-THIN R - WIDTH R
           /                  \ LONG-AND-THIN-COMPUTATION R
HAMMER R
           \
            \ BAR-SHAPED R
```

Let us suppose, as might be the case, that LENGTH R and WIDTH R are already present and successful in the PK. The exploration of LONG-AND-THIN R would not duplicate them; they would be found and linked to, and the new links exploited by notifying LONG-AND-THIN R that two of its

requirements have been met. (The actual numerical values of the length and width of R will have been stored as properties of the corresponding PK elements.)

LONG-AND-THIN-COMPUTATION R, when explored in turn, does not pursue further conjectures, but actually compares the length and width with a threshhold and either succeeds or fails on the spot. A success will complete the establishment of LONG-AND-THIN R. All PK trees eventually lead down to such terminal computation nodes.

GLOBAL MONITOR AND PRIORITY MODULES CHOOSE WHICH SUGGESTIONS TO EXPLORE

We see by now that there will be a lot of suggestions floating around, proposed by results or pursued by hypotheses. These suggestions form a "pool" of processing possibilities; this pool concept is important because it eliminates the need for suggestions to be taken up or rejected immediately. Instead they can accumulate and "compete", as further results improve our ability to discern the most promising lines of inquiry. We know how to explore and then exploit the suggestions, but the question remains: when to explore them, or indeed, which to explore.

Lurking in the background are critical doubts. Can we choose intelligently, or will the number of suggestions "blow up" to an unmanageable number? The suggestion making processes, exploration and exploitation, are basically local operations, centered around the individual results and pieces of general knowledge. One can argue the advantages of such a uniform, modular structure; but how is chaos avoided: is some global regulation imposed?

Basically, we are hoping for something of a "Waltz convergence effect" [Waltz 1972] on the pool of suggestions. (However, it should be noted that this research does not focus on the problems raised by very large data bases. These will probably require further means of partitioning our knowledge, e.g. into frames [Minsky 1974]; SEER's basic interest is fruitful interaction within our knowledge.) Waltz' program may be viewed as carrying a large number of line labelling conjectures, based on partial evidence, forward in parallel. Since the legitimate combinations of labels are limited, as processing proceeds more possibilities are ruled out than are added, and only one or two complete labellings for the scene are left.

In SEER a number of conjectures are also carried forward in parallel. Here too new results can

contradict old conjectures. However, we emphasize more the positive effect that new results can have in encouraging previous conjectures, directing our attention to the most promising and the most efficient suggestions. A priority system is employed for this purpose. A monitor uses the priority information to determine the order in which suggestions will be explored. These are the global elements of the system, which evaluate the local activity.

The priority of a PK conjecture is basically a cost/benefit analysis. The likelihood of a conjecture holding true is compared with the expected difficulty in establishing it. These figures can change as results accrue. The conjecture that R is a hammer is initially difficult and not too likely (though its high "interest" for us can also be taken into account). When R is found to have a hammer handle, the difficulty of the hammer conjecture goes down while its likelihood goes up, thus its overall priority increases. (It is, again, the exploitation of the hammer handle success which induces that change in priority.)

To the monitor the PK looks something like a mountain range. The peaks represent those nodes which have as yet nothing above them in the link structure. Below each peak are linked nodes, and nodes linked to them in turn, continuing until we reach as yet unexplored nodes, with no further links below them. This structure of nodes below the peak represents the state of the investigation of the peak node. These structures blend into one another, especially in the "foothills", e.g. CONVEX R is likely to appear in several investigations.) The monitor must first decide which investigation is most promising, then which suggestion within that investigation to explore next. It chooses to further investigate the peak node of highest priority. Choosing the next suggestion to explore is not quite so simple, but basically we want to explore the easiest, least likely suggestion first, in order to resolve the investigation most quickly.

PROCESSING CYCLES THROUGH SUCCESSIVE PK STATES

After the initial passive stage, in which regions of potential interest are found, an initial PK state is established with a few "seed suggestions" about prominent regions. These are "primitive" conjectures like CONVEX R. CONVEX is primitive in the sense that there is essentially nothing below it in the GK network except properties like area and boundary which always can be established. From

this point on, processing entails active knowledge, use of previous results and general knowledge.

Processing consists of a sequence of cycles, which cary us from one state of the PK to the next. Each cycle begins with the exploration of a suggestion chosen by the monitor. There may follow one or more exploitations if successful results are obtained. (A similar process makes use of failures.) Exploration and exploitation will change the PK: new conjectures may be added, new links, priorities may change, conjectures may become successes or failures. At the conclusion of the cycle, control returns to the monitor to begin the next cycle. (See figure 1 - 1 .)

Thus we have continual review of local interactions by the global mechanisms. SEER cannot continue blindly down one avenue, but must evaluate the impact of results in a wider context. Also while activity within a cycle is local in the sense that it is centered on individual nodes, this activity is not at all parochial. There is not, for example, the usual "master/slave" relationship between processes. A result does not report only to the calling process, but is a gregarious fellow who happily tells anyone who might be interested, to whom he might be useful, of his good fortune. Even a success in one investigation, may when exploited have a greater effect on another investigation; the monitor can observe this and switch its attention to the second investigation. In any case, as investigations succeed, they in turn spawn larger investigations, until eventually we reach a satisfactory level of recognition, e.g. find the hammer.

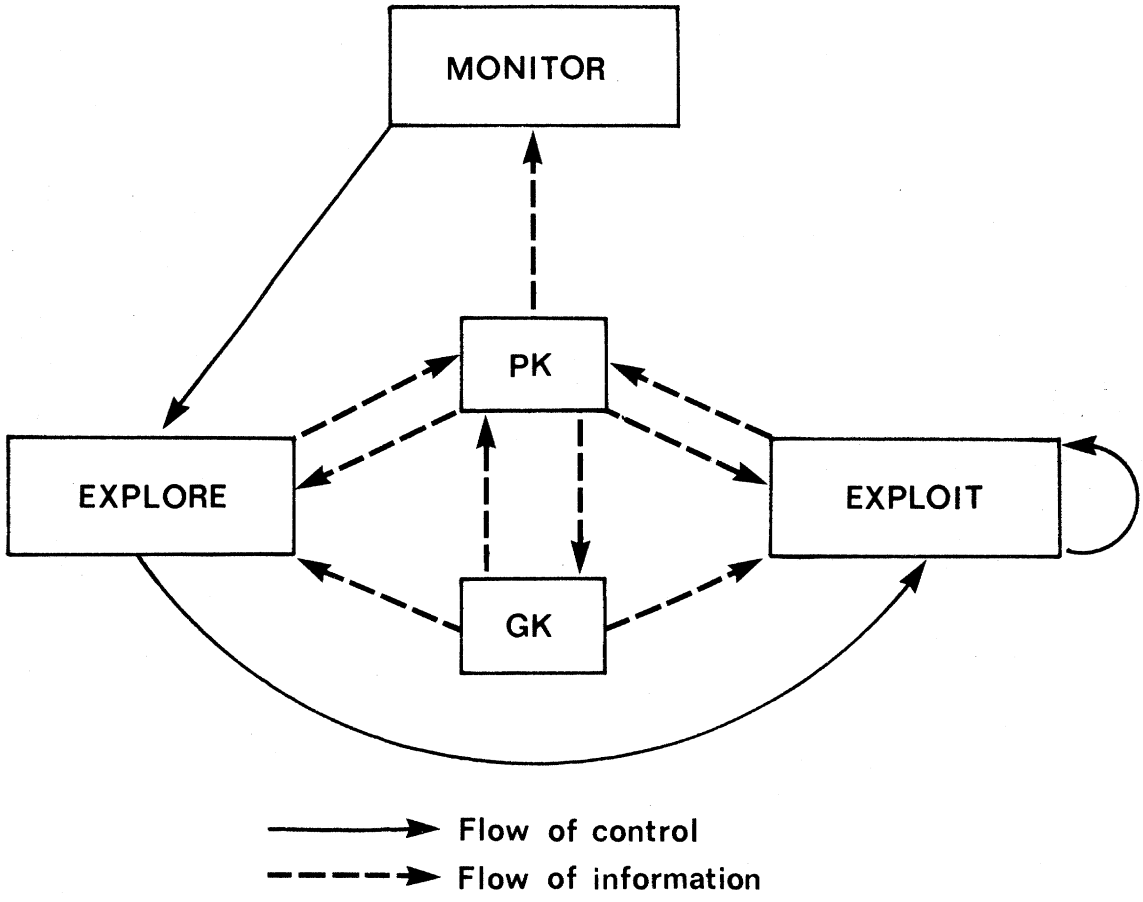
METHODS ADVISE HOW TO ESTABLISH RESULTS

We now have an idea of SEER's approach to what to do and when to do it. There remains the important question of how to do it. SEER makes suggestions about what to do, it also gives advice on how to do it. Actually the primary implementation of advice is as a form of suggestion. SEER suggests methods for establishing results.

A method is also a node in the GK or PK structure. The discussion up until now may have given the impression that these structures are intertwined trees of AND gates. Actually they involve AND/OR trees. An AND node links to a set of subnodes which together define or establish the node, a set of properties or parts; we have seen examples. An OR node links down to a set of subnodes each of which is an alternative method for establishing the node. Thus HANDLE may actually link to

FIGURE 1-1

CYCLE



HANDLE-METHOD-1, HANDLE-METHOD-2, etc. HANDLE-METHOD-1, for example, may involve finding the handle when we already have the head to advise us where to look. When the head is found the exploitation of that result will suggest HANDLE-METHOD-1. This approach to advice is very explicit; advice on how to do something is a suggestion of a specific method for doing it.

AN EXAMPLE

AN EXAMPLE WILL ILLUSTRATE MORE OF SEER IN OPERATION

As in the AND/OR business, we began this discussion by simplifying SEER's structure and operation as much as possible, and the picture has only been complicated as absolutely necessary. For example, we eventually observed that exploitation involved updating priorities as well as creating new conjectures; however, we avoided the fact that priority activities are carried out by following a set of priority links which are distinct from the links that indicate how items define one another. (Most often, these two link types will occur together; however, they do represent distinct relationships which have been extracted from the basic concept of "relevance" between pieces of visual knowledge.) This is not the place for full detail; however, an example of SEER's operation is in order which will provide a feeling for the types of problems and solutions which arise.

AN AFFINITY PASS PROVIDES AN INITIAL SET OF REGIONS

Consider a scene consisting of a ball peen hammer on a wooden work bench. First SEER makes a region growing pass over the scene using what we term an affinity process. Briefly, rather than seeking absolute uniformity, which may not be present in the surfaces of a realistic object, the process attempts to find regions which are relatively homogeneous, whose pieces seem to hang together more than they seem to belong with other neighboring pieces of the scene. Rather than finding one division of the scene, the process builds up a tree of potentially interesting regions. Below each region in the tree are the subregions which were merged to form it. The merging process works upwards from a initial division of the scene into small cells, forming successively larger regions.

SEER CHOOSES A PROCESSING SEQUENCE, BASED ON THE GK STRUCTURE AND DEVELOPING RESULTS IN THE SCENE: SEER "ZIGZAGS" GRADUALLY UPWARD FROM THE SEED SUGGESTION THAT REGION 196 IS CONVEX, TO A CONJECTURE THAT THE REGION IS A HAMMER HEAD

The regions that "stand out" in the scene will be found near the top of the tree, and so we begin by making primitive conjectures about these. In this case the initial PK will consist of seed conjectures for regions R196, R191, R185, R140. SEER explores the conjecture that region 196 is convex: CONVEX R196. Figure 1 - 2 shows this region superimposed on the scene. The exploration of CONVEX R196 leads to the PK structure:

```
                / CONVEX-NEEDS R196
CONVEX R196
                \ CONVEX-COMPUTATION R196
```

CONVEX-NEEDS represents the items needed to perform the computation CONVEX-COMPUTATION which will judge convexity. The computation cannot be made until these items are found, so the GK node CONVEX, and this PK instance, CONVEX R196, are "serial" nodes. This means that the process of choosing which suggestion to explore next is constrained at this point. If SEER wishes to further investigate CONVEX R196, it must establish CONVEX-NEEDS R196 before it can explore CONVEX-COMPUTATION R196. CONVEX-NEEDS is explored in the next cycle and the PK investigation of CONVEX R196 expands:

```
                              / CONVEX-HULL R196
                / CONVEX-NEEDS R196
               /              \ AREA R196
CONVEX R196
               \
                \ CONVEX-COMPUTATION R196
```

Both AREA and CONVEX-HULL must be found in order to satisfy the AND node CONVEX-NEEDS R196; however, either may be found first in this case. In writing up this portion of the GK, we do not need to specify a serial order, but can program in parallel, in some sense. Neither do we need to consider explicitly all the different possible execution time conditions that might influence the decision on which to explore first: "if A has been found, do B, however if C ..." The relevant conditions will be reflected in the state of the PK, and the ad hoc local decisions we might make have been generalized and given to SEER's priority and monitor modules. These can take a global view, based on the state of processing at decision time.

FIGURE 1 - 2
REGION 196

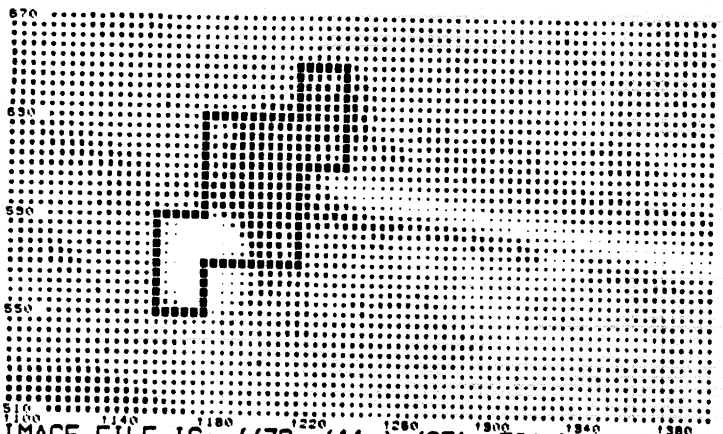


IMAGE FILE IS: ((76. 411.) (971. 794.) POINTS H1 DSK FREUDE)
SELECTED WINDOW IS: ((100. 510.) (400. 670.))
RESOLUTION IS: 4.0

***** LEGEND *****

- 321. :
321. - 341. :
341. - 361. :
361. - 381. :
381. - 401. :
401. - 421. :
421. - 441. :
441. - 461. :
461. - 481. :
481. - 501. :
501. - 521. :
521. - 541. :
541. - 561. :
561. - 581. :
581. - 601. :
601. - :

If AREA R196 has been established already, attention will automatically focus on CONVEX-HULL R196. Perhaps neither is yet a success, but CONVEX-HULL R196 has been partially established, a subnode has succeeded. The exploitation of that subnode will have improved the priority of CONVEX-HULL R196. This may influence the monitor to choose to explore CONVEX-HULL R196 before AREA R196. Of course, in this example no results have yet been obtained, and SEER chooses which node to explore based on initial priorities found in the GK. After SEER begins developing one branch, however, it is not locked into that direction. Perhaps an easy method for obtaining the convex hull fails, priorities shift, and the monitor can shift attention to finding the area. As it happens, finding the area and convex hull are rather straightforward, and R196 is in fact determined to be convex.

SEER now begins a typical "zigzag" upward flow. Exploiting CONVEX R196 leads to several suggestions being placed above CONVEX R196 in the PK. One of these is chosen and investigated. Several cycles of explorations expand the investigation back downward in the PK. Finally the investigation succeeds and exploitation again expands the PK upward. Once again an investigation is chosen to explore, and so it goes. This basic flow is, of course, subject to many alterations; results within an investigation can also propose new investigations; most significantly SEER can choose to shift attention between investigations, just as it can between branches within an investigation.

Notice that the processing flow is neither strictly "top down" nor "bottom up" but more "middle out" [Hewitt, Bishop and Steiger 1973]. A top down control structure can direct its computations very efficiently. In the extreme, we know precisely what we are looking for and can employ a sensitive template to merely verify its presence. A top down system, however, lacks generality, because it begs the question of how to choose the starting point in an environment with multiple possibilities. SEER attempts a compromise. A result proposes a higher hypothesis, which can then be used as a context for proceeding downward. Our view of knowledge elements as "little men" [Papert 1972] who know how to explore and exploit themselves, facilitates this middle out model.

SEER soon establishes that region 196 is a "bar", a basic shape category. This result proposes several interesting suggestions: R196 is a hammer handle, an occluded hammer handle, a hammer head. The first two are explored but quickly lead to failures.

INSTANTIATION FUNCTIONS ARE USED TO GENERATE ARGUMENTS FOR SUGGESTIONS: A REGION IS FOUND FOR THE FACE OF THE HAMMER HEAD, AND THEN ANOTHER IS CONJECTURED FOR THE HAMMER ITSELF

The GK indicates that to establish that R196 is a hammer head, SEER must establish that R196 has a striking face. Here we are forced to confront another major factor which was glossed over earlier: the generation of arguments. Up until now we have used the same argument, R196, in all our PK statements. There has been no problem in making new suggestions; exploring CONVEX-NEEDS R196, SEER looks at the GK node for CONVEX-N, finds AREA node below it, and suggests AREA R196, what else? But IS-A-HEAD R196 cannot suggest IS-A-HEAD-FACE R196.

Actually the GK nodes also have arguments, variable arguments: CONVEX-NEEDS X, AREA Y, IS-A-HEAD Z. Attached to the links in the GK are functions, instantiation functions, which indicate the relationship between the arguments of connected nodes. Very often the instantiation function is simply the identity function, I. Thus when SEER explores CONVEX-NEEDS R196, it consults CONVEX-NEEDS X in the GK, follows a downward link to AREA Y, and then applies the instantiation function I to the value associated with X in the PK, R196, to obtain a value for Y: I(R196) = R196. The resulting statement AREA R196, is one of the suggestions to be made in exploring CONVEX-NEEDS R196.

These instantion functions can be arbitrarily complex; however, we try to keep most of the work involved on the system in the nodes. Thus a common GK structure is generate and test:

```
          / FOO-GENERATE A
FOO A
          \ FOO-TEST A
                       \ ZOT B
```

Here the FOO-GENERATE A node computes B and stores it as a property of the node. The instantiation function on the GK link from FOO-TEST X to ZOT Y picks up B from the property list. B is then tested for the ZOT property.

Often an additional complication is the need to generate several alternatives for testing. If we search for the hammer head face in the original affinity region tree, for example, we want to allow for several tries at finding a region with the right properties. There are various options. The function on the link between FOO-TEST and ZOT can be multivalued, for example. Though ZOT Y only appears once in the GK, several instances of it may be suggested when we explore a FOO-TEST node in the

PK. (FOO-TEST will be an OR node obviously.)

It is often painful to generate every option at once, however, and a looping mechanism must be employed. There are methods of implementing these in SEER, and one is in fact utilized in searching for the hammer head face in this case. However, they are a bit awkward at present, and such technical details should not delay us here. Suffice it to say that the face of the hammer is indeed found in the affinity tree, region 145. Actually this tree search is just one method for finding a hammer head face: HAS-A-HEAD-FACE-METHOD-1 R196 was the successful node. In general other techniques are possible for generating desired regions. We are about to see one of these.

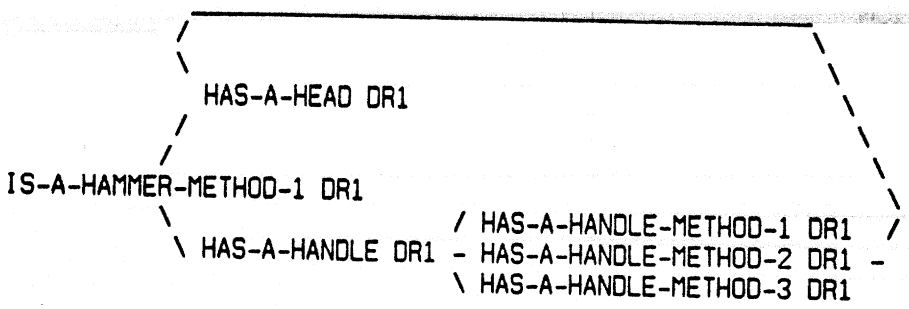
When IS-A-HEAD R196 succeeds, it suggests the presence of an encompassing hammer, and advises a method for finding the hammer handle.

Actually, there is an intermediary step. IS-A-HEAD R196 suggests HAS-A-HEAD DR1, region DR1 has a hammer head as a subregion. This again is a situation where a new argument is required. One of the obvious things to do is to generate a "dummy" argument. DR1, "dummy region one", stands for the hammer region which SEER hopes to define by the union of R196 and another region for the handle, yet to be found. The instantiation function in the GK link between IS-A-HEAD and HAS-A-HEAD supplies the argument DR1.

RELATIONSHIPS PROVIDE ADVICE: THE ESTABLISHED HEAD HELPS FIND THE HANDLE

HAS-A-HEAD DR1 succeeds immediately; normally a method for establishing HAS-A-HEAD will need to prove proper relationships hold between the head and the hammer region (or the handle region). However, this is obviously not an issue now. When the handle is found the proper relationships will be checked. Better yet we will use these relationships to guide the search for the handle. Thus the exploitiation of HAS-A-HEAD suggests HAS-A-HANDLE-METHOD-2, a method which uses the head to find the handle. Also suggested is IS-A-HAMMER-METHOD-1 DR1.

Investigating IS-A-HAMMER-METHOD-1, SEER suggests several methods for finding the handle but method two obviously has priority at this point:



This method is based on a program developed by Tomas Lozano-Perez which uses an ATN grammar to look for regions with specified intensity profiles [Lozano-Perez 1975]. The program requires a starting point and a direction in which to look, along with a specification of the type of intensity profile being sought. SEER uses the head, R196, which it has already found, and the expected relationship between hammer heads and hammer handles to determine a starting point and direction to send to Lozano's program, along with the characteristic curved profile of a plot of light intensities across the width of a hammer handle. Having hypothesized what we are looking for, and having been guided in where to look, we are able to use this specialized technique, and it is successful. A region NR1, "new region one", not in the original affinity tree, is returned. A few additional checks ensure that it will serve for the hammer handle.

Notice that the basic relationships between hammer and head do not need to be checked, they have already been verified in the process of generating NR1. Contrast this with the distinct passes of some passive knowledge systems: a descriptive phase would find regions and determine relationships, then a recognition phase would match these relationships with those in a model. In SEER description and recognition tend to merge together. The importance of this merging is one reason SEER was designed to work directly from camera input, as opposed to a "hand coded" symbolic description of the scene. We do not wish to beg the key question of acquiring a perceptual description from the sensory data; rather we wish to bring recognition level knowledge to bear on it. The interaction with raw data keeps us honest and ensures that our results have a bearing on real visual problems.

SEER now has the entire hammer and the recognition is complete.

EVALUATION

SEER PERMITS A FLEXIBLE RESPONSE TO THE VARIETY IN REALISTIC SCENES

It should be emphasized that the above example does not present the way that SEER recognizes hammers. One of SEER's aims has been to deal with realistic scenes and to provide a system with the flexibility to deal with the variety that realistic scenes hold. (Even a single hammer can be given many different appearances by varying its orientation, background, lighting.) SEER's modular organization allows us to program basic alternatives, then add others incrementally as we expand SEER's competence. SEER's mode of processing, instantiating successive pieces of the modular GK structure, which is both program and description, allows it to enter the "hammer program" at a variety of points and proceed in a variety of paths through the structure, exploring and establishing different subsets of the "hammer description" in different orders--all in response to the results being obtained on a given scene.

Thus, in the example we discussed, SEER began by working up to a head recognition, then moved back down to find the handle. The handle was narrow and angled sharply, its wooden texture blended into the workbench background. It might have been hard to recognize; it was not found in the initial affinity pass. (A "finer" pass with this process might have suceeded, but could be prohibitively expensive in time and space.) However, with the advice provided by the previously found head, the handle was found, and relatively easily.

We could have discussed another scene in which the light wooden handle stands out while the rusty head blends into the dark background. In this scene SEER identifies the handle first, using a different method than the one employed here, and then goes on to find the head.

SEER ANALYZES AND IMPLEMENTS ACTIVE KNOWLEDGE

SEER is a programming language or system that provides a framework for active knowledge programming, in the sense that a language like PLANNER [Hewitt 1972] identifies and implements various processing methods suitable for problem solving activity. Some previous vision systems pioneered by providing examples of active knowledge types of activity, but were not extensible or

general in their approach. The Wizard line finding system [Winston 1972b], for example, used lines, as they were found, to suggest the location of further lines. This was accomplished basically by assuming a parallelipiped environment and trying to find lines that completed parallelograms. However Wizard could not supply ready or direct answers to questions like: what about wedges; how do I add a facility for dealing with wedges to the system and integrate it with what is already there? What is the general nature of the suggestion process; how can I apply it to another body of knowledge; what kind of analysis of the knowledge is required; in what ways can I profit; what are the basic processes that are required to implement suggestions; how do I handle advice?

SEER attempts to provide tentative answers to questions like these. A language, GK, is provided for programming visual knowledge. The structure of the GK encourages the user to make use of active knowledge, to think in terms of questions like: if we have x, how does that help us, can it help us get y, is there an easy way to get y given x? The control structure processes the GK representation of relevance relationships and applies them to a scene, implementing active knowledge. SEER contains the basic atomic elements, notably exploitation, that one would want for an active knowledge based system. I have analyzed some of the functions, like suggestion and advice, that such a system should carry out, and provided a specific implementation as a concrete basis for further work.

In short you should be able to come to this work with the idea: "I want to attack my problem with active knowledge." The response should be: "Fine. Here is how to organize your knowledge; give the result to a system with the elements of SEER, and the knowledge will be applied actively." It is encouraging to note that there is already in the literature a piece of vision research [Popplestone, Brown, Ambler and Crawford 1975] which is good enough to give partial credit for its control principles to an early version of SEER [Freuder 1973b].

SEER EMBODIES AN EXPERIMENTAL MODEL

SEER provides a primitive model of visual processing, subject to experimentation and development. This model can be quickly outlined:

SEER assumes the necessity of some low level passive processing. SEER's processing at this

the datum below. Thus we have "success-priority" (SP) links, which indicate that the success of the datum below affects the priority of the datum above, and also "failure-priority" (FP) links. Either failure or success of a datum B can help establish A; we have SE and FE links. Similarly, success (or failure) of the datum below may cause initiation of the datum above (SI FI). In addition, however, a conjectured datum may initiate datums below it (PUI). There is a "disestablishing" structure as well as the establishing structure. (See chapter three.)

Figure 2 - 7 presents some of the PK structure around D28 in more detail. It includes examples of many of the link types. (Since E links always imply P-links, we do not indicate those P-links separately.) A variety of properties are attached to the nodes and links; their purposes will become clearer as we proceed.

WHEN SEER ANALYZES A SCENE, THE PK REPRESENTS THE CURRENT STATE OF PROCESSING, AS WELL AS THE CURRENT STATE OF KNOWLEDGE ABOUT THE SCENE

A datum in the actuation structure with nothing above it in that structure, will be called the subject of an investigation. The nodes below it in the A-structure form the investigation. The nodes at the bottom of the investigation are called frontier nodes.

Various "paths" through the PK structure are of interest. A path is a connected set of nodes in a link structure, with associated information. Normally an investigation would be used as the basis set of nodes for a path. These paths provide interesting "traces" of the processing flow.

A path in the actuation structure numbers the set of nodes as to the order in which they were explored. A path in the I-structure indicates the order in which nodes were created. A path in the E-structure numbers the nodes to indicate the order in which they were established, and indicates which nodes helped establish which. These paths also include information on the cycle during which the node was initiated, explored, or established.

The numbering in the actuation or establishment paths may be merely consecutively ordinal within the path set. We may also number each execution or establishment during the overall processing of a scene. These execution or establishment numbers can be used in a path to reflect not only relative ordering, but the place of path elements in the larger processing run. (this will be the

Observation can help determine how sophisticated a control system is required for efficient operation.

SEER was intended to be a vision "laboratory", within which subproblems of region generation, texture, shape description, whatever, could be studied in an interactive setting. SEER encourages us to investigate and implement the ways in which results in one area of knowledge can help processing in another. It has long been suspected that independent "experts" in these areas will be insufficient. SEER tries to worry a bit "beforehand" about what the interaction between them should look like. Such analysis should assist the development of subproblem expertise in a form which can benefit from and provide "suggestion and advice". (Obviously further experience with the "pieces" of a vision system will in turn improve our understanding of the type of "glue" a system like SEER should provide.)

SEER'S PRACTICAL ACCOMPLISHMENTS ARE QUITE LIMITED

In practical terms SEER is quite primitive. It can recognize a hammer in a few scenes. The system is extensible, but it is still a considerable effort to expand its competence. The GK is an awkward language to program in at present. An intermediate program is needed to prompt the user and "compile" simplified input formats, e.g. for loops, into the proper GK network structure. (An earlier verison of SEER had a primitive input assistant of this sort.) The state of vision research is such that one still has to design and debug many basic visual mechanisms as one proceeds, e.g. for determining surface shape, particularly where one is looking for methods that can benefit from advice. Making and investigating wrong suggestions is painful, and the overhead of building and maintaining the PK data structure is significant: efficiency needs to be increased in many ways. For example, it is convenient but often wasteful to search the affinity tree in generating region arguments; too many useless regions can be conjectured about. The search process may be subject to improvement, some has already been made; but direct methods for verifying conjectured regions, like the Lozano program, should be more efficient, if they can be made as tolerant of nonuniformity as the affinity process is. Indeed it may prove that we need to be more definite about the organization of the scene into primitives units before beginning to make conjectures. Of course, given the difficulty of the visual processing problem, and the non-specialized nature of our computational hardware, we can

expect any reasonably general system to be rather demanding of time and space.

SEER is limited to hammers at the moment. Obviously the basic elements of the system, the control structures, data structures, region generation, etc., are quite general, and GK knowledge could be added for other objects. The system already deals with many different objects and properties, in the parts and features of hammers. There are issues that would be raised more forcefully by a more diverse and a larger domain. However, SEER has chosen to concentrate on those problems raised by the diversity inherent in different realistic examples of even a single class of object. SEER was tested on several scenes with real everyday hammers, with a couple of orientation, background and lighting combinations, and a case of occlusion; as opposed, for example, to several objects, a single stylized model of each, uniformly painted, and on a single contrasting background.

SEER EXTENDS AND IMPLEMENTS SEVERAL BASIC A.I. PRINCIPLES

The immediate impetus for SEER was the heterarchy concept of Minsky and Papert [Minsky and Papert 1967, 1970a, 1970b, 1972], and its implementations in the M.I.T. "copy demo" system of Winston, Horn, Freuder, et. al. [Winston 1972a]. Earlier ideas in domain directed processing [Ernst 1961; Simon 1969] also can be seen in the active knowledge approach. The basic implementation mechanisms of exploitation and exploration reflect the antecedent/consequent distinction most effectively analyzed by Hewitt [Hewitt 1972]. The development, or at least the description, of SEER has been influenced by several ideas that came to prominence during the progress of this work. The little man or actor model (Papert, Hewitt, Kay, etc. [Papert and Solomon 1972; Hewitt and Smith 1975]) provides a good metaphor for the local activity centered in the knowledge structure nodes. Minsky's frame model [Minsky 1974] can be related to the larger organization of the knowledge.

ORGANIZATION OF THE REPORT

The first part of this report describes the implementation of SEER in some detail. This provides a concrete setting for the discussion chapters that follow. Chapter six links the two sections with several examples of SEER in operation.

2

KNOWLEDGE STRUCTURES

THE FORM IN WHICH KNOWLEDGE IS REPRESENTED MAY SERVE TO PERMIT OR PREVENT, ENCOURAGE OR DETER ITS ACTIVE ROLE

For example, a mathematically oriented model encourages us to view a relationship like "handle next-to head" as a predicate to be tested on given handles and heads. I want to view this relationship as providing advice on how to find a head if I have found a handle, or vice versa. It also should suggest that object "X" might be a head, if I discover a handle is next-to X. Once we have an analysis of our knowledge which discovers and encourages its active roles, we can transfer it to a programming system which institutionalizes concrete mechanisms for implementing these roles.

We distinguish the particular knowledge (PK) structure from the general knowledge (GK) structure. PK consists of statements created during execution concerning the specific scene being analyzed. GK is information about possible visual phenomena--objects, properties, relations--that is available to help SEER process any given scene.

Both structures are control oriented. They are not merely descriptive, but represent means of acquiring identifications. The GK represents the variety of potential approaches; the PK consists of instantiations of some portion of these. It records the current state of processing and potential future directions in a manner that permits a sophisticated flow of control. We will present these structures more formally than we did in chapter one. A number of abbreviations are used. Abbreviations will sometimes be expanded, but after a while these expansions would prove as burdensome as the original abbreviations may seem cryptic. A table of abbreviations is provided in figure 2 - 1 for reference.

THE PK STRUCTURE IS A NETWORK OF NODES AND LINKS

Figure 2 - 2 illustrates the PK "around" the node D50, at the end of processing cycle 262 of the recognition task used for the example in chapter one. The arrows indicate links. The links have a direction; the node at the tail of an arrow is said to be "above" the node at the head. The figure

FIGURE 2 - 1

TABLE OF ABBREVIATIONS

HAMMER PARTS:
 H --> HAMMER
 HP1 --> HAMMER-PART-HANDLE
 HP2 --> HAMMER-PART-HEAD
 HP2P1 --> HAMMER-PART-HEAD-PART-FACE
LINK STRUCTURES:
 I --> INITIATION
 A --> ACTUATION
 E --> ESTABLISHMENT
 P ---> PRIORITY
STATUS:
 C --> CONJECTURE
 S --> SUCCESS
 F --> FAILURE
DATUM SUFFIXES:
 -C --> -COMPUTATION
 -N --> -NEEDS
 -W --> -WHOLE
 -Mi --> -METHODi
 -GT ---> -GENERATE-AND-TEST
 -G ---> -GENERATE
 -T ---> -TEST
 -REL ---> -RELATIONSHIPS
NUMBERED ITEMS
 Di ---> DATUM
 Qi --- QUANTUM
 Ci ---> CYCLE
 Li ---> GK LINK
 PKLi ---> PK LINK
 Ri ---> REGION
 DRi ---> DUMMY REGION
 NRi ---> NEW REGION (NON AFFINITY TREE REGION)
PREDICATES:
 O-BDRY ---> OUTER BOUNDARY
 L-AXIS ---> LENGTH AXIS
 W-AXIS ---> WIDTH AXIS
 SCH --> SMOOTHED-CONVEX-HULL
 VEX-STR-ANA ---> CONVEXITY-STRUCTURE-ANALYSIS
LINK TYPES:
 SE ---> SUCCESS-ESTABLISHMENT
 FE ---> FAILURE-ESTABLISHMENT
 PUI ---> PURSUE-INITIATION
 SI ---> SUCCESS-INITIATION
 FI ---> FAILURE-INITIATION
 SD ---> SUCCESS-DISESTABLISHMENT
 FD ---> FAILURE-DISESTABLISHMENT
 SP ---> SUCCESS-PRIORITY
 FP ---> FAILURE-PRIORITY
 A ---> ACTUATION

```
CYCLE TRACE:
 * ---> INITIATED
 INVES ---> INVESTIGATION LIST
 + ---> ADD TO
 - ---> DELETE FROM
 NIL ---> NONE
PK STATE:
 $$ ---> CURRENT EXPLORATION
 # ---> PREVIOUS EXPLORATION
GK SEGMENT:
 (I) ---> IDENTITY FUNCTION
```

FIGURE 2 - 2

```
                PK STATE
                     AT END OF CYCLE C262
                     ABOVE AND BELOW D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)
                     TO DEPTH 5.
                     SHOWING THE STATUS PROPERTY

      <- D263 (IS-A-HAMMER-METHOD1 DR4) SUCCESS
      <- D316 (HAS-A-HAMMER-PART-HEAD-METHOD1 DR4) CONJECTURE
     <- D264 (HAS-A-HAMMER-PART-HANDLE DR4) SUCCESS
    <- D262 (HAS-A-HAMMER-PART-HANDLE-METHOD2 DR4) SUCCESS
      <- D318 (RECOG DR4) SUCCESS
     <- D317 (IS-A-HAMMER DR4) SUCCESS
    <- D263 (IS-A-HAMMER-METHOD1 DR4) SUCCESS
   <- D261 (HAS-A-HAMMER-PART-HEAD DR4) SUCCESS
 <- D260 (IS-A-HAMMER-PART-HEAD R196) SUCCESS

D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196) SUCCESS

 -> D233 (HAS-A-HAMMER-PART-HEAD-PART-FACE R196) SUCCESS
  -> D234 (HAS-A-HAMMER-PART-HEAD-PART-FACE-METHOD1 R196) SUCCESS
   -> D235 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1 R196 R145) SUCCESS
    -> D237 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CE R196 R145) SUCCESS
     -> D240 (COVER-END R196 R145) SUCCESS
     -> D239 (CONVEX R145) SUCCESS
     -> D238 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CE1-METHOD2 R196 R145) SUCCESS
    -> D236 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CF R196 R145) CONJECTURE
 -> D44 (IS-A-HAMMER-PART-HEAD-WHOLE R196) SUCCESS
  -> D43 (IS-A-BAR R196) SUCCESS
   -> D28 (IS-A-BAR-METHOD1 R196) SUCCESS
    -> D41 (BAR-DIMENSIONS R196) SUCCESS
     -> D42 (BAR-DIMENSIONS-COMPUTATION R196) SUCCESS
     -> D39 (BAR-DIMENSIONS-NEEDS R196) SUCCESS
    -> D25 (HAS-BAR-SIDES R196) SUCCESS
     -> D23 (HAS-BAR-SIDES-METHOD1 R196) SUCCESS
```

shows first the nodes in the portion of the PK reached by moving along successive links upward from node D50. We also see the nodes below D50; for example D233 and D44 are the two nodes immediately below D50. D50 is a conjecture that method one for establishing that region 196 is a crude hammer head can be established for this scene.

PK pictures of this sort are restricted in several ways. The "depth" indicated reflects the number of successive links above or below the central node that we are willing to follow. Below, we only move successively downward, similarly above, only upward. If a node is encountered twice, the structure below it is not repeated. Some nodes would normally be deleted during processing, but may be retained in tracing modes to give a fuller picture of the PK structure created. Figures 2 - 3 and 2 - 4 illustrate the PK around the nodes D28 and D263.

THE NODES ARE CALLED DATUMS

A datum has a statement consisting of a predicate and arguments, which are constants. This is a statement about the scene being analyzed. The datum also has a set of properties that play roles in processing or data storage. The datum D28 has the statement "is-a-bar-m1 R196", "region 196 is a bar, method one".

A datum has a status. It is either a conjecture, (indicated by "C" in the figures) or is a success (S) or a failure (F). If its statement has been shown to hold in the scene, the datum is a success; if we have been unable to establish the statement, the datum is a failure; when the outcome is still in doubt, the datum is a conjecture. An "unexplored" conjecture is a suggestion. (Briefly, "exploration" implies we have begun an attempt to establish the datum.)

Other properties of datums, e.g. priority, will be discussed in appropriate sections. It is difficult to display a full picture of a multiply connected structure like this. Figure 2 - 5 presents the structure in the previous D50 figure, referencing links to other parts of the PK. E.g. there is a path downward from D50 through D44 to D43, but D43 also is below D45, which is not below D50. The PK is dynamic; it develops as the processing of a scene proceeds. Figure 2 - 6 shows how the PK below D15 changes over several cycles.

FIGURE 2 - 3

```
                    PK STATE
                    AT END OF CYCLE C262
                    ABOVE D28 (IS-A-BAR-M1 R196)
                    TO DEPTH 5.
                    SHOWING THE STATUS PROPERTY

<- D28 (IS-A-BAR-M1 R196) S
 <- D43 (IS-A-BAR R196) S
  <- D45 (IS-A-HP1-M1 R196) S
   <- D46 (IS-A-HP1 R196) S
    <- D47 (HAS-A-HP1 DR1) S
     <- D49 (IS-A-H-M1 DR1) F
     <- D48 (HAS-A-HP2-M1 DR1) F
  <- D44 (IS-A-HP2-W R196) S
   <- D50 (IS-A-HP2-M1 R196) S
    <- D260 (IS-A-HP2 R196) S
     <- D261 (HAS-A-HP2 DR4) S


                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D28 (IS-A-BAR-M1 R196)
                    TO DEPTH 5.
                    SHOWING THE STATUS PROPERTY

-> D28 (IS-A-BAR-M1 R196) S
 -> D41 (BAR-DIMENSIONS R196) S
  -> D42 (BAR-DIMENSIONS-C R196) S
  -> D39 (BAR-DIMENSIONS-N R196) S
   -> D32 (LENGTH R196) S
    -> D40 (LENGTH-C R196) S
    -> D31 (L-AXIS R196) S
     -> D29 (L-AXIS-M1 R196) S
   -> D38 (WIDTH R196) S
    -> D36 (WIDTH-M1 R196) S
     -> D37 (WIDTH-M1-C R196) S
     -> D35 (W-AXIS R196) S
 -> D25 (HAS-BAR-SIDES R196) S
  -> D23 (HAS-BAR-SIDES-M1 R196) S
   -> D24 (HAS-BAR-SIDES-M1-C R196) S
   -> D22 (HAS-BAR-SIDES-M1-N R196) S
    -> D12 (O-BDRY R196) S
     -> D14 (BDRY R196) S
     -> D13 (O-BDRY-C R196) S
    -> D21 (SCH-AS-DESC R196) S
     -> D6 (CONVEX R196) S
     -> D19 (SCH R196) S
```

FIGURE 2 - 4

```
                    PK STATE
                    AT END OF CYCLE C262
                    ABOVE D263 (IS-A-H-M1 DR4)
                    TO DEPTH 5.
                    SHOWING THE STATUS PROPERTY

<- D263 (IS-A-H-M1 DR4) S
 <- D317 (IS-A-H DR4) S
  <- D318 (RECOG DR4) S


                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D263 (IS-A-H-M1 DR4)
                    TO DEPTH 5.
                    SHOWING THE STATUS PROPERTY

-> D263 (IS-A-H-M1 DR4) S
 -> D264 (HAS-A-HP1 DR4) S
  -> D266 (HAS-A-HP1-M1 DR4) C
  -> D262 (HAS-A-HP1-M2 DR4) S
   -> D270 (HAS-A-HP1-M2-N DR4) S
    -> D31 (L-AXIS R196) S
     -> D29 (L-AXIS-M1 R196) S
    -> D38 (WIDTH R196) S
     -> D36 (WIDTH-M1 R196) S
   -> D269 (HAS-A-HP1-M2-TOMAS DR4) S
   -> D268 (HAS-A-HP1-M2-DUM DR4) S
    -> D271 (BAR-DIMENSIONS NR1) S
     -> D273 (BAR-DIMENSIONS-N NR1) S
     -> # D272 (BAR-DIMENSIONS-C NR1) # S
   -> $$ D267 (HAS-A-HP1-M2-V DR4) $$ S
   -> D261 (HAS-A-HP2 DR4) S
    -> D260 (IS-A-HP2 R196) S
     -> D50 (IS-A-HP2-M1 R196) S
  -> D265 (HAS-A-HP1-M3 DR4) C
 -> D261 (HAS-A-HP2 DR4) S
```

FIGURE 2 - 5

```
                PK STATE
                     AT END OF CYCLE C262
                     ABOVE AND BELOW D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)
                     TO DEPTH 5.
                     SHOWING THE LINKS PROPERTY

      <- D263 (IS-A-HAMMER-METHOD1 DR4) -> (D261)
      <- D316 (HAS-A-HAMMER-PART-HEAD-METHOD1 DR4)
     <- D264 (HAS-A-HAMMER-PART-HANDLE DR4) -> (D266 D265)
    <- D262 (HAS-A-HAMMER-PART-HANDLE-METHOD2 DR4) -> (D270 D269 D268 D267)
      <- D318 (RECOG DR4)
     <- D317 (IS-A-HAMMER DR4)
    <- D263 (IS-A-HAMMER-METHOD1 DR4) -> (D264)
  <- D261 (HAS-A-HAMMER-PART-HEAD DR4)
 <- D260 (IS-A-HAMMER-PART-HEAD R196)

D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)

 -> D233 (HAS-A-HAMMER-PART-HEAD-PART-FACE R196)
  -> D234 (HAS-A-HAMMER-PART-HEAD-PART-FACE-METHOD1 R196)
   -> D235 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1 R196 R145) <- (D259)
    -> D237 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CE R196 R145)
     -> D240 (COVER-END R196 R145)
     -> D239 (CONVEX R145) <- (D258)
     -> D238 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CE1-METHOD2 R196 R145)
    -> D236 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CF R196 R145)
 -> D44 (IS-A-HAMMER-PART-HEAD-WHOLE R196)
  -> D43 (IS-A-BAR R196) <- (D45)
   -> D28 (IS-A-BAR-METHOD1 R196)
    -> D41 (BAR-DIMENSIONS R196)
     -> D42 (BAR-DIMENSIONS-COMPUTATION R196)
     -> D39 (BAR-DIMENSIONS-NEEDS R196)
    -> D25 (HAS-BAR-SIDES R196) <- (D27 D26)
     -> D23 (HAS-BAR-SIDES-METHOD1 R196)
```

FIGURE 2 - 6

PK DEVELOPMENT

INVESTIGATING D15 (SCH-N R196)

AT START OF CYCLE C11

```
-> $$ D15 (SCH-N R196) $$ C
 -> D10 (AREA R196) S <- (D8)
```

AT START OF CYCLE C12

```
-> # D15 (SCH-N R196) # C
 -> $$ D9 (CONVEX-HULL R196) $$ C <- (D8)
 -> D10 (AREA R196) S <- (D8)
```

AT START OF CYCLE C13

```
-> D15 (SCH-N R196) C
 -> # D9 (CONVEX-HULL R196) # C <- (D8)
  -> $$ D17 (VEX-STR-ANA R196) $$ C
  -> D16 (CONVEX-HULL-C R196) C
 -> D10 (AREA R196) S <- (D8)
```

AT START OF CYCLE C14

```
-> D15 (SCH-N R196) C
 -> D9 (CONVEX-HULL R196) C <- (D8)
  -> # D17 (VEX-STR-ANA R196) # C
   -> D12 (O-BDRY R196) S <- (D10)
   -> $$ D18 (VEX-STR-ANA-C R196) $$ C
  -> D16 (CONVEX-HULL-C R196) C
 -> D10 (AREA R196) S <- (D8)
```

AT START OF CYCLE C15

```
-> D15 (SCH-N R196) C
 -> D9 (CONVEX-HULL R196) C <- (D8)
  -> D17 (VEX-STR-ANA R196) S
  -> $$ D16 (CONVEX-HULL-C R196) $$ C
 -> D10 (AREA R196) S <- (D8)
```

LINKS BETWEEN DATUMS CORRESPOND TO BOTH CONCEPTUAL RELATIONSHIPS AND SYSTEM OPERATIONS.

There are four types of links: establishing, priority, initiating and actuating. There are several priority factors which may function independently. We will refer to the establishing structure, priority structure, etc. Abbreviations will be used: E-link, P-structure, etc.

The purpose of establishing and priority structures is fairly obvious. The establishing structure reflects "definitions" or "programs" to establish datums (make them a success),more on this below. The priority structure indicates how datums support the interest of the control structure in other datums. Actuation and initiation are more directly motivated by SEER system functions. A datum D initiates a datum D' when it causes D' to be placed in the PK data base. (Links are entered into the PK.) PK initiation links are only a record of this action, unlike the other link types, which may be placed in the PK for possible future processing use. An actuation link from D to D' tells the monitor that one way to "follow up" an interest in D, is to look at D' in turn.

All these links have directions. A link "from D to D'" is "drawn" with an arrow from D pointing at D'. A link structure thus defines an order, within the network. As I indicated, one datum may be said to be above or below another, e.g. D is said to be above D' in the link structure. An E-link from D to D' indicates that D' can help establish the datum D above it. Below and above are transitive, so if B is below A and C is below B, C is below A. B is "immediately below" A if they are at the ends of the same link. For A above B in the E-structure, we say that "B establishes A", for A above B in the P-structure, we say that "B encourages A".

Two datums may be joined by links of more than one link type. When no confusion arises, I will refer to all the links between two nodes collectively, in the singular, as their link. (Actually the links are implemented "physically" in the program as a single link with multiple labels.) Generally, establishing, priority and actuating links will agree in direction, and we can merely say that B is below A. A link may have properties also, e.g. related to priorities.

We may distinguish link types further. The categories given reflect the effect which the link "transmits", e.g. one datum may affect the priority of another, or help establish another. SEER also needs to classify links in other ways. The "cause" of most effects is either the success or failure of

the datum below. Thus we have "success-priority" (SP) links, which indicate that the success of the datum below affects the priority of the datum above, and also "failure-priority" (FP) links. Either failure or success of a datum B can help establish A; we have SE and FE links. Similarly, success (or failure) of the datum below may cause initiation of the datum above (SI FI). In addition, however, a conjectured datum may initiate datums below it (PUI). There is a "disestablishing" structure as well as the establishing structure. (See chapter three.)

Figure 2 - 7 presents some of the PK structure around D28 in more detail. It includes examples of many of the link types. (Since E links always imply P-links, we do not indicate those P-links separately.) A variety of properties are attached to the nodes and links; their purposes will become clearer as we proceed.

WHEN SEER ANALYZES A SCENE, THE PK REPRESENTS THE CURRENT STATE OF PROCESSING, AS WELL AS THE CURRENT STATE OF KNOWLEDGE ABOUT THE SCENE

A datum in the actuation structure with nothing above it in that structure, will be called the subject of an investigation. The nodes below it in the A-structure form the investigation. The nodes at the bottom of the investigation are called frontier nodes.

Various "paths" through the PK structure are of interest. A path is a connected set of nodes in a link structure, with associated information. Normally an investigation would be used as the basis set of nodes for a path. These paths provide interesting "traces" of the processing flow.

A path in the actuation structure numbers the set of nodes as to the order in which they were explored. A path in the I-structure indicates the order in which nodes were created. A path in the E-structure numbers the nodes to indicate the order in which they were established, and indicates which nodes helped establish which. These paths also include information on the cycle during which the node was initiated, explored, or established.

The numbering in the actuation or establishment paths may be merely consecutively ordinal within the path set. We may also number each execution or establishment during the overall processing of a scene. These execution or establishment numbers can be used in a path to reflect not only relative ordering, but the place of path elements in the larger processing run. (this will be the

FIGURE 2 - 7

```
                    PK SEGMENT
                    ABOVE D28

D28 (IS-A-BAR-M1 R196)
     STATUS = S
     DIFF = 17.42 LIKE = 0.5 INT = 1.
     AND = 0.
     LINKS:
     ---> (D41 D25)
     <--- PKL50
          TYPES: (SE SI PUI A)
          FIRED: (SE)
          FROM:
          D43 (IS-A-BAR R196)
               STATUS = S
               DIFF = 21.2 LIKE = 0.5 INT = 1.
               OR = 1.
               LINKS:
               ---> NIL

                    BELOW D28

D28 (IS-A-BAR-M1 R196)
     STATUS = S
     DIFF = 17.42 LIKE = 0.5 INT = 1.
     AND = 0.
     LINKS:
     <--- (D43)
     ---> PKL49
          TYPES: (PUI SI SE A)
          FIRED: (SE)
          TO:
          D41 (BAR-DIMENSIONS R196)
               EXPLORED
               STATUS = S
               DIFF = 1.0 LIKE = 0.5 INT = 1.
               SER
               AND = 0.
               LINKS:
               <--- NIL
     ---> PKL31
          TYPES: (PUI SI SE A)
          FIRED: (SE)
          TO:
          D25 (HAS-BAR-SIDES R196)
               STATUS = S
               IS = (((180. 630.) (160. 550.)) ((220. 570.) (240. 650.)))
               DIFF = 3.78 LIKE = 0.5 INT = 1.
               OR = 1.
               LINKS:
               <--- (D27 D26)
```

numbering scheme used in the figures presented.)

A recognition path consists of only those nodes below an established datum. We may further limit ourselves to those nodes actually used to establish the subject node, or those nodes explored or established while investigating the subject node (we can obtain results in one investigation while working on another). A recognition path may be formed in the various structures as described above.

Figures 2 - 8 , 2 - 9 and 2 - 10 are recognition paths in the I, A and E structures respectively. In the latter figure, the number 279 after datum D263 indicates that this was the 279th datum initiated. C262 is the cycle during which it was established. D263, is-a-hammer-method1 DR4, was the datum being investigated when D263 was established. D267 was the datum explored to start off cycle C262. A "NIL" indicates that the datum was never established. The initiation and actuation paths present similar data on the creation and exploration of datums. Figure 2 - 11 is an abbreviated actuation path, showing just the order of exploration. The flow of processing is difficult to present in a single static figure, but these provide the data, and a little work will will allow one to follow the flow from datum to datum, branch to branch and investigation to investigation, to determine when and why datums were initiated, explored and established.

At the beginning of each cycle we can take a "snapshot" of the state of the current investigation. Comparing successive snapshots gives us another picture of the dynamic development of the PK. In more detail we can view the activity during the cycle as it alters the PK through "cycle traces" produced during processing. We will see these in the next chapter. The format of the cycle trace is intended to reflect the structure of the affected PK. The state diagrams and cycle traces are produced by SEER as a cycle is processed.

THE GK IS A NETWORK AS WELL

The similar structure of the GK and the PK facilitates interaction between the original GK stored knowledge and the dynamically acquired results and processing steps represented in the PK. (The uniform and modular representation of knowledge also facilitiates sharing of PK items.)

FIGURE 2 - 8

```
PK STATE
AT END OF CYCLE C262
BELOW D263 (IS-A-H-M1 DR4)
TO DEPTH 3.
SHOWING THE I PATH

-> D263 (IS-A-H-M1 DR4)
   263. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
 -> D264 (HAS-A-HP1 DR4)
    264. C221 (D263 (IS-A-H-M1 DR4)) (D263 (IS-A-H-M1 DR4))
  -> D266 (HAS-A-HP1-M1 DR4)
     266. C222 (D263 (IS-A-H-M1 DR4)) (D264 (HAS-A-HP1 DR4))
  -> D262 (HAS-A-HP1-M2 DR4)
     262. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
   -> D270 (HAS-A-HP1-M2-N DR4)
      270. C223 (D263 (IS-A-H-M1 DR4)) (D262 (HAS-A-HP1-M2 DR4))
   -> D269 (HAS-A-HP1-M2-TOMAS DR4)
      269. C223 (D263 (IS-A-H-M1 DR4)) (D262 (HAS-A-HP1-M2 DR4))
   -> D268 (HAS-A-HP1-M2-DUM DR4)
      268. C223 (D263 (IS-A-H-M1 DR4)) (D262 (HAS-A-HP1-M2 DR4))
   -> $$ D267 (HAS-A-HP1-M2-V DR4) $$
      267. C223 (D263 (IS-A-H-M1 DR4)) (D262 (HAS-A-HP1-M2 DR4))
   -> D261 (HAS-A-HP2 DR4)
      261. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
  -> D265 (HAS-A-HP1-M3 DR4)
     265. C222 (D263 (IS-A-H-M1 DR4)) (D264 (HAS-A-HP1 DR4))
 -> D261 (HAS-A-HP2 DR4)
    261. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
  -> D260 (IS-A-HP2 R196)
     260. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
   -> D50 (IS-A-HP2-M1 R196)
      50. C33 (D49 (IS-A-H-M1 DR1)) (D42 (BAR-DIMENSIONS-C R196))
```

FIGURE 2 - 9

```
PK STATE
AT END OF CYCLE C262
BELOW D263 (IS-A-H-M1 DR4)
TO DEPTH 3.
SHOWING THE A PATH

-> D263 (IS-A-H-M1 DR4)
   219. C221 (D263 (IS-A-H-M1 DR4)) (D263 (IS-A-H-M1 DR4))
 -> D264 (HAS-A-HP1 DR4)
    220. C222 (D263 (IS-A-H-M1 DR4)) (D264 (HAS-A-HP1 DR4))
  -> D266 (HAS-A-HP1-M1 DR4)
     NIL
  -> D262 (HAS-A-HP1-M2 DR4)
     221. C223 (D263 (IS-A-H-M1 DR4)) (D262 (HAS-A-HP1-M2 DR4))
   -> D270 (HAS-A-HP1-M2-N DR4)
      222. C224 (D263 (IS-A-H-M1 DR4)) (D270 (HAS-A-HP1-M2-N DR4))
   -> D269 (HAS-A-HP1-M2-TOMAS DR4)
      223. C225 (D263 (IS-A-H-M1 DR4)) (D269 (HAS-A-HP1-M2-TOMAS DR4))
   -> D268 (HAS-A-HP1-M2-DUM DR4)
      224. C226 (D263 (IS-A-H-M1 DR4)) (D268 (HAS-A-HP1-M2-DUM DR4))
   -> $$ D267 (HAS-A-HP1-M2-V DR4) $$
      260. C262 (D263 (IS-A-H-M1 DR4)) (D267 (HAS-A-HP1-M2-V DR4))
   -> D261 (HAS-A-HP2 DR4)
      NIL
  -> D265 (HAS-A-HP1-M3 DR4)
     NIL
-> D261 (HAS-A-HP2 DR4)
   NIL
 -> D260 (IS-A-HP2 R196)
    NIL
  -> D50 (IS-A-HP2-M1 R196)
     196. C198 (D50 (IS-A-HP2-M1 R196)) (D50 (IS-A-HP2-M1 R196))
```

FIGURE 2 - 10

```
PK STATE
AT END OF CYCLE C262
BELOW D263 (IS-A-H-M1 DR4)
TO DEPTH 3.
SHOWING THE E PATH

-> D263 (IS-A-H-M1 DR4)
   279. C262 (D263 (IS-A-H-M1 DR4)) (D267 (HAS-A-HP1-M2-V DR4))
 -> D264 (HAS-A-HP1 DR4)
    278. C262 (D263 (IS-A-H-M1 DR4)) (D267 (HAS-A-HP1-M2-V DR4))
  -> D266 (HAS-A-HP1-M1 DR4)
     NIL
  -> D262 (HAS-A-HP1-M2 DR4)
     277. C262 (D263 (IS-A-H-M1 DR4)) (D267 (HAS-A-HP1-M2-V DR4))
   -> D270 (HAS-A-HP1-M2-N DR4)
      231. C224 (D263 (IS-A-H-M1 DR4)) (D270 (HAS-A-HP1-M2-N DR4))
   -> D269 (HAS-A-HP1-M2-TOMAS DR4)
      232. C225 (D263 (IS-A-H-M1 DR4)) (D269 (HAS-A-HP1-M2-TOMAS DR4))
   -> D268 (HAS-A-HP1-M2-DUM DR4)
      275. C261 (D263 (IS-A-H-M1 DR4)) (D272 (BAR-DIMENSIONS-C NR1))
   -> $$ D267 (HAS-A-HP1-M2-V DR4) $$
      276. C262 (D263 (IS-A-H-M1 DR4)) (D267 (HAS-A-HP1-M2-V DR4))
   -> D261 (HAS-A-HP2 DR4)
      230. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
  -> D265 (HAS-A-HP1-M3 DR4)
     NIL
 -> D261 (HAS-A-HP2 DR4)
    230. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
  -> D260 (IS-A-HP2 R196)
     229. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
   -> D50 (IS-A-HP2-M1 R196)
      228. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))
```

FIGURE 2 - 11

```
                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D263 (IS-A-H-M1 DR4)
                    TO DEPTH 7.
                    SHOWING THE A PATH

-> D263 (IS-A-H-M1 DR4) 219.
 -> D264 (HAS-A-HP1 DR4) 220.
  -> D266 (HAS-A-HP1-M1 DR4) NIL
  -> D262 (HAS-A-HP1-M2 DR4) 221.
   -> D270 (HAS-A-HP1-M2-N DR4) 222.
    -> D31 (L-AXIS R196) NIL
     -> D29 (L-AXIS-M1 R196) 22.
      -> D30 (L-AXIS-M1-C R196) 23.
      -> D26 (L-AXIS-M1-N R196) 21.
       -> D12 (O-BDRY R196) 5.
       -> D25 (HAS-BAR-SIDES R196) NIL
    -> D38 (WIDTH R196) NIL
     -> D36 (WIDTH-M1 R196) 26.
      -> D37 (WIDTH-M1-C R196) 27.
      -> D35 (W-AXIS R196) NIL
       -> D33 (W-AXIS-M1 R196) 24.
   -> D269 (HAS-A-HP1-M2-TOMAS DR4) 223.
   -> D268 (HAS-A-HP1-M2-DUM DR4) 224.
    -> D271 (BAR-DIMENSIONS NR1) 225.
     -> D273 (BAR-DIMENSIONS-N NR1) 226.
      -> D275 (LENGTH NR1) 227.
       -> D277 (L-AXIS NR1) 228.
       -> D276 (LENGTH-C NR1) 256.
      -> D274 (WIDTH NR1) 249.
       -> D301 (WIDTH-M1 NR1) 250.
     -> # D272 (BAR-DIMENSIONS-C NR1) # 259.
   -> $$ D267 (HAS-A-HP1-M2-V DR4) $$ 260.
   -> D261 (HAS-A-HP2 DR4) NIL
    -> D260 (IS-A-HP2 R196) NIL
     -> D50 (IS-A-HP2-M1 R196) 196.
      -> D233 (HAS-A-HP2P1 R196) 197.
       -> D234 (HAS-A-HP2P1-M1 R196) 198.
      -> D44 (IS-A-HP2-W R196) NIL
       -> D43 (IS-A-BAR R196) NIL
 -> D265 (HAS-A-HP1-M3 DR4) NIL
-> D261 (HAS-A-HP2 DR4) NIL
```

GK NODES ARE CALLED QUANTUMS

Quantums also have statements consisting of predicates, and arguments which are variables. These statements, when instantiated (replaced with constant arguments) become statements suitable for PK datums. When the statement of datum D is an instantiation of the statement of quantum Q, D is said to be an instantiation of Q, and Q and D are said to correspond in the GK and PK structures.

Quantums have properties, useful in initiating and operating on corresponding datums. Some of these will be discussed here and some in chapter three. Figures 2 - 12 , 2 - 13 and 2 - 14 illustrate the GK near the quantums Q85, Q12 and Q78, corresponding to the datums illustrated above, and show links to other parts of the GK. Figure 2 - 15 gives a diagrammatic view of the GK near Q85.

A quantum in the actuation structure is labelled (i.e. has the property) serial or parallel. Corresponding datums are also serial or parallel. This property is used in exploration. At a serial node, the subnodes or branches are ordered (in the order in which they must be established) and we can refer to e.g. the "first node below". Other properties of quantums include initial values for priority factors, used when setting up new datums.

Some predicates will always be found to hold for any arguments. These are called certain; others are called uncertain. For example, a region will always have an area, but not always be convex. Quantums, and datums, will be called uncertain, if their predicates are uncertain. (Technically we are concerned with whether we can establish a property, not its abstract truth value. Thus if areas were difficult to determine, the area might in fact not be certain.)

LINKS BETWEEN QUANTUMS

These links are of the types described for the PK. Initiation links here represent potential initiations. An I-link to Q' indicates that an instantiation of Q could, at an appropriate time, initiate an instantiation of Q'. Establishment, priority and actuating links represent basically the same concepts as they do in the PK structure. However, here the relationships are between abstract uninstantiated statements. For these relationships to really make sense, we need to keep track of correspondences between the variables in the statements. For example, we need to distinguish the two alternatives in figure 2 - 16 .

FIGURE 2 - 12

```
                    GK SEGMENT
                    ABOVE AND BELOW Q85 IS-A-HAMMER-PART-HEAD-METHOD1
                    TO DEPTH OF 5.
                    SHOWING PROPERTY LINKS

      <- Q77 HAS-A-HAMMER-PART-HEAD-METHOD1 -> (Q98)
      <- Q78 IS-A-HAMMER-METHOD1 -> (Q76)
     <- Q79 HAS-A-HAMMER-PART-HANDLE -> (Q114 Q69)
    <- Q110 HAS-A-HAMMER-PART-HANDLE-METHOD2 -> (Q113 Q117 Q112 Q111)
      <- Q108 RECOGNITION
     <- Q80 IS-A-HAMMER
    <- Q78 IS-A-HAMMER-METHOD1 -> (Q79)
   <- Q76 HAS-A-HAMMER-PART-HEAD -> (Q77)
  <- Q86 IS-A-HAMMER-PART-HEAD

Q85 IS-A-HAMMER-PART-HEAD-METHOD1

 -> Q84 HAS-A-HAMMER-PART-HEAD-PART-FACE
  -> Q3 HAS-A-HAMMER-PART-HEAD-PART-FACE-METHOD1
   -> Q62 HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1 <- (Q1)
    -> Q63 HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CF
     -> Q60 HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CFT
     -> Q61 HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CFG
    -> Q59 HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CE
     -> Q119 HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1-CE1-METHOD2
     -> Q35 CONVEX <- (Q34)
     -> Q6 COVER-END
 -> Q83 IS-A-HAMMER-PART-HEAD-WHOLE
  -> Q14 IS-A-BAR <- (Q74)
   -> Q12 IS-A-BAR-METHOD1
    -> Q10 BAR-DIMENSIONS
     -> Q9 BAR-DIMENSIONS-COMPUTATION
     -> Q11 BAR-DIMENSIONS-NEEDS
    -> Q13 HAS-BAR-SIDES <- (Q15 Q23)
     -> Q38 HAS-BAR-SIDES-METHOD1
```

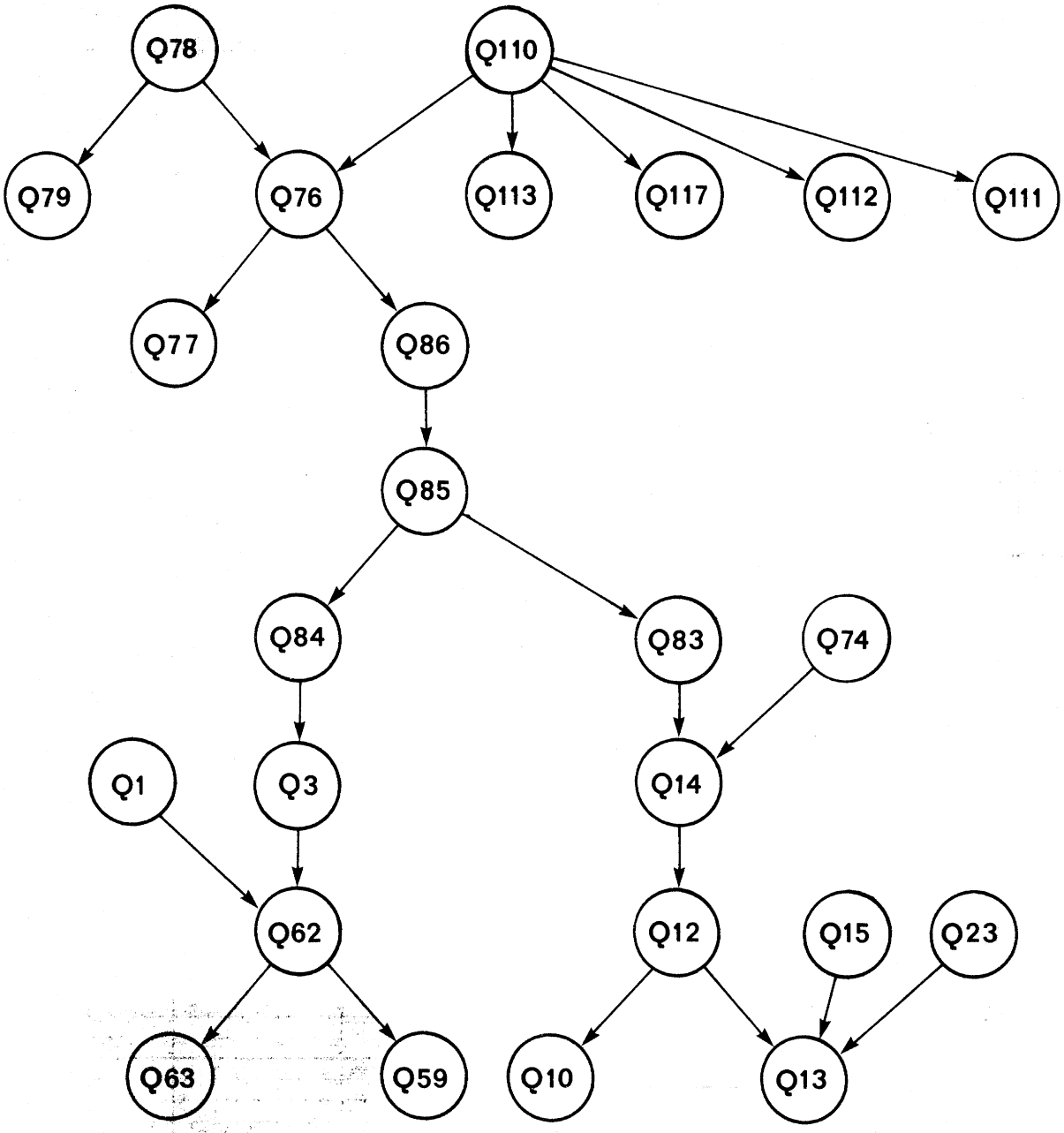
FIGURE 2 - 13

```
                    GK SEGMENT
                    ABOVE Q12 IS-A-BAR-M1
                    TO DEPTH OF 5.
                    SHOWING PROPERTY LINKS

<- Q12 IS-A-BAR-M1 -> (Q10 Q13)
 <- Q14 IS-A-BAR
  <- Q74 IS-A-HP1-M1
   <- Q75 IS-A-HP1
    <- Q79 HAS-A-HP1 -> (Q114 Q110 Q69)
     <- Q78 IS-A-H-M1 -> (Q76)
     <- Q77 HAS-A-HP2-M1 -> (Q98)
  <- Q83 IS-A-HP2-W
   <- Q85 IS-A-HP2-M1 -> (Q84)
    <- Q86 IS-A-HP2
     <- Q76 HAS-A-HP2 -> (Q77)


                    GK SEGMENT
                    BELOW Q12 IS-A-BAR-M1
                    TO DEPTH OF 5.
                    SHOWING PROPERTY LINKS

-> Q12 IS-A-BAR-M1 <- (Q14)
 -> Q10 BAR-DIMENSIONS
  -> Q9 BAR-DIMENSIONS-C
  -> Q11 BAR-DIMENSIONS-N
   -> Q22 WIDTH
    -> Q21 WIDTH-M1
     -> Q20 WIDTH-M1-C
     -> Q19 W-AXIS
   -> Q27 LENGTH
    -> Q28 LENGTH-C
    -> Q16 L-AXIS <- (Q15)
     -> Q26 L-AXIS-M1
 -> Q13 HAS-BAR-SIDES <- (Q15 Q23)
  -> Q38 HAS-BAR-SIDES-M1
   -> Q37 HAS-BAR-SIDES-M1-C
   -> Q36 HAS-BAR-SIDES-M1-N
    -> Q24 O-BDRY
     -> Q101 O-BDRY-C
     -> Q100 BDRY
    -> Q34 SCH-AS-DESC
     -> Q35 CONVEX
     -> Q33 SCH
```

FIGURE 2 - 14

```
                    GK SEGMENT
                    ABOVE Q78 IS-A-H-M1
                    TO DEPTH OF 2.
                    SHOWING PROPERTY LINKS

<- Q78 IS-A-H-M1 -> (Q76 Q79)
 <- Q80 IS-A-H
  <- Q108 RECOG


                    GK SEGMENT
                    BELOW Q78 IS-A-H-M1
                    TO DEPTH OF 5.
                    SHOWING PROPERTY LINKS

-> Q78 IS-A-H-M1 <- (Q80)
 -> Q76 HAS-A-HP2 <- (Q110)
  -> Q77 HAS-A-HP2-M1
   -> Q98 HAS-A-HP2-M1-GT
    -> Q97 HAS-A-HP2-M1-T
     -> Q82 HAS-THE-HP2
    -> Q96 HAS-A-HP2-M1-G
     -> Q70 HAS-A-HP2-M1-GC
     -> Q73 HAS-A-HP2-M1-GN
   -> Q79 HAS-A-HP1 <- (Q78)
    -> Q114 HAS-A-HP1-M3
     -> Q116 HAS-A-HP1-M3-C
     -> Q115 HAS-A-HP1-M3-N2
     -> Q110 HAS-A-HP1-M2 <- (Q79)
    -> Q110 HAS-A-HP1-M2
     -> Q113 HAS-A-HP1-M2-V
     -> Q117 HAS-A-HP1-M2-DUM
     -> Q112 HAS-A-HP1-M2-TOMAS
     -> Q111 HAS-A-HP1-M2-N
     -> Q76 HAS-A-HP2 <- (Q78)
    -> Q69 HAS-A-HP1-M1
     -> Q68 HAS-A-HP1-M1-T
     -> Q67 HAS-A-HP1-M1-G
-> Q79 HAS-A-HP1 <- (Q77)
```

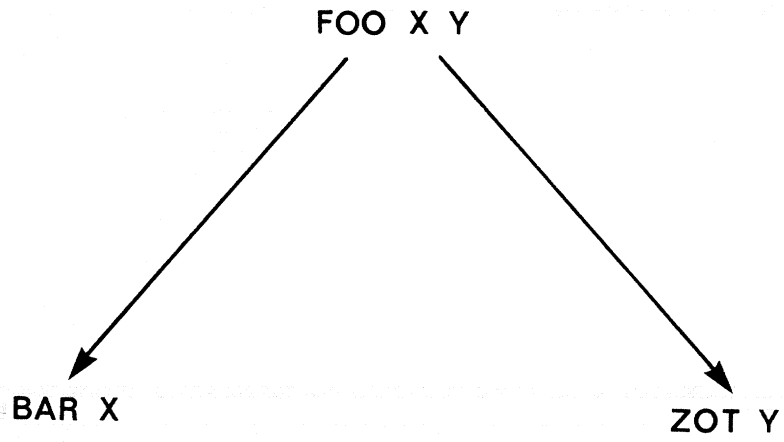
FIGURE 2-15

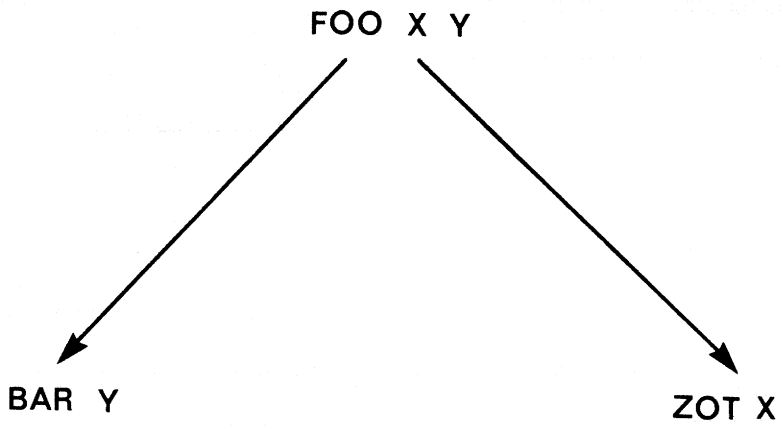


GK AROUND Q85

FIGURE 2-16

GK ARGUMENTS





The E, P, A links also represent "potential" PK links, between appropriate instantiations of quantums into datums. The GK around Q12 is presented in more detail in figure 2 - 17 . The establishment structure bears some further discussion. This is basically an AND/OR network. Each quantum is of type AND or type OR, except for terminal nodes. A terminal node in the E-structure is one without any node below it in the E-structure. It is called a primitive quantum. These have associated with them execution modules.

This structure has an obvious interpretation in the PK. An AND datum, D, for example, corresponding to an AND quantum Q, is established when a set of datums below it in the E-structure is established. These datums are derived from the quantums below Q, employing E-relevant statements. The process of establishing datums is described further in chapter three.

THE PK INSTANTIATES PIECES OF THE GK

GK represents all the ways to describe or acquire scene features. It contains a variety of potential programming sequences. The PK represents the dynamic state of an investigation and the scene description. It contains processes, specific attempts to determine features in a scene, and in particular the successfully established descriptions, or winning programs. One way in which description merges with recognition, and process with knowledge, in SEER, is structurally in the PK. The description of an object, in any specific instance, and its recognition, are the trace of a successful processing path in the PK.

During processing quantums are instantiated as datums and quantum links entered as datum links, as the PK changes dynamically. How this is accomplished is the subject of chapter three. I can only give here some definitions that will prove useful to that discussion.

Given a link from Q to Q' we can ask whether Q' "knows about" the link. Q knows about the link if the link is available among its properties. A link from Q to Q' may be known about by either Q or Q' or both. If Q knows about a link to Q', we say that Q knows about Q'. The quantums linked to Q, which Q knows about, are termed relevant to Q. Links have directions as in the PK; thus we may speak of the relevant quantums above Q.

Given a datum, D, we may obtain a set of statements relevant to D as follows:

FIGURE 2 - 17

```
                    GK SEGMENT
                    BELOW Q12 IS-A-BAR-M1

Q12 IS-A-BAR-M1
     DIFF = 21.2 LIKE = 0.5 INT = 1. BBB = Q35
     AND
     LINKS:
     <--- (Q14)
     ---> L8
          INS: (I)
          TYPES: (PUI SI SE A)
          ENC: 0.5
          TO:
          Q10 BAR-DIMENSIONS
               DIFF = 17.42 LIKE = 0.5 INT = 1. BBB = Q35
               SER
               AND
               LINKS:
               <--- NIL
     ---> L7
          INS: (I)
          TYPES: (PUI SI SE A)
          ENC: 0.5
          TO:
          Q13 HAS-BAR-SIDES
               IS
               DIFF = 3.78 LIKE = 0.5 INT = 1. BBB = Q35
               OR
               LINKS:
               <--- (Q15 Q23)
```

First, we obtain the quantum, Q, corresponding to D. Then we get the quantums relevant to Q. Finally, the statements of these quantums are instantiated to obtain the statements relevant to D. See the diagram in figure 2 - 18 .

The links in GK have provided us with the relevant quantums, and thus the predicates for the relevant statements. The arguments are instantiated by applying functions attached to these links, called instantiation or generation functions.

These functions take as arguments the arguments of the datum D. They may also operate on other information in the PK structure. A function attached to the link from Q to Q' can return as values instantiations of the arguments of the Q' statement. A function can return single or multiple sets of values, and more than one function may attach to a link. These yield alternative instantiations.

Note also that a function is associated with a direction: it tells us how to get an instantiation of Q' given one of Q, or vice versa. If both ends Q and Q' know about the link, different functions may be required to go in the different directions. Some links may not have an associated function in one direction. In this case the link is not known about in that direction and no relevant statements result.

In the simplest case an instantiation function may be the identity function. See, for an example, figure 2 - 19 . A slightly more complex case is illustrated in figures 2 - 20 and 2 - 21 . In the first figure we see that Q45 links to Q16. However, to judge perpendicularity we really need the length axis of both regions involved. Thus we see that the link has two instantiation functions. Thus in the second figure we see that "perp R196 R140" pursues "l-axis R196" and "l-axis R196".

More complex functions involve generation of new arguments, not found among the arguments of D. Generation of parts for suggested wholes, for example. Here is where several alternative conjectures may be required. The generation problem is a very basic one; it is discussed further in later chapters. We can distinguish I-relevant statements, P-relevant statements, etc., by employing those types of links in the GK. This shuttling between PK and GK, involved in generating statements relevant to a given datum, is active knowledge operating in SEER. The relevant statements are used to extend and operate on the PK during processing.

Subsets of the GK may be chosen and potential processing paths in the various structures traced. Many potential recognition paths may be picked out of the GK. They are formed by

FIGURE 2-18

INSTANTIATION

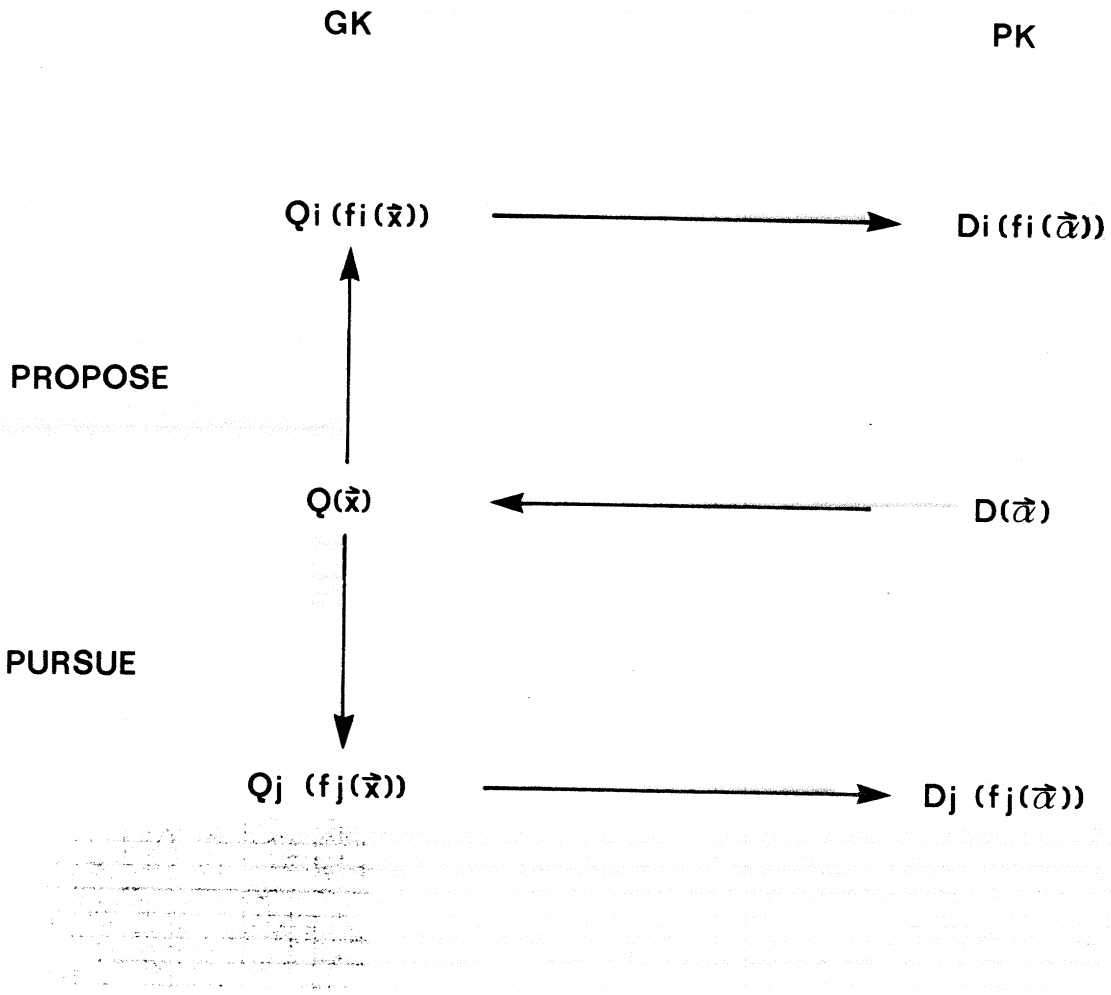


FIGURE 2-19

INSTANTIATION EXAMPLE

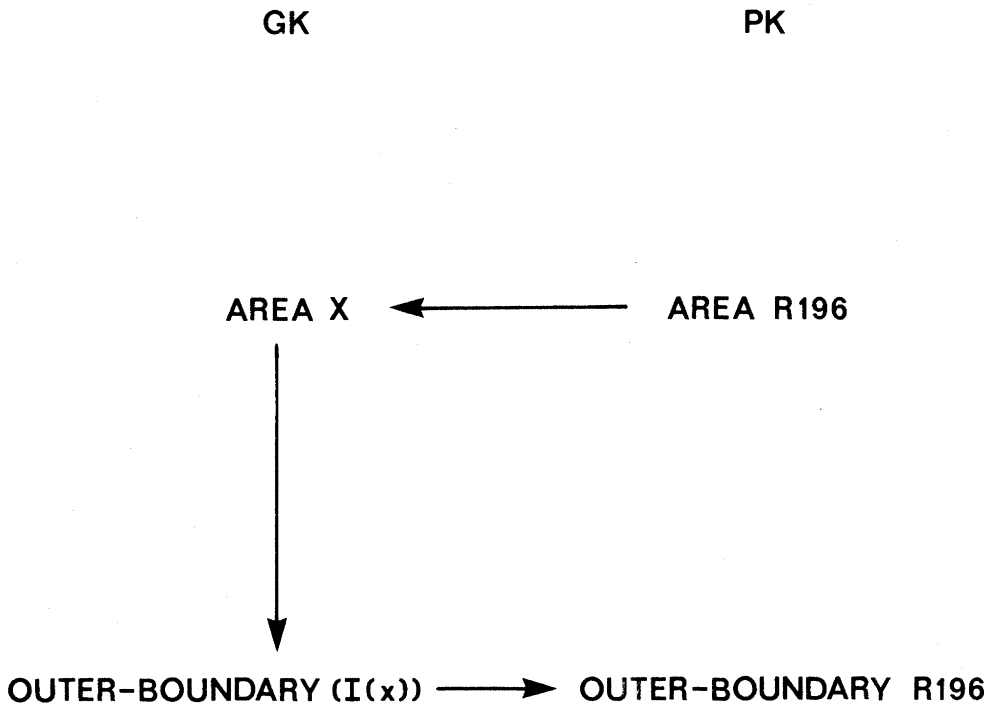


FIGURE 2 - 20

```
                    GK SEGMENT
                    BELOW Q45 PERP-N

Q45 PERP-N
     DIFF = 9.63 LIKE = 0.5 INT = 1. BBB = Q35
     AND
     LINKS:
     <--- (Q44)
     ---> L111
          INS: (EXG-CAR EXG-CDR)
          TYPES: (SE PUI A)
          ENC: 0.5
          TO:
          Q16 L-AXIS
               IS
               DIFF = 4.81 LIKE = 0.5 INT = 1. BBB = Q35
               OR
               LINKS:
               <--- (Q15 Q27)
```

FIGURE 2 - 21

```
CYCLE C53
INVESTIGATING D49 (IS-A-H-M1 DR1)
AT: D71 (PERP-N R196 R140)


STATE OF INVESTIGATION
OF D49 (IS-A-H-M1 DR1)
AT START OF CYCLE C53

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> D53 (HAS-A-HP2-M1-T DR1) C
     -> D59 (HAS-THE-HP2 DR1 R140) C
      -> D60 (HAS-THE-HP2-M1 DR1 R140) C
       -> D62 (H-REL-P2 DR1 R140) C
        -> D63 (H-REL-P1-P2 R196 R140) C
         -> D64 (T-JOINED R196 R140) C
          -> D66 (TOUCH R196 R140) C
           -> D67 (TOUCH-M1 R196 R140) C
            -> D69 (TOUCH-M1-N R196 R140) C
             -> # D65 (PERP R196 R140) # C <- (D64)
              -> $$ D71 (PERP-N R196 R140) $$ C
              -> D70 (PERP-C R196 R140) C
            -> D68 (TOUCH-M1-C R196 R140) C
          -> # D65 (PERP R196 R140) # C <- (D69)
       -> D61 (IS-A-HP2 R140) C
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D71
---> * D72 (L-AXIS R140)
     ENTERED: (SE PUI A)
     FIRED:
---> D31 (L-AXIS R196)
     ENTERED: (SE PUI A)
     FIRED: SE
     DIFF -> 4.81
```

"simulating" the operation of SEER assuming various results of computations on the scene. They correspond to potential PK processing sequences and recognitions.

HIERARCHICAL RELEVANCE ANALYSIS

We are interested in what I somewhat grandly term a "hierarchical relevance analysis". We must break up knowledge into pieces in a manner that reflects various useful forms of interaction. This involves determining appropriate units and identifying the types of relevance relationships. These relationships organize the units into hierarchies.

Note the range of relationships covered. Necessary and sufficient defining conditions are not the only considerations. Probability, causality, context is represented. Contributions to difficulty are included. Redundancy and alternatives are encouraged. Redundancy is a major feature of visual input. We must be able to avoid irrelevancies, and avoid computing all properties when various subsets will suffice for identification. On the other hand we should be able to choose among alternatives to find the easiest approach to a given scene. In SEER, various "methods" are provided, and provision for advising which ones are appropriate. It is important to find the right "atomic" relationships. These relationships themselves interact. For example, an E-relation implies a P-relation. However, if we recognize that priority is an independent relationship, we can implement it apart from the establishment process.

We are encouraged to analyze "descriptive knowledge" in terms of active questions like:

how would it help do x, if we know y?
if we know y, what x would it help?
if we are having trouble with x, what would it help to know?
is there a special case if we could demonstrate the right context?
an easy trick if we have y?
a likely distortion if y holds?
if we have x, does that limit were to look for y?

The problem of finding an orientation axis is difficult in general. However, for objects with a pair of long parallel straight sides, like a hammer handle, we can simply align the axis in the direction of these "bar sides" (see figure 2 - 22). If we know a region is symmetric this can help us find its boundary shape.

Basically we want to determine the relationships that will direct us in exploiting results and

FIGURE 2 - 22

```
GK SEGMENT
BELOW Q26 L-AXIS-M1
TO DEPTH OF 2.
SHOWING PROPERTY LINKS

-> Q26 L-AXIS-M1 <- (Q16)
 -> Q25 L-AXIS-M1-C
 -> Q23 L-AXIS-M1-N
  -> Q24 O-BDRY
  -> Q13 HAS-BAR-SIDES <- (Q12 Q15)
```

exploring conjectures effectively. Then we need to put these into the initiation, establishment, priority and actuation links in the GK. SEER provides mechanisms for implementing such relevance observations, as "suggestions" and "advice" during processing. The heuristic hierarchical relevance gets formalized as GK links some of which transfer to PK links. Links get fired or consulted during processing as we shall see in the next chapter.

The descriptive motivation for the GK is functional. A hammer requires a piece to grasp, another to hit with; this in turn requires a flat face to strike with, etc. These functional considerations motivate a description that is adaptive to the variety of hammer forms. Functional criteria provide clues to the essential properties of our objects.

We do want to use details when they provide an easier route to recognition in a given context. When the hammer head is heavily obscured or distorted we may need a careful evaluation of the handle shape to suggest a hammer. For such details more effective purely descriptive schemes will be needed. Gross distinctions, however, suffice for a great many recognitions. And where they do not context may assist. The shape distinctions between a screwdriver and a narrow chisel are subtle at best. However, a background of wood shavings should justify a good working judgement. For many distinctions, however, e.g. between hammer and screwdriver, fine descriptive distinctions are not needed. The problem lies in acquiring even a simple description from realistic data, or extracting the basic identifying features from a variety of images of the object.

3

CONTROL STRUCTURES

PROCESSING IS ORGANIZED INTO CYCLES WHICH BALANCE LOCAL CONTROL OF EXPLORATION AND EXPLOITATION WITH GLOBAL EVALUATION BY THE MONITOR AND PRIORITY SYSTEM

Processing a scene involves recognizing the objects it contains. SEER starts by noticing some "interesting" regions. Initial conjectures are made about those regions, to form an initial PK state. From then on SEER uses active knowledge to guide its processing. Processing involves suggesting more conjectures, choosing which ones to investigate and in what order. We proceed until the objects in the scene are recognized, or a sought object has been found.

A datum knows how to make suggestions that will help to establish itself. Once established, it knows how to exploit this new knowledge. It does so by making further suggestions and affecting the priority and establishment structure of the PK. This exploitation provides a specific mechanism for influencing the flow of processing.

SEER operates on the PK. The PK contains a network of datums. It includes a "suggestion pool". Processing proceeds in a sequence of cycles. A cycle changes the state of PK. To begin a cycle the monitor chooses the next datum to execute. It does this by first choosing the most promising investigation and then the most promising available suggestion to further the investigation. "Promising" is defined by the priority system.

A cycle consists of an exploration subcycle, which may be followed by one or more exploitation (or exorcising) subcycles. The chosen suggestion is explored. There are two types of exploration subcycles. If the explored datum corresponds to a primitive quantum, the datum is executed and succeeds or fails. Otherwise, the datum pursues further datums. If enough of these are already present and established or failed, the explored datum may still succeed, or fail, during exploration.

If the datum succeeds it is exploited. This initial exploitiation subcycle may be followed by others. The exploitation of the original datum may establish others above it. These in turn will be

exploited. This process may recurse some distance up the PK structure. (Similarly with "exorcise" subcycles for failures.)

The end result will be a new PK state. It may contain newly initiated datums. Datums may have changed status. The priority factors and est count of the datums and the list of top level investigations may have altered. Various changes can alter the pool of available suggestions. Control now returns to the monitor to begin a new cycle.

SEER produces an investigation state snapshot at the beginning of each cycle, and a cycle trace of the activity during the cycle. The snapshot shows the state of the investigation chosen at the start of the cycle, and the cycle trace indicates the PK changes which occur during the cycle.

A sample snapshot is shown in figure 3 - 1 . It is basically an illustration of a piece of PK, similar to the PK illustrations shown in the previous chapter. Both the status of datums and their links to other portions of the PK are indicated. Rather than cutting off the display at some arbitrary depth, the snapshot stops at successes and failures, and continues below conjectures until reaching terminal nodes. The "#" indicates the datum explored in the previous cycle; "$$" indicates the datum chosen to be explored in the current cycle.

A cycle trace is also based on a PK illustration; take away all the extra information that can be shown, and we are left with a picture of the PK below and/or above the explored datum, indentations reflecting the tree structure as usual, arrows representing links. In this case, rather than illustrating to a set depth, we include all those nodes or links affected (or perhaps created) by the cycle operations. The trace in figure 3 - 2 , from an early cycle in the example used in chapter one, illustrates a simple cycle with exploration but no exploitation. A few cycles further on in the same example, in fact the same investigation, we find the cycle in figure 3 - 3 , which has both exploration and several exploitations (a computation is performed when D16 is explored, which succeeds). Figure 3 - 4 presents this same cycle diagrammatically. (The snapshot used above was produced at the beginning of this cycle.)

Normally a good deal of other information is shown in a cycle trace to indicate the changes in the PK that occur during the cycle. We will introduce some of this other information as needed, as we begin to use cycle traces for illustration below. Eventually when we discuss exploration and

FIGURE 3 - 1

STATE OF INVESTIGATION
OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
AT START OF CYCLE C15

-> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) CONJECTURE
 -> D9 (CONVEX-HULL R196) CONJECTURE <- (D8)
  -> D17 (VEX-STR-ANA R196) SUCCESS
  -> $$ D16 (CONVEX-HULL-COMPUTATION R196) $$ CONJECTURE
 -> D10 (AREA R196) SUCCESS <- (D8)

FIGURE 3 - 2

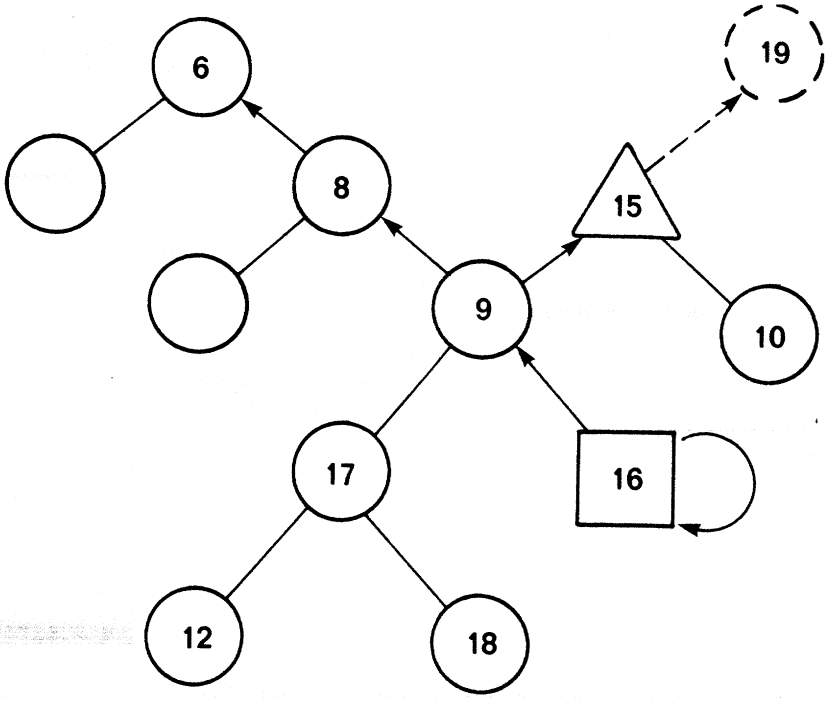
```
                    CYCLE C12
                    INVESTIGATING D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT: D9 (CONVEX-HULL R196)

EXPLORE D9
---> * D17 (VEX-STR-ANA R196)
---> * D16 (CONVEX-HULL-COMPUTATION R196)
```

FIGURE 3 - 3

```
                    CYCLE C15
                    INVESTIGATING D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT: D16 (CONVEX-HULL-COMPUTATION R196)

EXPLORE D16
EXPLOIT D16
<--- D9 (CONVEX-HULL R196)
     EXPLOIT D9
     <--- D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
          EXPLOIT D15
          <--- * D19 (SMOOTHED-CONVEX-HULL R196)
     <--- D8 (CONVEX-NEEDS R196)
          EXPLOIT D8
          <--- D6 (CONVEX R196)
```

FIGURE 3-4

CYCLE 15



Legend:

Subject of investigation

Explored datum

Added to PK during cycle

Exploration

Exploitation

exploitation, we will present the complete cycle traces for the two cycles shown above. The snapshots and cycle traces will be used to illustrate examples in later chapters; to a certain extent I employ them to give solidity to the discussion, emphasizing that there is a program behind all this that printed out these materials. However, the program was not especially sophisticated in its ability to describe its operations graphically, so the "syntax" of these figures takes a little getting used to; they could be ignored in a less than detailed reading.

Much of this chapter is organized around suggestions (or more generally datums), which are the basic element of SEER's control structure. I attempt to demystify this vague concept by responding to a series of questions I suspect will be in the reader's mind. In order this chapter:

expands on how suggestions are made
explains how the priority system chooses which one to explore next
describes what exploration involves
tells how a datum is established
explains how SEER exploits a datum when it is established
explains how advice functions as suggestion
indicates how failure functions in a complementary manner to success
indicates how the recognition process begins and ends.

"Suggestion" will be used mainly in its narrow sense in these sections: a suggestion is an unexplored conjecture; "making a suggestion" is inititiating a conjecture. "Suggestion" also has a broader usage connoting the various exploitation and exploration processes of the control structures: initiating, encouraging, establishing. This reflects my "historical" development of the "suggestion based control structure" concept. The initial motivation was: "suggest something relevant to what you have or want." The concept quickly developed to respond to a dynamic data base (e.g. is the suggestion already there) and deeper analysis (e.g. what kinds of relevance are there; can we encourage without initiating).

HOW ARE SUGGESTIONS INITIATED, I.E. PLACED AS DATUMS INTO THE PK?

This is really a whole series of questions: how, when, why, where, by whom, which, what is involved? The most general answer is: any old way. Suggestions theoretically can be added to the data base at any time, from any program. While such general "power" is nice, obviously we must analyze, and to some degree automate the concept for it to be useful.

BY WHOM? Suggestions are made by other datums, acting on the initiation relations contained

in the GK.

WHEN? A datum makes suggestions when it is explored and when it is established. Roughly it is "explored" when the monitor first decides to do something about following it up, and established when it is determined to be a true statement about the scene (see below).

WHICH, WHY, WHERE? When a datum is explored, it makes suggestions that will help establish it or affect its priority (either). (Alternatively a computation may be performed.) When established it suggests datums that it will help establish and whose likelihood it affects (both), i.e. uncertain properties make suggestions when established. (Some versions of the GK permit some certain properties to make suggestions.) Suggestion upon establishment is the key element of the suggestion based control structure. There is room to experiment here with the conditions which will prompt such suggestions. At exploration time a datum suggests conjectures below it in the establishment and priority structures. When established it suggests conjectures above itself.

HOW? How does a datum know which suggestions to make? The datum has a corresponding quantum of GK. This quantum has initiation links to other quantums. These links provide the I-related statements for the datum, as explained elsewhere. The I-related statements above the datum are suggestion statements to be "proposed", below it, to be "pursued".

The criteria for suggestion described above translate in GK into relationships between the link types. For example, A pursue-initiates B if and only if B establishes A or B encourages A. The arguments are instantiated to form datums from the quantums, using the instantiation functions attached to the links. These functions, remember, act on the PK context, normally the arguments of the initiating datum in particular.

The conjunction of GK and PK in creating new suggestions is part of the implementation of active knowledge at processing time. In simple cases, suggested datums will have the same arguments as the initiator. At other times, e.g. in pursuing a part of a whole, the generation of appropriate arguments is a substantial issue. The generation of arguments in the instantiation of suggestions will be discussed in more detail later.

In figure 3 - 5 D23 initiates D24 during exploration. In figure 3 - 6 D23 initiates D25 during exploitation. The asterisk in a cycle trace indicates an initiated datum.

FIGURE 3 - 5

```
                    CYCLE C21
                    INVESTIGATING D23 (HAS-BAR-SIDES-M1 R196)
                    AT: D23 (HAS-BAR-SIDES-M1 R196)


                    STATE OF INVESTIGATION
                    OF D23 (HAS-BAR-SIDES-M1 R196)
                    AT START OF CYCLE C21

-> $$ D23 (HAS-BAR-SIDES-M1 R196) $$ C
 -> # D22 (HAS-BAR-SIDES-M1-N R196) # S

EXPLORE D23
---> * D24 (HAS-BAR-SIDES-M1-C R196) <Exploration of D23 initiates D24.>
---> D22 (HAS-BAR-SIDES-M1-N R196)
```

FIGURE 3 - 6

```
                    CYCLE C22
                    INVESTIGATING D23 (HAS-BAR-SIDES-M1 R196)
                    AT: D24 (HAS-BAR-SIDES-M1-C R196)


                    STATE OF INVESTIGATION
                    OF D23 (HAS-BAR-SIDES-M1 R196)
                    AT START OF CYCLE C22

  -> # D23 (HAS-BAR-SIDES-M1 R196) # C
   -> $$ D24 (HAS-BAR-SIDES-M1-C R196) $$ C
   -> D22 (HAS-BAR-SIDES-M1-N R196) S

  EXPLORE D24
  EXPLOIT D24
  <--- D23 (HAS-BAR-SIDES-M1 R196)
       EXPLOIT D23
       <--- * D25 (HAS-BAR-SIDES R196) <Exploitation of D23 causes initiation of
D25.>
            EXPLOIT D25
            <--- * D28 (IS-A-BAR-M1 R196)
            <--- * D27 (W-AXIS-M1-N R196)
            <--- * D26 (L-AXIS-M1-N R196)
```

WHAT IS INVOLVED? What is involved in a datum, D, actually initiating a suggestion? First a new datum, D', is created as a node in the PK structure with a suggested statement, I-related to D. The properties of a datum are set up with appropriate initial values, based on knowledge connected with the associated quantum.

This completes the actual initiation. However, initiation occurs as part of the exploration and exploitation processes that will be discussed further below. Initiating a datum will be accompanied by link activity as well. Links may be set up between initiator and suggestion. Initial processing of new priority and establishment links is performed.

Note when a statement is suggested a datum with that statement may be present in the PK already. (If the datum is present, the links the initiator knows about may or may not also be present.) Care must be taken to avoid duplication, two datums with the same statement. We say that D initiated D', e.g. in determining an initiation path, only if D actually first entered D' into the PK.

It is natural to refer both to suggesting a statement and suggesting the datum initiated with that statement. Strictly speaking suggestion involves "pursuing" or "proposing" a statement, which leads to initiation of a datum with that statement; if such a datum is not already present. Pursuing and proposing are system functions described below.

ORDERING SUGGESTIONS

The order in which suggestions are made depends on the structure of the GK and on results during processing of a particular scene. The order in which they are executed is determined by the monitor employing the priority system. These aspects of SEER are particularly open to experimentation and development.

PRIORITY ELEMENTS

There are three elements employed at present in priority calculations: likelihood, difficulty, and interest. GK quantums and PK datums have values for each of these factors. The difficulty factor is a measure of time. The difficulty associated with a quantum indicates the average time required to establish a datum instantiated from that quantum using what are a priori the most promising methods. The difficulty associated with a datum estimates the time required to complete the establishment of the

datum. This figure obviously moves up and down with partial successes and failures in the investigation.

Likelihood represents a sort of probability estimate. Formal probabilities are difficult to apply. The figures associated with quantums represent the a priori likelihood of a datum corresponding to the quantum holding true in a scene. The likelihood of a datum reflects the current likelihood taking into account previous PK results.

We often use these two factors together either as a ratio, likelihood/difficulty, the priority ratio, or as a product. The values assigned to quantums become the initial values for corresponding datums. These datum values are updated during processing. The initial assignments are largely determined by working upward from the bottom of the GK E-structure. We define first difficulty and likelihood of quantums.

For AND nodes, the difficulty is essentially the sum of the difficulty of the nodes below in the difficulty (establishment) structure. An initial likelihood is assigned for AND nodes. The difficulty and likelihood of quantums is supposed to reflect a priori values for PK instantiations, so we must consider how these will occur. When we sum over the nodes below, we must count nodes more than once if the instantiation functions indicate several instantiations will be made when we transfer to the PK. For example, in figure 3 - 7 , Q45 links to the quantum Q16, but there are two instantiation functions as we need the length axis of both arguments. Thus the difficulty of Q16 must be counted twice. Figure 3 - 8 shows two length axis datums being pursued. In more complex instantiations depending on the state of the PK or involving a looping process, we will not know how many datums will be pursued. For example, in figure 3 - 9 , D62 will only produce an instantiation like D63 if an appropriate hammer-part-handle has already been found. In these cases we must also assign an initial difficulty value.

For OR nodes, we find the node below whose likelihood/difficulty ratio is maximum. The OR node receives the same likelihood and difficulty assignment. For primitive nodes, difficulty and likelihood are assigned.

Initial assignments, when required can be based on primitive learning, or the programmer can assign values, e.g. corresponding to a simple semantic quantization: likely, very likely, etc. Recall that the primitive nodes at the bottom have associated execution modules, where computations are

FIGURE 3 - 7

```
                    GK SEGMENT
                    BELOW Q45 PERP-N

Q45 PERP-N
     DIFF = 9.63 LIKE = 0.5 INT = 1. BBB = Q35
     AND
     LINKS:
     <--- (Q44)
     ---> L111
          INS: (EXG-CAR EXG-CDR)
          TYPES: (SE PUI A)
          ENC: 0.5
          TO:
          Q16 L-AXIS
               IS
               DIFF = 4.81 LIKE = 0.5 INT = 1. BBB = Q35
               OR
               LINKS:
               <--- (Q15 Q27)
```

FIGURE 3 - 8

```
                    CYCLE C53
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D71 (PERP-N R196 R140)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C53

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> D53 (HAS-A-HP2-M1-T DR1) C
     -> D59 (HAS-THE-HP2 DR1 R140) C
      -> D60 (HAS-THE-HP2-M1 DR1 R140) C
       -> D62 (H-REL-P2 DR1 R140) C
        -> D63 (H-REL-P1-P2 R196 R140) C
         -> D64 (T-JOINED R196 R140) C
          -> D66 (TOUCH R196 R140) C
           -> D67 (TOUCH-M1 R196 R140) C
            -> D69 (TOUCH-M1-N R196 R140) C
             -> # D65 (PERP R196 R140) # C <- (D64)
              -> $$ D71 (PERP-N R196 R140) $$ C
              -> D70 (PERP-C R196 R140) C
            -> D68 (TOUCH-M1-C R196 R140) C
          -> # D65 (PERP R196 R140) # C <- (D69)
       -> D61 (IS-A-HP2 R140) C
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D71
---> * D72 (L-AXIS R140)
---> D31 (L-AXIS R196)
```

FIGURE 3 - 9

```
                    CYCLE C46
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D62 (H-REL-P2 DR1 R140)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C46

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> D53 (HAS-A-HP2-M1-T DR1) C
     -> D59 (HAS-THE-HP2 DR1 R140) C
      -> # D60 (HAS-THE-HP2-M1 DR1 R140) # C
       -> $$ D62 (H-REL-P2 DR1 R140) $$ C
       -> D61 (IS-A-HP2 R140) C
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D62
---> * D63 (H-REL-P1-P2 R196 R140)
```

performed. A simple learning process can keep track of execution times during processing trials and average them to produce a difficulty figure. For higher level nodes requiring an assignment we can look below at establishment time and sum the computations performed. Likelihoods can also be learned in a simple minded fashion, as a ratio of successes to attempts, though more sophisticated learning or programmer assistance will be required to avoid distortions. When a quantum is first introduced to the GK difficulties are set to a ballpark figure, likelihood to .5 or an extreme if obviously required.

As I indicated, when a datum is initiated it receives the initial difficulty and likelihood values of the corresponding quantum. PK results update these values. When datum B is established (or fails) this affects the priority of any datum, A, above it in the P-structure.

For OR nodes, if B is established, so is A, and its priority is not really an issue -- difficulty is 0 and likelihood 1. If B fails, SEER has to take a new maximum ratio to obtain likelihood and difficulty, as in the initial computation, but over a depleted set of possibilities. We return to the quantum corresponding to A and take the max in the GK structure as before but eliminating those quantums from consideration that correspond to already failed datums below A in the PK structure.

For an AND node, if B fails, A fails, and again priority is a fairly moot issue. When B is established, the difficulty of A is recomputed by subtracting from it the initial difficulty of B. Updating the likelihood of A is a bit more complex.

The likelihood-priority link from B to A has an associated likelihood value, called the encouragement A receives from B. The link says that if we know B holds, the likelihood of A will be at least this value. We compare the contribution of B to A with the current likelihood of A. The new current likelihood is the greater of the two. (This encouragement can also be learned by keeping track of the outcome of A after B has been established.)

Note that B may be P-linked to A yet not E-linked. There are several uses for this possibility. When two datums below A in the E-structure are established, the likelihood of A may become greater than when either was established alone. This effect can be handled through the link structure by employing a node representing this combination, which is below A in the likelihood-priority structure, though not in the establishment or actuate structure. Other results, e.g. broad contextual assertions, affect priority, without being establishing requirements.

We handle the likelihood effects of combinations of results explicitly because the contributions of individual results are fixed and independent. They are not true conditional probabilities varying with the effects of a changing PK context.

When SEER updates the priority of A based on the establishment of B, we say it is firing or acting on the P-link between A and B. In figure 3 - 10 the P-link implied by the SE link between D19 and D21 is fired. (As an SE link implies an SP link the traces do not indicate them separately. Short arrows are used in a cycle trace to indicate changes in properties of datums, as opposed to the longer arrows which represent the links.)

SEER acts on the P-link between A and B while exploiting the establishment of B, if the link exists, or is created, at that time (see the exploitation section). The link will not be present for exploitation, if B does not initiate A and A is not present when B is established, or if B simply does not know about the P-link and A has not created it yet. In that case, if A creates the P-link for the first time when A is executed, and B is established already, SEER will act on the P-link then, as part of the exploration process (see the exploration section). SEER works similarly for failure of B.

I have decided that the first order system need only update from established nodes. We could try to push partial results on up the structure. Previous versions of SEER did this to a degree for the difficulty factor. However, it creates many technical problems, even more so for the likelihood factor. I feel we can use some experience with this simple system, and some insight into what kind of a processing model it reflects before implementing schemes that have more continuous feedback on priority changes as results are required.

Note that we have been talking about priority "link" in the singular. Actually we need two links, one for difficulty, the other for likelihood. A datum may affect one factor of another datum without affecting the other. E.g. a "certain" property required in a computation, adds to its difficulty, but does not affect its likelihood of success.

The final priority factor is "interest". This factor, or set of factors, covers a multitude of concerns. I will discuss the possible uses of such factors further below. For the moment we can think of it as a preference factor. The idea is that two datums may have an equal likelihood/difficulty ratio, and yet we may still prefer to investigate one over the other. The interest factor expresses this and

FIGURE 3 - 10

```
                    CYCLE C17
                    INVESTIGATING D19 (SCH R196)
                    AT: D20 (SCH-C R196)


                    STATE OF INVESTIGATION
                    OF D19 (SCH R196)
                    AT START OF CYCLE C17

-> # D19 (SCH R196) # C
 -> $$ D20 (SCH-C R196) $$ C
 -> D15 (SCH-N R196) S

EXPLORE D20
EXPLOIT D20
<--- D19 (SCH R196)
     EXPLOIT D19
     <--- * D21 (SCH-AS-DESC R196)
          FIRED: SE <SE and SP links between D19 and D21 fired.>
          DIFF -> 0.96 <Difficulty factor changed.>
```

tells us by how much. We assign an interest factor to quantums and corresponding datums have this interest value.

PRIORITY AND MONITOR

How do we put the priority factors together to help determine which suggestion to explore next? The monitor uses a two stage process. First it chooses which investigation to work on, then which suggestion within that investigation to explore. The investigations are represented by the top datums in the E-structure. Those which are still conjectures are held on an investigation list. The priority of those datums, and their investigations, is defined:

priority = (interest x likelihood) / difficulty

The monitor first chooses the highest priority investigation. (SEER keeps track of the "top priority investigation", updating if necessary when priorities change.) The next stage involves two operations in turn. First SEER chooses the best bet below (BBB) the subject of the investigation. This will be the most promising, uncertain, accessible datum in the investigation. Finally it chooses the accessible suggestion S below the BBB, with minimal difficulty. S is explored.

A datum D' is accessible from D if there is a path down from D in the A-structure, through explored datums of status C. Also, at serial nodes the path must go through the first status C subnode. (A quantum is accessible from another simply if there is a path in the A-structure, going through the first branch at serial nodes.)

The BBB of a datum D is determined recursively, in the A-structure, by the following definition. (The first four clauses are in the form of a "cond"; that is, employ the first of these that applies.)

The BBB of D is:

(1) if D is unexplored: NIL (but see clause (v))

(2) if D is serial: the BBB of the first conjecture immediately below D

(3) if D is an OR node: the BBB of the conjecture immediately below with maximum priority ratio

(4) if D is an AND node: the BBB of the conjecture Ci immediately below, such that the ratio of the likelihood of Ci and the difficulty of the BBB of the quantum of the BBB of Ci is minimum for all

the Ci

(5) finally, if in any of these cases the BBB is computed to be NIL, change it to D itself, if D is uncertain.

You will notice that I threw in a ringer: the BBB of the quantum of a datum. But this is really a key to what is going on. In the GK, BBB's have been computed, very much as above. These tell us "a priori" what lies ahead. However, in the GK, there are no established or failed results, to distinguish from conjectures. For example, in computing BBB in the GK we always take the first branch at a serial node, we always have all the OR options open. As PK results accumulate we must recompute the BBB in the PK. If the first serial branch is already established we move down through the second, if one OR branch has failed, we ignore it.

Why all the rules for proceeding downward? Basically, we want to look next at the most promising suggestion below and our primary concern is with the uncertain aspects of the investigation. At serial nodes, however, we have no choice. At OR nodes, we also have a desire to pursue first the branch which overall has the best priority ratio. This takes precedence over BBB concerns.

In figure 3 - 11 , suppose the BBB of D1 is B1, of D2, B2, B1 and B2 are primitive. B1 is a good BBB, but it is overwhelmed by the difficulty in the rest of D1. If D were an AND, this would not matter, we have to do everything anyway. But since we have an OR option, our desire to avoid the great difficulty of D1 takes precedence, and we choose the D2 branch before we worry about the BBB.

At an AND node, the extra complication arises to deal with the situation in figure 3 - 12 . The established node below D2 has raised its priority to .9. The likelihood of B2 has also been raised, in context, but the priority system is not sophisticated enough to reflect this. Thus a truer picture is obtained by taking the ratio of the likelihood of the higher node, with the difficulty below. In computing this difficulty, we first go down in the PK as far as we can. We will reach some unexplored node. Then we return to the GK to find the difficulty of the BBB of the corresponding quantum. If the BBB, D, in the PK is a primitive node, then this difficulty will simply be the difficulty of D.

FIGURE 3-11

AN OR BRANCH BBB EXAMPLE

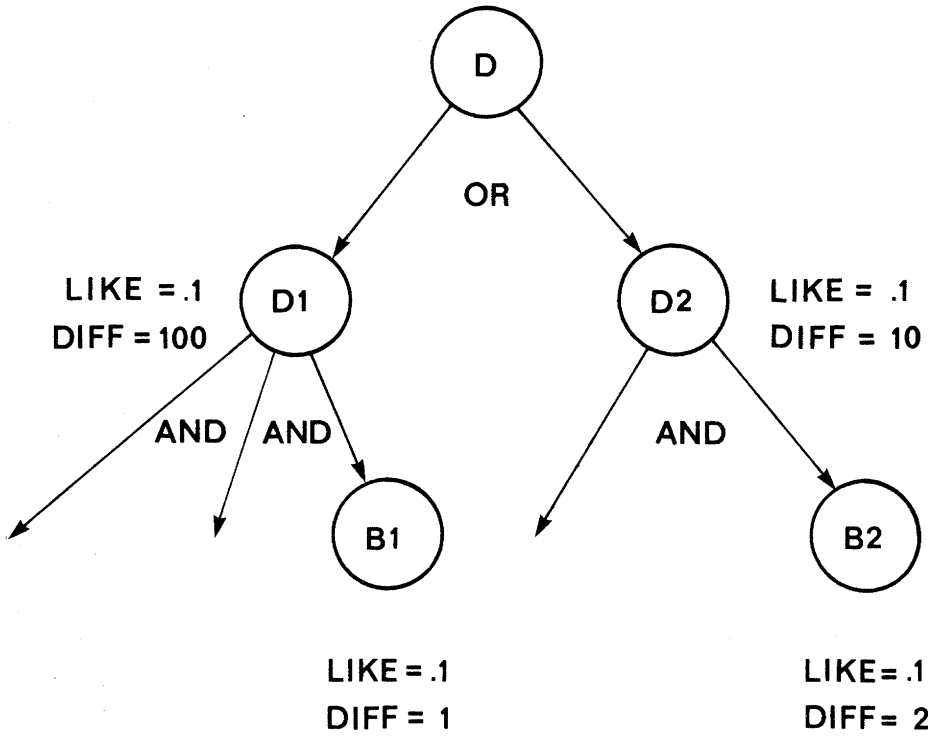
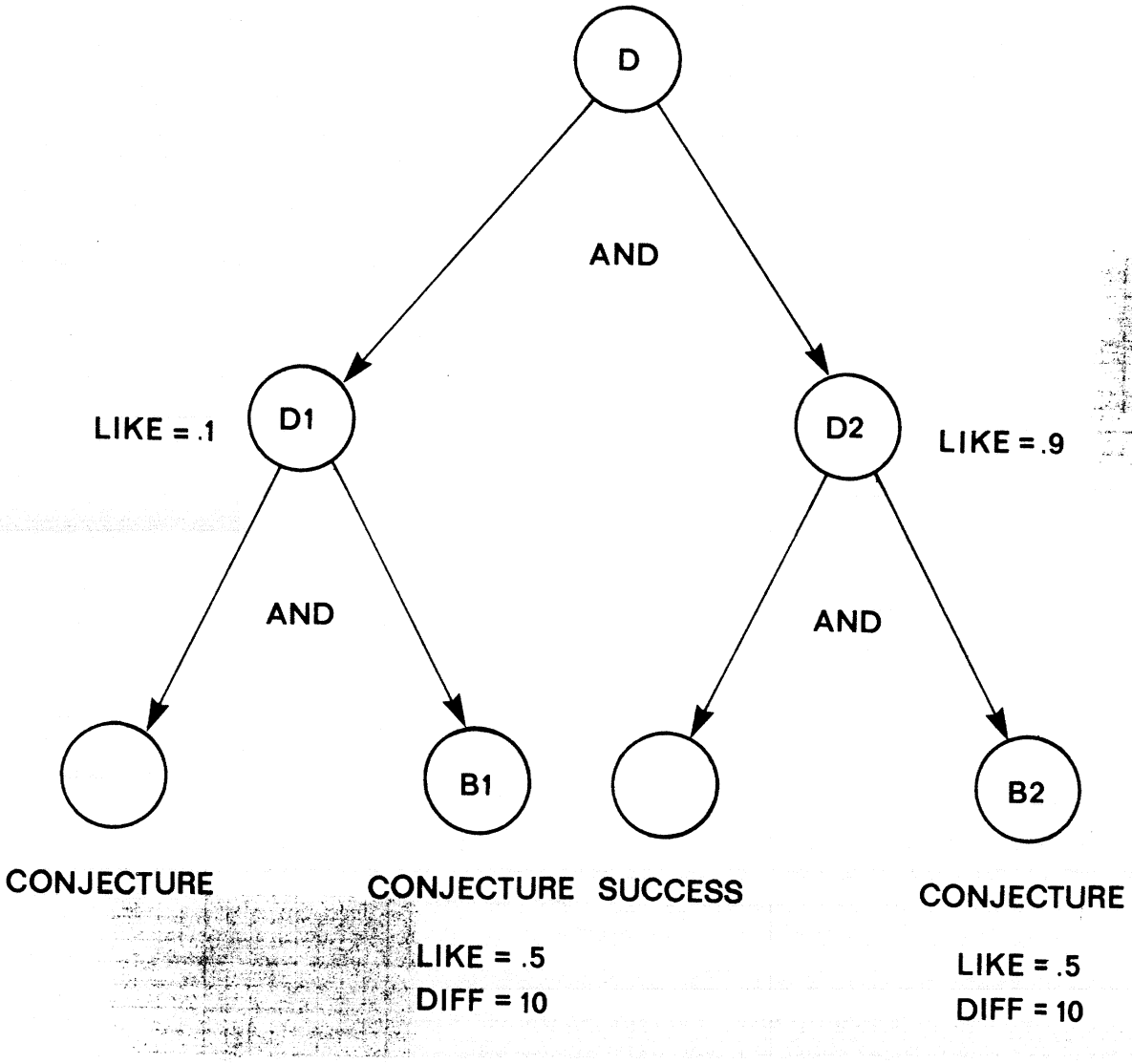


FIGURE 3-12

AN AND BRANCH BBB EXAMPLE



WHAT IS INVOLVED IN EXPLORING A SUGGESTION?

There are two possibilities. We may "pursue" further datums. When there are no further datums to pursue, we stop and perform a computation.

In the latter case, the suggestion corresponds to a primitive quantum, at the bottom of the GK structure. This quantum has an associated execution module "black box" computation. This module is applied to the suggestion datum. It may expect other PK information to be available for use. If the computation succeeds, it establishes the suggestion. A value may also be returned if appropriate. If the computation fails, the suggestion fails.

In the former case, SEER first obtains the relevant statements below the suggestion being executed, together with information about the GK links which led to these statements. Each statement, with the associated link information, is pursued.

Pursuing the statement first involves looking for a datum, B, with that statement in the PK. If one is not present, such a datum, B, will be initiated, if the statement is I-relevant to A. This is the initiation of a new suggestion during exploration, as described above. Links are entered between A and B, corresponding to the types of relevance the statement holds to A.

The statement may be relevant to A, but not I-relevant. A is then not empowered to initiate a B, and the corresponding links can only be entered if B is already present. On the other hand, if B is present, the links, or some of them, may be present already as well. This will depend on the status of B and its knowledge of the links to A. Note also that there may be datums or potential datums, relevant to A which A does not know about. Some of these may have nevertheless already entered links to A.

After the instantiated statements are pursued, the links below the suggestion are explored. This involves firing the unfired E and P links from established or failed datums, as described elsewhere. The suggestion may have just pursued an already present and established (or failed) datum. (Normally, this datum, would have fired the links when established; however, it may not have known about the links, and the suggestion, being unexplored at the time, would not have entered them as yet.) Firing these links may even cause the suggestion to be established or fail right away. It would then be exploited or exorcised.

Figure 3 - 13 illustrates a simple type of cycle, with the initial exploration phase, but no

FIGURE 3 - 13

```
                    CYCLE C12
                    INVESTIGATING D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT: D9 (CONVEX-HULL R196)


                    STATE OF INVESTIGATION
                    OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT START OF CYCLE C12

-> # D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) # CONJECTURE
 -> $$ D9 (CONVEX-HULL R196) $$ CONJECTURE <- (D8)
 -> D10 (AREA R196) SUCCESS <- (D8)

EXPLORE D9
---> * D17 (VEX-STR-ANA R196)  <A datum is created.>
     ENTERED: (PUI SI SE A)  <Links are created from D9 to D17.>
     FIRED:  <No links are fired.>
---> * D16 (CONVEX-HULL-COMPUTATION R196)
     ENTERED: (PUI SI SE A)
     FIRED:
```

Exploration: a datum pursues the results required to establish itself.

exploitation. The figure indicates the state of the PK below datum D15 at the start of the cycle, and includes a cycle trace showing the operation of the cycle and the changes produced in the PK. Recall that an asterisk before a datum indicates that the datum has just been initiated (added to the PK).

In the cycle represented in this figure, datum D9 was explored (again "$$" marks the datum currently being explored, "#" marks the previously explored datum). The monitor first found that the D15 "investigation" was the most promising to pursue. It then chose the D9 datum as the best point at which to follow up on this investigation. The exploration here has initiated the datums D16 and D17. They are added to the suggestion pool. Figure 3 - 14 illustrates the state of the D15 investigation at the beginning and at the end of the cycle (i.e. beginning of the next cycle). This cycle reflects the following: a convex-hull conjecture knows that it requires a convexity structure analysis, on which it must perform a computation, in order to establish itself. Thus when explored the conjecture pushes out these further conjectures.

In figure 3 - 15 D49 is explored and pursues D51 and D47; the former is initiated, the latter is present. In figure 3 - 16 , D21 is explored, pursues D6, and finds D6 already present, but not linked to D21. The link is entered. In figure 3 - 17 D58 is explored, finds D31 and D38 already present and established, the links are entered and fired and D58 is established and exploited immediately.

In summary, explorations push an investigation downward until basic computations are made at the bottom or previous successes are encountered. Results then work their way up, as we shall see in the next sections.

A DATUM IS ESTABLISHED WHEN IT IS FOUND TO HOLD TRUE FOR THE SCENE

There are two ways in which a datum may be established. If the datum corresponds to a primitive quantum, execution results in a decision on whether the datum is established or fails. Otherwise, we must wait until results obtained on the suggestions pursued below the datum "back up" to determine its status. An OR datum will be established when any datum below it in the E-structure is established. An AND datum will be established when all datums below it in the E-structure are established.

How does SEER know when these conditions occur? The suggestions below a datum are not

FIGURE 3 - 14

```
                    STATE OF INVESTIGATION
                    OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT START OF CYCLE C12

-> # D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) # CONJECTURE
 -> $$ D9 (CONVEX-HULL R196) $$ CONJECTURE <- (D8)
 -> D10 (AREA R196) SUCCESS <- (D8)


                    STATE OF INVESTIGATION
                    OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT START OF CYCLE C13

-> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) CONJECTURE
 -> # D9 (CONVEX-HULL R196) # CONJECTURE <- (D8)
  -> $$ D17 (VEX-STR-ANA R196) $$ CONJECTURE
  -> D16 (CONVEX-HULL-COMPUTATION R196) CONJECTURE
 -> D10 (AREA R196) SUCCESS <- (D8)
```

FIGURE 3 - 15

```
                    CYCLE C34
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D49 (IS-A-H-M1 DR1)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C34

-> $$ D49 (IS-A-H-M1 DR1) $$ C
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D49
---> * D51 (HAS-A-HP2 DR1)
     ENTERED: (PUI SI SE A)
     FIRED:
---> D47 (HAS-A-HP1 DR1)
     ENTERED: NIL
     FIRED:
```

FIGURE 3 - 16

```
                    CYCLE C18
                    INVESTIGATING D21 (SCH-AS-DESC R196)
                    AT: D21 (SCH-AS-DESC R196)


                    STATE OF INVESTIGATION
                    OF D21 (SCH-AS-DESC R196)
                    AT START OF CYCLE C18

-> $$ D21 (SCH-AS-DESC R196) $$ C
 -> D19 (SCH R196) S

EXPLORE D21
---> D6 (CONVEX R196)
     ENTERED: (PUI SI SE A)
     FIRED:
---> D19 (SCH R196)
     ENTERED: NIL
     FIRED:
```

FIGURE 3 - 17

```
                    CYCLE C40
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D58 (HAS-A-HP2-M1-GNN DR1 A1)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C40

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) C
     -> # D56 (HAS-A-HP2-M1-GN DR1 A1) # C
      -> $$ D58 (HAS-A-HP2-M1-GNN DR1 A1) $$ C
      -> D57 (HAS-A-HP2-M1-GNC DR1 A1) C
     -> D55 (HAS-A-HP2-M1-GC DR1 A1) C
    -> D53 (HAS-A-HP2-M1-T DR1) C
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D58
---> D31 (L-AXIS R196)
     ENTERED: (SE PUI A)
     FIRED: SE
     DIFF -> 10.6
---> D38 (WIDTH R196)
     ENTERED: (SE PUI A)
     FIRED: SE
     STATUS -> S <D58 is established as a success,>
     EXPLOIT D58 <and exploited.>
     <--- D56 (HAS-A-HP2-M1-GN DR1 A1)
          ENTERED: NIL
          FIRED: SE
          DIFF -> 1.0
```

necessarily executed right away or in any set order. In any case they may pursue further suggestions.

Here is where we employ the establishment links. When a datum B is established it fires or acts on the E-links to datums, e.g. A, above it. It tells these datums that B has been established. If A is an OR node, it is now established in turn. If A is an AND node, firing the E-link subtracts 1 from its est count. The quantum associated with A knows how many quantums are below in the E-structure, i.e. how many datums below A need to be established to establish the AND. This number is the initial est count for A, when A is initiated. When the est count reaches zero, A is established.

When B is established the E-link to A, even A itself, may not be present. Often B will add them to the PK as part of the exploitation process, and then act on the link. However, B may not know about A, or the E-link, or be unwilling to initiate A (no I-link). Nevertheless A may be E-related to B, and A may create the link when A is executed. A will act on the E-link then upon finding B to be established already, as we learned in the previous section.

It is possible for a suggestion to be established through exploitation of datums below even before it is explored (in which case we never need explore it). Of course, its priority can also change before exploration. Partial results encourage exploration by improving priority factors.

Notice how the establishment process propagates. A datum established at one level may cause datums above to established. These in turn inform the datums above them, some of which may be established in turn... In any case, partial results accumulate.

In summary, SEER pursues suggestions down to primitive datums. Establishment of these then propagates back upward. In figure 3 - 18 D34 is executed and succeeds. D34 is established and exploited. D33 is then established; then D35 is established. We have seen several other examples in previous figures.

AS SOON AS DATUM B IS ESTABLISHED SEER EXPLOITS THAT NEW PK KNOWLEDGE

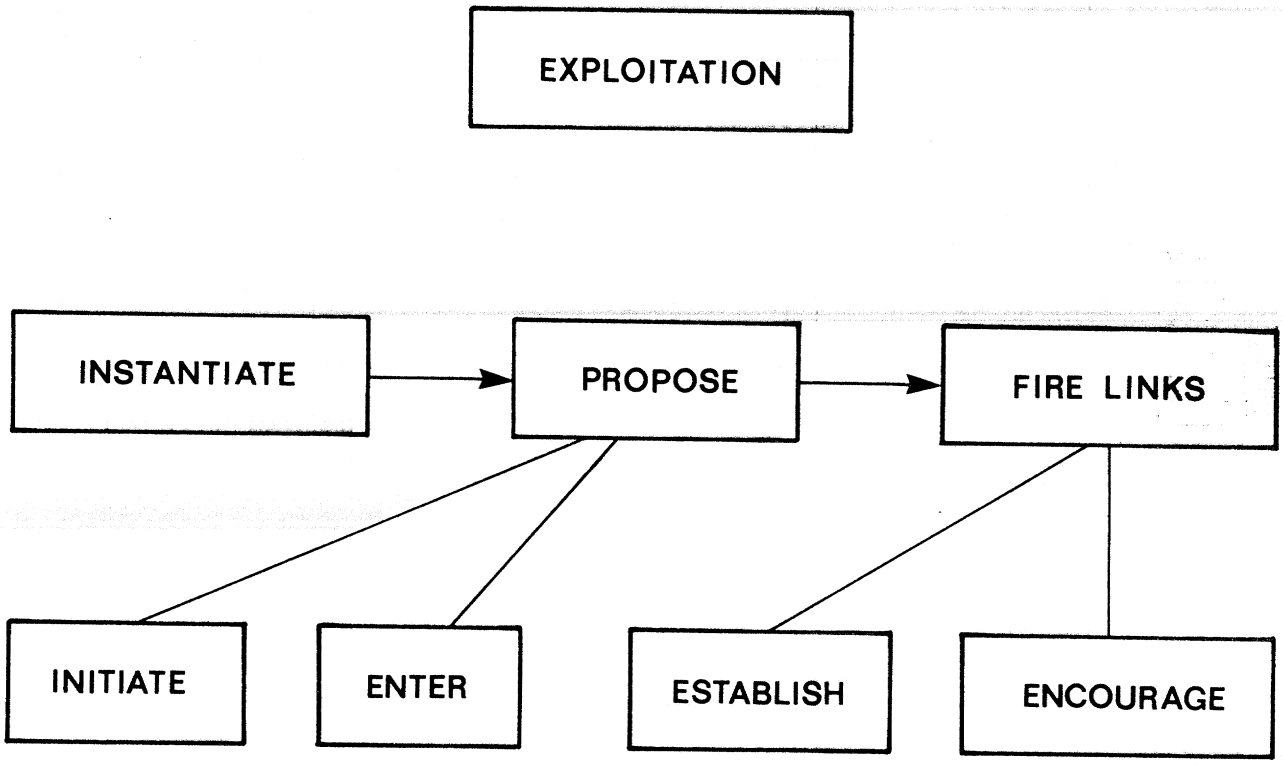
The exploitation process is analogous to the exploration process. There is an instantiation phase which obtains relevant statements below the datum. These statements are proposed, initiating new suggestions and entering new links as required. Finally the E and P links are exploited. A link will be fired in this case if the higher datum is still a conjecture. Figure 3 - 19 is a block diagram of the

FIGURE 3 - 18

```
                    CYCLE C27
                    INVESTIGATING D33 (W-AXIS-M1 R196)
                    AT: D34 (W-AXIS-M1-C R196)


                    STATE OF INVESTIGATION
                    OF D33 (W-AXIS-M1 R196)
                    AT START OF CYCLE C27

-> # D33 (W-AXIS-M1 R196) # C
 -> $$ D34 (W-AXIS-M1-C R196) $$ C
 -> D27 (W-AXIS-M1-N R196) S

EXPLORE D34
EXECUTED: 0.17 SECONDS 0.23 REALTIME
STATUS -> S
EXPLOIT D34
<--- D33 (W-AXIS-M1 R196)
     ENTERED: NIL
     FIRED: SE
     -INVES
     STATUS -> S
     EXPLOIT D33
     <--- * D35 (W-AXIS R196)
          +INVES
          ENTERED: (SE SI PUI A)
          FIRED: SE
          -INVES
          STATUS -> S
          EXPLOIT D35
          <--- * D36 (WIDTH-M1 R196)
               +INVES
               ENTERED: (PUI SI SE A)
               FIRED: SE
               DIFF -> 1.0 LIKE -> 1.0
               PRIORITY -> 1.0
NEW TOP PRIORITY INVESTIGATION: D36 (WIDTH-M1 R196)
```

FIGURE 3-19

EXPLOITATION



exploitation process.

The cycle trace in figure 3 - 20 illustrates exploitation. This is the cycle that we illustrated graphically above. Datum D16, convex-hull-c R196, has been explored. Rather than pursuing further datums, the exploration executes a computation, which is successful and causes D16 to be established. (Recall that short arrows indicate changes in datum properties, here a change of status, e.g. D16 is no longer a conjecture, but a success.) This result is then exploited. Exploiting D16 establishes D9 by firing a link between D16 and D9; D9 becomes a success: we have the convex hull of region R196. D9 is exploited in turn. Exploitation propagates further upward in several directions. In summary, as a result of all this activity:

A new suggestion is initiated: SCH 196.

The datums D8, D9 and D15 are established.

The priority of D6 is increased.

Figure 3 - 21 is taken from another cycle trace. Here we have established the PK datum D45: we have a possible hammer handle. Exploiting D47 involves:

Suggesting an inclusive hammer.

Advising a "method" for establishing an associated hammer head. This method will take advantage of the fact that the handle has already been found.

Figure 3 - 22 presents another example. First, D31 obtains the related statements above by returning to the GK and instantiating. The I-related statements are initiated as new datums D27 and D32, unless datums with those statements are present already, as in the case of D27. Links of various flavors are entered to datums present in the PK with related statements. E.g. an E-link is entered for the E-related statement of D27. (In general these datums may have just been initiated or been present already. If they are not present, they cannot be linked to. The links may also be already present, having been entered from above.) Unfired P and E links are acted upon. (We act on the E-links first; if a datum above is established it is no longer necessary to act on the P-link. In addition to the links that D31 knew about, and just entered if necessary, there could have been other links D31 did not know about, to datums which entered the link when explored.)

There are various ways of viewing these processes, describing and structuring them. The

FIGURE 3 - 20

```
                    CYCLE C15
                    INVESTIGATING D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT: D16 (CONVEX-HULL-COMPUTATION R196)

                    STATE OF INVESTIGATION
                    OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT START OF CYCLE C15

-> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) CONJECTURE
 -> D9 (CONVEX-HULL R196) CONJECTURE <- (D8)
  -> D17 (VEX-STR-ANA R196) SUCCESS
  -> $$ D16 (CONVEX-HULL-COMPUTATION R196) $$ CONJECTURE
 -> D10 (AREA R196) SUCCESS <- (D8)

EXPLORE D16
EXECUTED: 0.05 SECONDS 0.06 REALTIME  <A computation is performed.>
STATUS -> SUCCESS  <Datum sixteen succeeds.>
EXPLOIT D16  <This result is exploited...>
<--- D9 (CONVEX-HULL R196)  <along a link up to datum nine.>
     ENTERED: NIL  <No new links were entered; links were present already>
     FIRED: SE  <The success of datum sixteen fires a link...>
     STATUS -> SUCCESS  <which leads to the success of datum nine.>
     EXPLOIT D9  <Datum nine is exploited in turn.>
     <--- D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
          ENTERED: NIL <No new links entered.>
          FIRED: SE <Success-establishment link between D9 and D15 fired.>
          -INVES <D15 is removed from the investigation list because...>
          STATUS -> SUCCESS <D15 is established.>
          EXPLOIT D15 <When established it is exploited.>
          <--- * D19 (SMOOTHED-CONVEX-HULL R196)  <A new datum is created.>
               +INVES  <And added to the list of unresolved investigations.>
               ENTERED: (PUI SI SE A) <New links entered between D15 & D19.>
               FIRED: SE <Fire SE link; D19 not established; fire SP link,>
               DIFF -> 1.0 LIKE -> 1.0 <leading to priority factor changes,>
               PRIORITY -> 0.99 <and an updating of priority.>
     <--- D8 (CONVEX-NEEDS R196)
          ENTERED: NIL
          FIRED: SE
          STATUS -> SUCCESS
          EXPLOIT D8
          <--- D6 (CONVEX R196)
               ENTERED: NIL
               FIRED: SE
               DIFF -> 0.17

NEW TOP PRIORITY INVESTIGATION: D19 (SMOOTHED-CONVEX-HULL R196)
```

Exploitation: results propose what they can be useful for.

FIGURE 3 - 21

```
<...A possible hammer handle has been found.>
                    EXPLOIT D45
                    <--- * D46 (IS-A-HAMMER-PART-HANDLE R196)
                         +INVES
                         ENTERED: (SE SI PUI A)
                         FIRED: SE
                         -INVES
                         STATUS -> SUCCESS
                         EXPLOIT D46
<An encompassing hammer is suggested.>
                         <--- * D47 (HAS-A-HAMMER-PART-HANDLE DR1)
                              +INVES
                              ENTERED: (SE SI)
                              FIRED: SE
                              -INVES
                              STATUS -> SUCCESS
                              EXPLOIT D47
                              <--- * D49 (IS-A-HAMMER-METHOD1 DR1)
                                   +INVES
                                   ENTERED: (PUI SI SE A)
                                   FIRED: SE
                                   DIFF -> 417.51
                                   PRIORITY -> 119.75
<A method for establishing the hammer head is advised.>
                              <--- * D48 (HAS-A-HAMMER-PART-HEAD-METHOD1 DR1)
                                   +INVES
                                   ENTERED: (PUI SI SE A)
                                   FIRED: SE
                                   DIFF -> 267.51
                                   PRIORITY -> 1.86E-3
```

Suggestion and Advice

FIGURE 3 - 22

```
                    CYCLE C25
                    INVESTIGATING D29 (L-AXIS-M1 R196)
                    AT: D30 (L-AXIS-M1-C R196)


                    STATE OF INVESTIGATION
                    OF D29 (L-AXIS-M1 R196)
                    AT START OF CYCLE C25

-> # D29 (L-AXIS-M1 R196) # C
 -> $$ D30 (L-AXIS-M1-C R196) $$ C
 -> D26 (L-AXIS-M1-N R196) S

EXPLORE D30
EXECUTED: 0.29 SECONDS 0.39 REALTIME
STATUS -> S
EXPLOIT D30
<--- D29 (L-AXIS-M1 R196)
     ENTERED: NIL
     FIRED: SE
     -INVES
     STATUS -> S
     EXPLOIT D29
     <--- * D31 (L-AXIS R196)
          +INVES
          ENTERED: (SE SI PUI A)
          FIRED: SE
          -INVES
          STATUS -> S
          EXPLOIT D31
          <--- D27 (W-AXIS-M1-N R196)
               ENTERED: (PUI SI SE A)
               FIRED: SE
               -INVES
               STATUS -> S
               EXPLOIT D27
               <--- * D33 (W-AXIS-M1 R196)
                    +INVES
                    ENTERED: (PUI SI SE A)
                    FIRED: SE
                    DIFF -> 1.0 LIKE -> 1.0
                    PRIORITY -> 1.0
          <--- * D32 (LENGTH R196)
               +INVES
               ENTERED: (PUI SI SE A)
               FIRED: SE
               DIFF -> 1.0 LIKE -> 1.0
               PRIORITY -> 1.0
NEW TOP PRIORITY INVESTIGATION: D33 (W-AXIS-M1 R196)
```

exploitation and exploration model implemented at the moment is not the most efficiently organized. However, it is desirable at the moment for its pedagogical clarity and extensible flexibility.

This immediate exploitation of a success (and similarly for a failure) is the key element of SEER's control structure. Results do not sit passively waiting to be called for. They do not function only to passively answer calls from above. Inspiration and processing flow may proceed upwards independently, and actively. Even if results are called for from above in the course of furthering an investigation, these results are not limited to responding locally for that one purpose. In conjunction with the GK knowledge of relevance relationships, and the present PK links, a PK result can make suggestions, affect priorities, provide partial results in a number of areas "simultaneously". The GK instantiation functions acting on the PK are a specific active knowledge mechanism.

Exploitation propagates upward. When a datum is established its exploitation may establish others in turn, which are then exploited. As we see in the above figure this process can branch out. The first datum to be exploited in a cycle will be the datum chosen for exploration. To reach this datum we moved down from the subject of the investigation chosen for the cycle. Exploitation will fire links back upwards along that path, and further exploitations may carry us further along, perhaps all the way to the subject node. (When the subject is established it can propose higher level possibilities.) However, exploitation can also fire links to other parts of the investigation, or out to other investigation. New investigations can be proposed. Further exploitations can push upward in these areas. The results may affect other investigations more than the original subject of the cycle. The original investigation may succeed or fail, or relative priorities may change such as to cause a new investigation to assume top priority.

The three functions -- initiating datums, and influencing their establishment and priority -- which I have discussed individually above, come together in the exploitation process. They may be seen as aspects of the single process of acting on relationships between datums, as dictated by links in GK and PK. These functions are also carried out during exploration as explained earlier.

Exploitation and establishment follow from acting on the E-links; priority changes from acting on the P-links; initiation from acting on the I-links; the monitor acts on the A-links. The control structure keeps track of whick links have been acted upon or fired. SE (sucess-establishment) is the

most common abbreviation you will need to follow the link firings in the traces of cycle activity. As we noted, since an SE implies an SP link the traces do not list them separately, e.g. when a figure indicates that an SE link has been entered or fired, this will imply that a SP link has also been entered or fired (unless the SE firing has led to a success that makes the SP firing unnecessary). Initiation firings (adding datums to the PK) are indicated by the asterisk notation and actuation firings are reflected in the explorations, so these are not listed separately as "fired" in the traces.

The various options presented by different combinations of link types and different states of the PK (nodes and links present or not) complicate the execution and exploitation operations. Some of these alternatives may be unnecessary; others permit subtle technical and heuristic options and allow for varied processing flows; we will see some examples later. The exploration and exploitation processes have been redesigned several times. Exploitation, in particular, lies at the heart of SEER's control structure.

ADVICE ON HOW TO DO SOMETHING IS REALLY A SUGGESTION ON WHAT METHOD TO USE

This observation allows us to treat advice as we do suggestions. A piece of advice is a method, a specific node among a set of OR'd approaches to establishing a datum. (There are more general ways of viewing advice in SEER, of course. As advice is suggestion, so any suggestion is "advice" on what to do. The priority system advises which datums to execute when.)

A method may be "advised" by:

initiating it
increasing its priority or est count
adding it to the pool of accessible suggestions.

The latter refers to a serial situation in which there are preconditions for the application of a method. The conditions are first in the serial order below the method in the E-structure. They "advise" the method when established by adding the actual computation to the accessible suggestions available for exploration. The three forms of advising thus correspond to suggesting an approach, encouraging it, and permitting it. These functions can, of course, overlap.

A result B can advise a method A when established, or A can "look for" advice B when explored. In figure 3 - 23 D31 advises method D33 (through D27). In figure 3 - 24 the preconditions

FIGURE 3 - 23

```
                    CYCLE C25
                    INVESTIGATING D29 (L-AXIS-M1 R196)
                    AT: D30 (L-AXIS-M1-C R196)

                    STATE OF INVESTIGATION
                    OF D29 (L-AXIS-M1 R196)
                    AT START OF CYCLE C25

-> # D29 (L-AXIS-M1 R196) # C
 -> $$ D30 (L-AXIS-M1-C R196) $$ C
 -> D26 (L-AXIS-M1-N R196) S

EXPLORE D30
EXECUTED: 0.29 SECONDS 0.39 REALTIME
STATUS -> S
EXPLOIT D30
<--- D29 (L-AXIS-M1 R196)
     ENTERED: NIL
     FIRED: SE
     -INVES
     STATUS -> S
     EXPLOIT D29
     <--- * D31 (L-AXIS R196)
          +INVES
          ENTERED: (SE SI PUI A)
          FIRED: SE
          -INVES
          STATUS -> S
          EXPLOIT D31
          <--- D27 (W-AXIS-M1-N R196)
               ENTERED: (PUI SI SE A)
               FIRED: SE
               -INVES
               STATUS -> S
               EXPLOIT D27
               <--- * D33 (W-AXIS-M1 R196)
                    +INVES
                    ENTERED: (PUI SI SE A)
                    FIRED: SE
                    DIFF -> 1.0 LIKE -> 1.0
                    PRIORITY -> 1.0
          <--- * D32 (LENGTH R196)
               +INVES
               ENTERED: (PUI SI SE A)
               FIRED: SE
               DIFF -> 1.0 LIKE -> 1.0
               PRIORITY -> 1.0
NEW TOP PRIORITY INVESTIGATION: D33 (W-AXIS-M1 R196)
```

FIGURE 3 - 24

```
                    CYCLE C23
                    INVESTIGATING D26 (L-AXIS-M1-N R196)
                    AT: D26 (L-AXIS-M1-N R196)


                    STATE OF INVESTIGATION
                    OF D26 (L-AXIS-M1-N R196)
                    AT START OF CYCLE C23

-> $$ D26 (L-AXIS-M1-N R196) $$ C
 -> D25 (HAS-BAR-SIDES R196) S <- (D28 D27)

EXPLORE D26
---> D12 (O-BDRY R196)
     ENTERED: (PUI SI SE A)
     FIRED: SE
     -INVES
     STATUS -> S
     EXPLOIT D26
     <--- * D29 (L-AXIS-M1 R196)
          +INVES
          ENTERED: (PUI SI SE A)
          FIRED: SE
          DIFF -> 1.0 LIKE -> 1.0
          PRIORITY -> 0.99
NEW TOP PRIORITY INVESTIGATION: D29 (L-AXIS-M1 R196)
```

of a method are sought. One, D25, is present and successful and has already fired a link to D26. An outer boundary is pursued, D12 is found already present and established, a link is entered and fired. D26 succeeds and proposes the method D29. In figure 3 - 25 , finding a head advises methods for finding a handle. In figure 3 - 26 D264, when explored, finds a method has already been suggested.

In particular, advice implements the heuristic approaches: We have x already, how does this help y? We want to do y, what x would help? Special cases versus general approaches are encouraged by advice. A datum may pursue several methods for establishing itself, depending on advice from below to support one, as in the previous figure.

A datum may also pursue a particular method for accomplishing a subgoal. In this case SEER "working down in the context of x" advises a particular method for establishing y. E.g. in looking for the boundary of a region conjectured to be an object which is symmetric, we could push a particular boundary finding method that tries to use symmetry. (There is also the possibility for "exploiting" conjectures which is not done at present.) More generally this working down in context allows specialized techniques that may be regarded as another form in which "advice" enters SEER.

THE MECHANISMS FOR FAILURE ARE BASICALLY ANALOGOUS TO THOSE FOR SUCCESS

We "exorcise" a failure, as we exploit a success. A failure can contribute to the failure, or success, in turn, of other datums, affect their priority, initiate them. There is, however, a subtlety associated with the procedural aspect of SEER that induces some "asymmetry" of success and failure. The establishment network really is concerned with our execution of a process. Success in our "model" is not strictly equivalent to "truth" in the real world "interpretation". Of course, we assume that if we establish a datum, its statement will hold true. However, if a datum fails, this does not imply in general that it is false. The failure of X is not equivalent to the success of "not x".

The most common failure of a datum results from the failure of all the methods we know about that can establish it. SEER is thus simply unable to establish the datum. We have seen how this type of failure is naturally accounted for through the establishment structure. If one subnode of an AND or all subnodes of an OR fail, the node fails. Actually, "failure" is relative here. If the link from A to B is "FE", the failure of B helps establish A. The success of B in this case would be counted toward the

FIGURE 3 - 25

```
                    CYCLE C220
                    INVESTIGATING D50 (IS-A-HP2-M1 R196)
                    AT: D250 (COVER-END-C R196 R145)


                    STATE OF INVESTIGATION
                    OF D50 (IS-A-HP2-M1 R196)
                    AT START OF CYCLE C220

-> D50 (IS-A-HP2-M1 R196) C
 -> D233 (HAS-A-HP2P1 R196) C
  -> D234 (HAS-A-HP2P1-M1 R196) C
   -> D235 (HAS-THE-HP2P1-M1 R196 R145) C
    -> D237 (HAS-THE-HP2P1-M1-CE R196 R145) C
     -> D240 (COVER-END R196 R145) C
      -> D251 (COVER-END-N R196 R145) S
      -> $$ D250 (COVER-END-C R196 R145) $$ C
     -> D239 (CONVEX R145) S <- (D258)
     -> D238 (HAS-THE-HP2P1-M1-CE1-M2 R196 R145) S
    -> D236 (HAS-THE-HP2P1-M1-CF R196 R145) C
 -> D44 (IS-A-HP2-W R196) S

EXPLORE D250
EXECUTED: 0.95 SECONDS 21.36 REALTIME
STATUS -> S
EXPLOIT D250
<--- D240 (COVER-END R196 R145)
     ENTERED: NIL
     FIRED: SE
     STATUS -> S
     EXPLOIT D240
     <--- D237 (HAS-THE-HP2P1-M1-CE R196 R145)
          ENTERED: NIL
          FIRED: SE
          STATUS -> S
          EXPLOIT D237
          <--- D235 (HAS-THE-HP2P1-M1 R196 R145)
               ENTERED: NIL
               FIRED: SE
               STATUS -> S
               EXPLOIT D235
               <--- * D259 (HAS-THE-HP2P1 R196 R145)
                    +INVES
                    ENTERED: (SE SI PUI A)
                    FIRED: SE
                    -INVES
                    STATUS -> S
                    EXPLOIT D259
               <--- D234 (HAS-A-HP2P1-M1 R196)
                    ENTERED: NIL
                    FIRED: SE
                    STATUS -> S
```

```
                    EXPLOIT D234
                    <--- D233 (HAS-A-HP2P1 R196)
                         ENTERED: NIL
                         FIRED: SE
                         STATUS -> S
                         EXPLOIT D233
                         <--- D50 (IS-A-HP2-M1 R196)
                              ENTERED: NIL
                              FIRED: SE
                              -INVES
                              STATUS -> S
                              EXPLOIT D50
                              <--- * D260 (IS-A-HP2 R196)
                                   +INVES
                                   ENTERED: (SE SI PUI A)
                                   FIRED: SE
                                   -INVES
                                   STATUS -> S
                                   EXPLOIT D260
                                   <--- * D261 (HAS-A-HP2 DR4)
                                        +INVES
                                        ENTERED: (SE SI)
                                        FIRED: SE
                                        -INVES
                                        STATUS -> S
                                        EXPLOIT D261
                                        <--- * D263 (IS-A-H-M1 DR4)
                                             +INVES
                                             ENTERED: (PUI SI SE A)
                                             FIRED: SE
                                             DIFF -> 150.0
                                             PRIORITY -> 333.33
                                        <--- * D262 (HAS-A-HP1-M2 DR4)
                                             +INVES
                                             ENTERED: (SE PUI SI A)
                                             FIRED: SE
                                             DIFF -> 133.84
                                             PRIORITY -> 3.73E-3
NEW TOP PRIORITY INVESTIGATION: D263 (IS-A-H-M1 DR4)
```

FIGURE 3 - 26

```
                    CYCLE C222
                    INVESTIGATING D263 (IS-A-H-M1 DR4)
                    AT: D264 (HAS-A-HP1 DR4)


                    STATE OF INVESTIGATION
                    OF D263 (IS-A-H-M1 DR4)
                    AT START OF CYCLE C222

-> # D263 (IS-A-H-M1 DR4) # C
 -> $$ D264 (HAS-A-HP1 DR4) $$ C
 -> D261 (HAS-A-HP2 DR4) S <- (D262)

EXPLORE D264
---> * D266 (HAS-A-HP1-M1 DR4)
     ENTERED: (SE SI PUI A)
     FIRED:
---> D262 (HAS-A-HP1-M2 DR4)
     ENTERED: (SE PUI SI A)
     FIRED:
---> * D265 (HAS-A-HP1-M3 DR4)
     ENTERED: (SE PUI SI A)
     FIRED:
```

failure of A. For example, the failure of one method might suggest another. In general, methods further emphasize our concern with the success and failure of a process rather than a real world "fact".

We may also need to directly "disestablish" a datum. This says that it is useless to try to establish the datum, we will fail, or that we are no longer interested in establishing the datum. In the most obvious example, x is established, and disestablishes a "not x" conjecture. In the case of seed suggestions, once we have "accounted for" a region, we are not interested in pursuing the seed further. Again the seeds provide a good example of our interest in the process; the seeds merely want to explore certain conjectures, success or failure will satisfy them. See the next section.

The "disestablishment" structure, such as it is, can be reduced to OR connectives. That is, if a disestablishment link, SD or FD, is fired, it simply causes the failure of the datum above. If a conjunction of factors is required, these can combine first in an AND in the regular E-structure.

Certain methods may be developed specifically as options to be tried after failure of other approaches. This can be arranged in a serial OR. This may be one of those cases, however, where our serial urges are merely an ad hoc, local priority effort. In that case, we can leave the monitor/priority system to handle sequencing. Suggestion and advice between the methods can come into play. Most simply, one method can have "F" links to another: FI, FE, FP.

Since success and failure are handled in such similar fashion, I often omit explicit reference to failure alternatives elsewhere in this report.

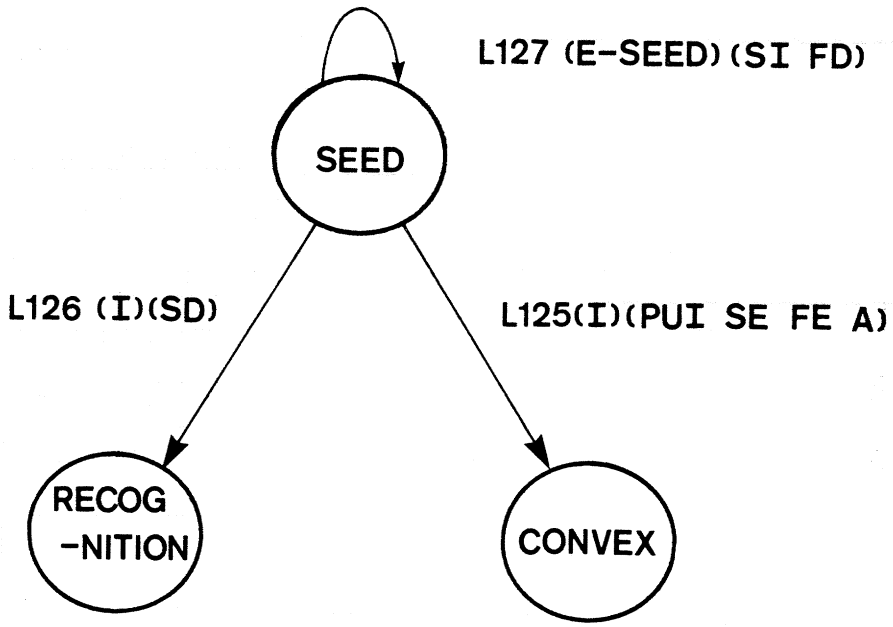
STARTING AND STOPPING

Recall that SEER first runs a region growing pass over the scene (see chapter five). This provides a set of potentially interesting regions, arranged in a tree structure, the affinity tree. Regions that "stand out" in the scene, e.g. are bright or distinctively homogeneous, will likely appear high in this tree. SEER initiates the suggestion based cycling process by introducing "seed suggestions". At the moment this process is quite primitive. A single seed quantum, seed, is used. It is illustrated in figure 3 - 27 . It provides some examples of the use of varied link combinations.

This suggestion pursues the basic uncertain suggestions about a region. (These may also be referred to as seeds.) A basic uncertain quantum is an uncertain quantum with no accessible uncertain

FIGURE 3-27

SEED GK



quantums below it. E.g. in figure 3 - 28 convex and convex-c are uncertain, but the area must be computed first and that is a certain predicate. Therefore convex is a basic uncertain predicate. In fact at the moment it is the basic uncertain predicate. I am not sure whether that is a feature (discovery) or a bug. A basic uncertain datum has a basic uncertain corresponding quantum.

Initially, seed suggestions for the highest, non background regions of the affinity tree are placed in the PK. A seed datum succeeds whether the datums it pursues are established or fail; it is only interested in getting them processed. Success or failure of the convexity conjecture will establish the seed. When it succeeds it proposes the same suggestion, seed, for the subregions in the affinity tree of the argument of the exploited suggestion. If a region is accounted for, e.g. recognized as a hammer head, the seed suggestion for that region can be disestablished.

More generally, when a region is accounted for, investigations headed by conjectures about that region can be removed from the list of investigations used by the monitor. A record of accounted for regions can be kept to inhibit future exploitation initiation of suggestions about these regions. (We may still want to pursue further information about them, for other investigations; we just are not interested in investigating them for their own sake.)

Also, accounted for regions, and those below them in the affinity tree, need not be considered when searching for candidates for conjectured recognitions. The links are indicated; e.g. link 126 goes from the seed quantum to the recognition quantum. The instantiation function is the identity function. There is one type of link--success-disestablishment. Seed knows about PUI, SE, FE, and A links from seed to convex. However, convex does not know about these links. Conjectures about affinity tree regions are not allowed to become the top priority investigation until the seed suggestion has been established. We do not want it established until it has been explored. Thus convex cannot exploit seed until seed has entered the link. This is a device to direct our attentation at higher level affinity tree nodes before lower ones. The interest factor of the seed node is low though, so we pursue the proposals that come from exploiting results on regions, after establishing the seed, before going on to other seeds.

At the moment, SEER is simply set to stop when a hammer has been found in the scene.

Notice that the seed suggestions are not processed in a "pass". The initial seeds form the

FIGURE 3 - 28

```
                    GK SEGMENT
                    BELOW Q35 CONVEX
                    TO DEPTH OF 2.
                    SHOWING PROPERTY LINKS

-> Q35 CONVEX <- (Q34)
 -> Q106 CONVEX-C
 -> Q105 CONVEX-N
  -> Q30 CONVEX-HULL <- (Q29)
  -> Q31 AREA <- (Q29)
```

intitial PK state. Immediately, the monitor begins the sequence of cycles that constitutes the body of SEER processing, moving from PK to PK'. As results initiate new suggestions they enter the "suggestion pool" with the seeds.

The seeds have a low priority. SEER will likely prefer the newer suggestions. However, when more interesting lines of inquiry do not pan out we can return to the seed and its immediate "progeny". Theoretically, the seeds underly all our higher level knowledge, and will eventually try everything by "brute force". Of course, this should be unnecessary as we get positive results and capitalize on them. Even negative results can suggest avoiding some work (or make positive suggestions). In any case the results we finally achieve at the next level up will initiate savings. Remember we do not move directly to the top, but upwards one step at a time, in general.

We can also return to the seeds when we have completed one recognition but parts of the scene remain unaccounted for. (We might consider beginning work on several areas of the scene in "parallel".)

4

PROGRAMMING STRUCTURES

WE PROGRAM SEER BY ADDING NEW GK

The GK structure should encourage and facilitate active knowledge programming. We organize the GK to work with the control structure in establishing and exploiting results. Heuristic ideas on how having one result can help find another are implemented through suggestion and advice links among the nodes. Of course, the GK is also definitional and descriptive, but we emphasize the process of acquiring the description, making the recognition. The GK represents possible programs for processing a scene as much as descriptions of visual phenomena. It emphasizes the relationships between phenomena that can be used to assist processing.

I will discuss the organization and design of GK. We now understand the basic elements of the knowledge structures, and the control structure motivation. We shall see something in this chapter of how the programmer goes about transforming his heuristics and relevance analysis into GK structure that will implement them during processing. As a programming language the GK is instantiated and "interpreted" by the control structure based on the PK developed from the scene. We need to represent the usual language mechanisms and computational structures, "loops" for example. Developing this implementation also brought key theoretical issues into relief. Generation, part/whole relationships, serial/parallel requirements are discussed.

ORGANIZING GK

Faced with a body of description or recognition knowledge we must break it up into nodes and enter appropriate links. The most basic GK structure is the establishment structure. This is basically an AND/OR definitional tree. Each predicate node P is either a terminal node with no subnodes or breaks up into subnodes. If P is an OR node the subnodes are alternative "methods" for establishing P. If P is an AND node, the subnodes are necessary conditions for establishing P.

NODES CAN BE "PARALLEL" OR "SERIAL"" IN THEIR SUBNODE STRUCTURE

Investigation may proceed on several parallel subnodes before any have been established or failed. They may all be explored and SEER can be working on further exploring the subnodes of one or another as priority dictates. In serial cases, each subnode has to be established or fail in order, before the next can be executed. (AND nodes require establishing each in order, obviously, OR nodes failing in order.)

Often desire for serial specification is found to reflect ad hoc ideas on efficient priority. SEER generally prefers to leave ordering decisions to the general priority system, which can decide on the basis of the developing global PK context. There are situations which require serial specification, however.

The simplest situation is really a technical one. If predicate P requires certain data to operate on, this data must be found before P can be computed. Thus we have the common structure in figure 4 - 1 , where P-N is needed for the P-C computatation. (It may be present already or need to be computed itself.)

Typically, we can analyze a method of establishing a predicate P as a computation P-C on some other set of predicates P-N. These can then be analyzed in turn. In particular the P-N may impose preconditions for trying (advising) a method, which must be satisfied before the actual computation occurs.

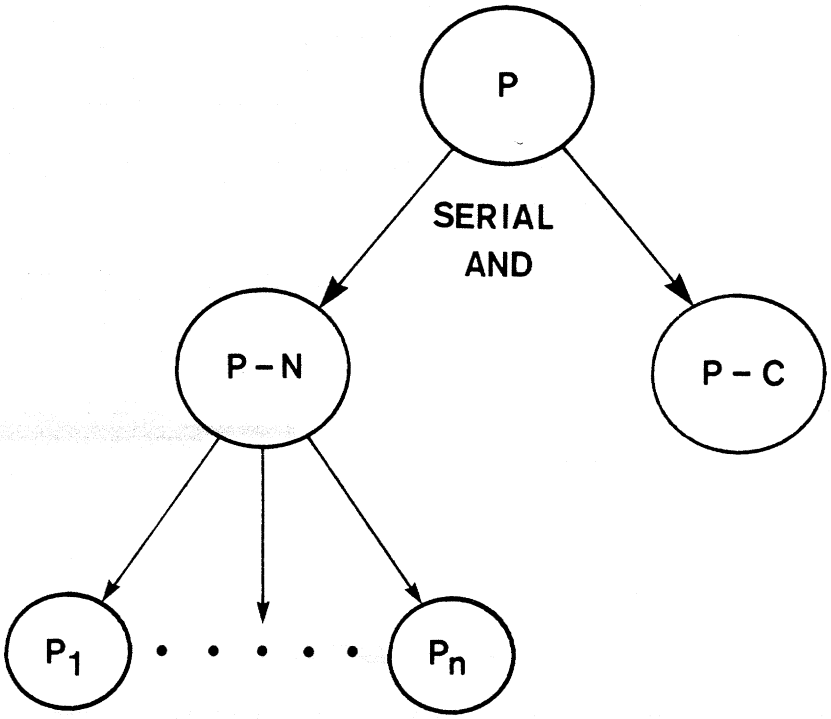
One method of implementing an iteration involves "OR'ing" a predicate and the remaining possibilities serially (see below). Here we want to require that the first possibility fails before going on to explore the others.

WHAT CONSIDERATIONS GOVERN THE MANNER IN WHICH WE BREAK UP OUR GK INTO NODES?

The basic considerations are description, control and programming: we must reflect the definition and provide a data base, allow suggestion and advice, and permit constructions like iteration. In a sense we employ "good programming practice" so that SEER can program itself specifically for the scene during processing. We break up predicates into subnodes, subroutines, whenever the subnode can be used in more than one description. Even when a subnode is only used once we may establish it

FIGURE 4-1

NEEDS/COMPUTATION



for clarity of organization.

A phenomena merits a separate quantum when we want to be able to exploit a corresponding datum result. Working "down", breaking up into datums provides for additional cycles at which processing direction can be reconsidered. The datums, of course, function as a data base of "assertions" that can be shared. A partial success, or failure, as a distinct datum can be fully exploited and shared individually. It is not lost as an internal result in a discarded investigation.

With alternatives as datums we can employ the priority/monitor systems to choose among them for us. A feature grouped inside a node must be handled by the programmer. As a separate node, the system functions can operate on the feature how and when they deem appropriate. This takes a burden off the programmer, and permits the global, scene driven decisions that the system is equipped to handle at run time.

There are other considerations that affect the organization of the GK. The break up into nodes, and the order in which they are entered, can affect priority decisions. There are various technical considerations, based on such fundamental considerations as facilitating suggestion and advice. For example, part/whole relationships are represented in a certain format (see below). We also discuss below the representation of programming structures like loops in the GK.

Descriptive goals impart their own considerations. For example, we would like our descriptive data base to be "asymptotically modular". As we enter descriptions, we want to build up a data base of nodes for features that can be used in further descriptions. Ultimately we hope that we will have enough basic features that most new descriptions will require very few if any new nodes. Thus the number of nodes added per description will be large at first, but tend eventually toward a small constant average. We would also like the GK to be easy to program and use. In particular, clarity of GK, and in a developing PK, is a useful criteria.

These various criteria may, of course, conflict at times. The basic consideration remains a knowledge structure designed for active knowledge. Remember that the exploitation and exploration referred to above encompass a variety of relevance relationships which are to be represented, first by organizing into nodes, then by entering the appropriate links.

NEXT WE NEED TO CONSIDER THE GENERATION FUNCTIONS THAT THE GK LINKS REQUIRE TO IMPLEMENT THE PK

Datums, you recall, explore or exploit other datums whose statements are determined by the GK relationships acting on the PK context. The GK links can provide the predicate names directly. The generation functions attached to the links must provide the arguments. Often these are simply the identity function: "handle x" explores "convex x".

Serious issues, however, arise in dealing with relationships, where one element is known and the other must be generated. This is the basic process which permits SEER to use relationships actively. One member of a relationship can help us find the other.

The generation problem arises in moving both up and down the relevance structures. Of central concern is the part/whole relationship. If we conjecture "hammer x", we must find a y such that "x has hammer handle y". If we have a "handle y" result, we suspect that there exists an x such that "hammer x" and "x has hammer handle y".

We will discuss the generation of new regions arguments further later on. Basically we can look for new regions in three ways. We can look in the affinity tree, e.g. for subregions of conjectured hammer x. We can look among the datums already present, e.g. for a "handle y". We can generate totally new regions. We can take a fresh look at the end of the conjectured hammer x to find a handle region. Or we can conjecture a "dummy" hammer region x to contain the established handle y. When we find the head z, y and z will constitute x.

This sort of generation may require a number of alternative conjectures. In implementation, these alternatives can be pursued or proposed in parallel. If the generation itself requires some effort we may want to generate the alternatives serially, not computing one until the previous one has failed. We can explore a set of alternatives as a parallel OR node, for example. Or we can employ the iteration techniques discussed below.

Generation complicates processing a great deal, of course. Priority, e.g. difficulty is hard to judge when we do not know how many alternatives will have to be tried. We need to decide when generation is worthwhile. For example, if it is too difficult or inefficient to look for a whole based on a certain part, we may not want to supply the links and generation functions to implement that path.

THERE ARE A LARGE VARIETY OF OPTIONS AVAILABLE IN ORGANIZING THE GK

These involve choosing nodes, link combinations, and generation functions; they can be used to represent various "semantic" possibilities, or to implement technical mechanisms. Consider the "sequencing" possibilities, for example. Datum D is relevant to D'. When D is established, D' may be present already or not. If not, D may initiate D' or not. If not D' may be initiated by another datum in the future, or not. Now we might imagine situations in which we would like D to affect D', its priority say, only if D' is already present, only if D' is initiated by D, only if D' appears later (or various combinations). The latter alternative, for example, requires D' to know about the P-link with D, while D does not.

If, in this latter case, we wished the relationship to be acted upon before D' was executed, a new mechanism, not difficult to imagine would be needed. This could be added to the initiation of a datum or worked through "demons". I have found that several mechanisms easily suggest themselves for accomplish most processing alternatives within SEER. The problem is more to restrain myself from generalizing to variations before a genuine need for them has been established.

PROGRAMMING WITH GK

We normally begin "programming" the GK with the E-structure. The AND/OR tree is set up. The various paths through the GK, remember, represent different descriptions and different programs for recognizing the phenomena. We will have some in mind originally. More can be added later.

We may build a GK structure to represent the possibilities for exploring a node downwards, for example. Looking over the structure afterwards we will see that it might be entered during processing from a variety of lower nodes, and processing paths could develop in various directions. Many of these we will want to allow for by including the appropriate links and perhaps some accessory nodes. Then again, entirely different methods can be added later to deal with new environmental complications or take advantage of advice from a previous result. The P, I, and A structures may require some additional nodes. These structures are, of course, stongly related to the basic E-structure. We enter the appropriate links.

The previous sections described how we structure the GK. The following sections will indicate

some standard forms that are used. We organize in a very "structured" manner. For example, we first provide the results needed for a computation, than perform the computation. On a higher level, we are encouraged to implement relevance relationships. If a general solution is not available, several methods can be conjoined and advised appropriately. We are encouraged to implement active knowledge potential in the GK link structure. In summary, the general relevance relationships and specific processing possibilities help organize the GK into nodes and provide the links between them.

PROGRAMMING LANGUAGE PRIMITIVES: THE BASIC PROGRAMMING COMPUTATION STRUCTURES CAN BE ACCOMPLISHED IN GK

SEER has a basic "parallel" orientation, but serial contructs can be effected. Loops, in particular, may be required, e.g. in generation. We can program iterative or recursive repetitions.

In the iterative construct, "loop" and "is" properties of datums are used to hold looping information. When an attempt fails it is flushed from the PK; the "looping" node can be re-explored. Figures 4 - 2 , 4 - 3 , 4 - 4 and 4 - 5 illustrate this sequence. (When running in some tracing modes the datums are not actually flushed, so you may see them in the figures.)

In the recursive construct, the initial node explores two further nodes. One of these is an honest computation, the other the recursive node, which pushes two more in turn. In generation, the nodes are pushed as a serial OR. The first node tests a possibility. If it fails the second node pushes two more: the next possibility and again the "rest" to be tried on failure. Figure 4 - 6 illustrates this structure.

Eventually I will want to provide the user with assistance in writing the GK, especially for such constructs, either in the form of "macros", a "higher level" language, or a simple "programming assistant". For an earlier version of the system, for example, I had a simple program that asked questions, suggested options, filled in some links, and "wrote" standard serial forms to some extent, interacting with the user.

"If, then" or "cond" constructs can be programmed using appropriate serial AND/OR configurations in the E-structure. A basic GK configuration is represented in figure 4 - 7 . The node labelled D may be a "dummy" of sorts. We may be unable or unwilling to generate N3 and N4 until we

FIGURE 4 - 2

```
                    CYCLE C37
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D52 (HAS-A-HP2-M1-GT DR1)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C37

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> # D48 (HAS-A-HP2-M1 DR1) # C
   -> $$ D52 (HAS-A-HP2-M1-GT DR1) $$ C
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D52
---> * D54 (HAS-A-HP2-M1-G DR1 A1)
     ENTERED: (SE PUI A)
     FIRED:
---> * D53 (HAS-A-HP2-M1-T DR1)
     ENTERED: (PUI SI SE A)
     FIRED:
```

FIGURE 4 - 3

```
                    CYCLE C43
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D53 (HAS-A-HP2-M1-T DR1)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C43

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> $$ D53 (HAS-A-HP2-M1-T DR1) $$ C
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D53
---> * D59 (HAS-THE-HP2 DR1 R140)
     ENTERED: (SE PUI A)
     FIRED:
```

FIGURE 4 - 4

```
CYCLE C89
INVESTIGATING D49 (IS-A-H-M1 DR1)
AT: D52 (HAS-A-HP2-M1-GT DR1)


STATE OF INVESTIGATION
OF D49 (IS-A-H-M1 DR1)
AT START OF CYCLE C89

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> $$ D52 (HAS-A-HP2-M1-GT DR1) $$ C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> D53 (HAS-A-HP2-M1-T DR1) F
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D52
---> * D114 (HAS-A-HP2-M1-G DR1 A2)
     ENTERED: (SE PUI A)
     FIRED:
---> * D113 (HAS-A-HP2-M1-T DR1)
     ENTERED: (PUI SI SE A)
     FIRED:
---> D54 (HAS-A-HP2-M1-G DR1 A1)
     ENTERED: NIL
     FIRED:
---> D53 (HAS-A-HP2-M1-T DR1)
     ENTERED: NIL
     FIRED:
```

FIGURE 4 - 5

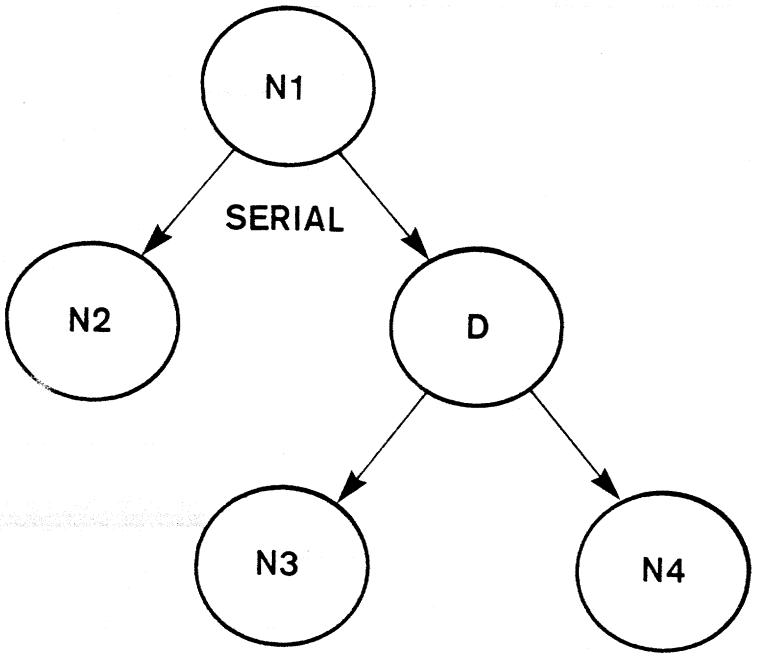
```
                    CYCLE C95
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D113 (HAS-A-HP2-M1-T DR1)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C95

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D114 (HAS-A-HP2-M1-G DR1 A2) S
    -> $$ D113 (HAS-A-HP2-M1-T DR1) $$ C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> D53 (HAS-A-HP2-M1-T DR1) F
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D113
---> * D119 (HAS-THE-HP2 DR1 R185)
     ENTERED: (SE PUI A)
     FIRED:
```

FIGURE 4 - 6

```
                    CYCLE C201
                    INVESTIGATING D50 (IS-A-HP2-M1 R196)
                    AT: D235 (HAS-THE-HP2P1-M1 R196 R145)


                    STATE OF INVESTIGATION
                    OF D50 (IS-A-HP2-M1 R196)
                    AT START OF CYCLE C201

-> D50 (IS-A-HP2-M1 R196) C
 -> D233 (HAS-A-HP2P1 R196) C
  -> # D234 (HAS-A-HP2P1-M1 R196) # C
   -> $$ D235 (HAS-THE-HP2P1-M1 R196 R145) $$ C
 -> D44 (IS-A-HP2-W R196) S

EXPLORE D235
---> * D237 (HAS-THE-HP2P1-M1-CE R196 R145)
     ENTERED: (SE SI PUI A)
     FIRED:
---> * D236 (HAS-THE-HP2P1-M1-CF R196 R145)
     ENTERED: (SE SI PUI A)
     FIRED:
```

FIGURE 4-7

DUMMY CONFIGURATION



have established N2. Therefore N1 pushes N2 and D. D may be a concept which we only grace with a distinct name to allow it to "mark time" waiting on N2. This configuration could be used for "if N2 then D', or "compute N3 and N4 using the value of N2" or "generate arguments for N3 and N4 based on the result of N2".

More sophisticated control structures of the "coroutine", "interrupt", or "tag" flavor, see [McDermott and Sussman 1974], tend to be handled by the system at the direction of results on the scene. The programmer's task is to provide options as needed.

Data base management is based on the "assertion" model. Values are stored in datum arguments or on their property lists. SEER emphasizes the resultant availability of results for active knowledge or simple sharing. There is some provision for flushing results of temporary, local, or expired interest. I have not devoted much energy to a good garbage collector yet, however.

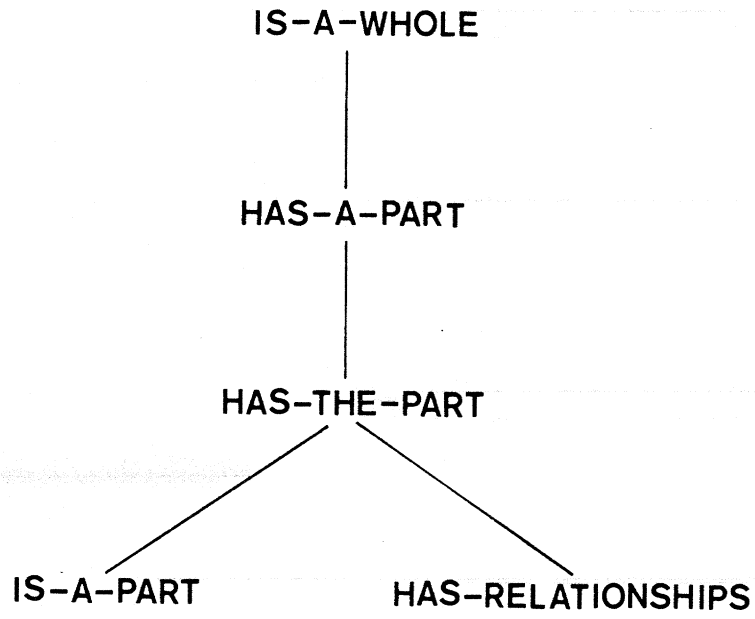
Of course, the execution modules for the primitive nodes are LISP programs with the full power of that language. The SEER system functions, the priority system, etc., are also written in LISP.

IMPLEMENTATION ISSUES AND THEORETICAL CONCERNS: A CASE STUDY

Many technical issues arise in implementing the seer system; these often illuminate larger theoretical issues, e.g. generation, parallel versus serial processing. We can take part/whole relationships as a case study that raises many of these technical and larger issues. The basic quantum structure for a part/whole concept is illustrated in 4 - 8 . The ususal predicate calculus or network representations of parts and wholes encourage us to think in terms of first recognizing all the parts and then testing the relationships. In SEER, of course, we want to encourage partial results to dynamically assist further acquisitions. The "has part" concept implements this here. In particular, parts as they are acquired can suggest wholes or advise how to find or verify further parts. Coming down through this structure we can seek parts in any order; pushing up, any part can suggest and advise. The has part concept is an AND and so at the lower level, we can also seek the "is-a" properties or the relationships in either order. Theoretically we can begin with either an independent recognition of a part, or by noticing a distinctive relationship, and move upwards to look for or suggest a whole. The relationship concept is quite flexible. We verify relationships to previously acquired

FIGURE 4-8

PART/WHOLE



parts. If relationships are used to help generate candidates in one method, that method need not verify the relationships separately. "Has part" breaks up into "has-a" and "has-the" to permit generation of alternative candidates. A whole must have a part, but we can have several possibilities for the the part before we find the right one. The need to overlay a loop structure here, the use of the "need/computation" construct, the possibility of several methods at each level, "as whole" nodes, all complicate the basic structure. Various standard organizational formats have been experimented with, but the basic components are as shown.

This structure is a good illustration of the application of the principles discussed above to GK design. The "whole" predicate is an AND of several parts. However, in pursuing a corresponding PK datum we may need to generate several alternative candidates for a part. These must be OR'd. The part/whole concept itself is again an AND of two elements: the identification of the part as such, and the proper relationship to the whole.

The given structure allows the system functions to operate, e.g. generation, exploration, establishment. It is also a natural model for the user to "program" with. Placing the two elements of the part/whole concept in parallel provides flexibility in achieving the result, and allows for active knowledge assistance. For example, the common approach would be to find the part, then compute the relationship. However, the relationship may "stand out" in a scene and it could suggest the identity of the part.

The relationship with the whole may actually involve relationships with other parts, especially if the whole is a "dummy region". These can only be verified between existing regions. Presumably the last part found will check its relationship to all other parts. However, there may be some "timing problems" here which require a top level node to verify that all relationships have been checked after all the parts are found.

There may be other ways in which the object "as a whole" is not merely the sum of its parts, in fact is a concept meriting its own quantum node. Holistic tests can serve as crude "filters" before finer part calculations. We saw the hammer-part-head-as-whole node used in this way. Aspects of the hammer-part-head-as-whole recogniton can also help the search for the face portion. In the case of a cylinder, the border shape of the whole cylinder, if known, provides three edges of the border of the

cylinder's side.

In one extreme, we may have methods of establishing a whole which do not require breakup into parts. There may be some "shortcut" connections through this structure. "Has-a-part-mi" may generate "has-the-part-mj" by using relationships to previously established parts. Thus, the latter may not need to pursue a "part-relationship" node. Establishing "is-a-part" can propose directly "has-a-part" for a dummy region.

We break this circularity by introducing the pseudo-object concept. The pseudo-object "hammer-part-handle", abbreviated "hp1", represents a "hammer part" corresponding to the handle. The properties that establish "R hp1" are sufficient to satisfy us that R is a hammer handle in the context of a whole hammer. "R hp1" can propose, e.g. "DR has a hammer handle" for a generated dummy region. However, if the hammer context does not prove valid, we have made no premature assertion that R is a hammer handle. ("R hp1" can propose that "R hammer handle", which may be established by itself.)

You may have noticed another circularity problem in verifying relationships of parts to wholes (and each other) as a recognition is being made. We do not want to acquire all parts first and then check relationships. We want to be able to use parts, as they are acquired, so required relationships can help acquire further parts and complete the recognition.

In seeking an hp2 for R we look to see if R has a hp1 which can guide the search for hp2. However, strictly speaking, at least if R is a dummy region, "R has a hp1" cannot be known until the relationships of hp1 and hp2 are verified. We therefore establish "has a" datums on the basis of relationships to what is already known. We assume the relationships are symmetric, and when the second element of a relationship is identified it will normally verify the relationship. The relationship is likely to have been verified anyway in being used to help generate the second element.

5

VISUAL MECHANISMS

THIS CHAPTER DESCRIBES SOME OF THE BASIC VISUAL COMPUTATIONS THAT SEER PERFORMS

We discuss the region finding problem, and the various descriptive properties and relationships. SEER, of course, emphasizes the use of context to permit special purpose techniques. The interactive aspects will not be emphasized in this chapter, which is concerned with providing an idea of the primitive visual mechanisms available. In discussing boundary shape, for example, we will introduce the general "vexity structure analysis", but will not discuss how we look specifically for a bar shape, or use symmetry to simplify shape verification.

I will discuss a number of mechanisms that I experimented with on my way to the SEER system. Not all are employed in the current system. Many have limited domains of reliable application. SEER is supposed to help us live with such problems. Nevertheless, this does not mean that I disagree with those who feel the importance of work on a good vocabulary of visual primitives. References to the literature on region finding can be found in the last chapter.

GENERATING REGIONS IS A CENTRAL CONCERN IN A VISION SYSTEM

We discuss first the initial pass that provides us with an "affinity tree" of potentially interesting regions. Once the main processing cycles have begun, we need to generate region arguments for suggestions. E.g. exploring a hammer conjecture, we look for handle and head regions. We may find such regions in the affinity tree, or we may directly generate new regions.

AFFINITY: A RELATIVE APPROACH TO REGION GROWING

There are two obvious choices for the atomic elements in a machine vision system: lines and regions. Regions seem particularly appropriate for dealing with non-planar objects, and for such properties as color and texture. A region analysis also seems more appropriate for an initial crude view of the scene.

The region finding process I describe is part of a system for object recognition. I only use it to provide some regions of possible interest at which to begin semantically guided processing of the scene.

I need a system that will:

1. provide regions that are likely to be of "semantic" interest (i.e. correspond to meaningful elements in the scene)

2. have the flexibility to deal with realistic scenes (e.g. texture, surface imperfections, lighting variations)

3. facilitate further higher level, semantic or global use of the region analysis.

Essentially I employ the heuristic insight that many visual decisions or perceptions are relative, relational and contextual, rather than absolute. I suggest a simple, specific, syntactic realization of a relative approach to initial regional units.

METHOD

The traditional region based approach has been founded on the concept of homogeneous or uniform surfaces. (Uniform in intensity or some other function.) This is a tolerable model as long as you are dealing with carefully painted white cubes in ideal lighting on a black background. But why do we see a hammer handle as a unit when it is grease stained and wood grained and lying in a toolbox? Homogeneity fails as a basis for initial processing.

I propose an extended notion of relative homogeneity or affinity. Pieces of a scene may hang together not because they are uniform, but because they are more drawn to each other than to other surrounding regions. The key point here is a relative as opposed to an absolute view. The pieces of the hammer handle may not be like one another. They are merely more like one another than they are like their other neighbors in the scene.

Absolute measures require thresholds. Picture segment A and picture segment B merge into the same region if $/f(A)-f(B)/ < h$, where f(A) is the result of applying function f to segment A, h is a threshold. Threshholds are traditionally tricky things. They can be tuned for one specified set of conditions; but generally the results are black magic to all but the tuner (maybe even to him). More importantly the results are not flexible enough to deal with a variable environment.

Relative decisions can be made without thresholds. A is surrounded by B and C. A merges with B if /f(A)-f(B)/ < /f(A)-f(C)/.

To be honest, thresholds may still be required, but they can be employed further on in the processing, when more global and semantic information is available. With an absolute homogeneity measure, we can make one pass over the scene conglomerating homogeneous points into distinct regions. With a relative approach, we iterate. At each stage we have a set of picture segments. Roughly speaking, we compare each segment with all its neighbors and merge the best match. Then we repeat with the new set of larger picture segments. When do we stop? That's the rub. We could make that decision syntactically. These decisions would require thresholds, though not necessarily fixed thresholds.

However, while these syntactic decisions are a fairly rich field for study, we do not have to make syntactic choices, at least not right away. We do not have to "stop". We can continue iterating merges until we have a single region comprising the entire scene. In the process we will have built a tree, with this all encompassing region as root, the original points or cells of the scene as leaves, and the intermediate conglomerations as intermediate nodes. Semantic knowledge can then be brought to bear on this tree.

While this tree structure is not the focus of my research, it does possess some properties that make it attractive as an initial data base. In particular, appropriate implementations of the affinity concept will cause many "natural" or "meaningful" regional units in the scene to appear, fairly accurately, as nodes on this tree, often near the top of the tree. The tree therefore provides us with attractive candidates for further study, after an appropriately global initial overview that raises us above the picture point level.

A certain amount of information on texture, curvature, highlights, and such is contained in the structure of the tree, if we wish to pursue it.

Most importantly, this approach can expect to produce useful results for various and inhomogeneous environments.

IMPLEMENTATION

We begin by dividing the grid of picture points into an array of "cells", or initial regions, each containing several points. This makes the initial domain manageable. (We can always "recurse" down for more detail.) If you prefer call the initialization defocussing or even "planning". (We may make an initial absolute homogeneity pass with a very strict threshold, to further cut down the data base quickly.)

Each initial region, R, has several neighboring regions. We evaluate some function on R and its neighbors.

Next we choose the neighbor "most like" R. This actually involves two sets of comparisons. First we compare the function value on R with the value on each of its neighbors. Then we compare these comparisons to determine which merge of region with neighbor would produce the minimum change.

In the simplest case, the function is average intensity. The first comparison between regions is the absolute difference of the intensities of the regions. The final comparison computes the minimum of the set of absolute difference comparisons. However, the functions and the comparisons could be made far more subtle. They could involve texture, size, boundaries, etc. And they could change during the course of iterating the process.

In any case, a "link" is made between the region and the chosen neighbor. If several neighbors are equally close, several links are made. Note the link has a direction: from the region, to the neighbor. We repeat this process for each region.

We could now simply merge all linked regions. This process has several attractions. Isolated noise regions will be immediately absorbed. We may wish to follow this procedure up to some point. However, if we continue to iterate in this manner we may make some undesirable merges.

Imagine, for example, that at some stage the background has already merged into a single region, while a hammer handle remains in three segments. When we process the background region it forms a link to one of the handle regions. (It has to link to <u>some</u> neighboring region.) All of the handle regions, however, link among each other, not to the background. Clearly we would prefer to continue building up the handle to a single region, before joining with the background. This can be

By avoiding local decisions, as to which regions are significant and when to stop merging, which would lock us into a single result, we obtain a tree of regions which can be examined globally and semantically. If we get off on a wrong tack other possibilities are on hand when we return. The structure of the tree also may contain other scene information, e.g. about curvature or texture.

My own concerns center on using semantic knowledge to guide further decisions about the nature of the regions obtained. However, we do have some hope of analyzing the region structure further syntactically, and with "absolute" decision criteria, once it has been completed.

The important point is that local syntactic decisions about what is interesting, or when to stop processing, cannot be generally successful. When we are working our way up, we cannot fully judge the significance of differences in function values. An apparently large difference at one level may turn out to be just noise, or a difference between texture elements of a single surface. Looking over the whole tree with a global view, we may have some hope of making syntactic decisions on what differences are significant.

In fact proceeding back down the completed region tree we may get away with some fairly local decisions, as a good deal of global processing has gone into forming the high level nodes. For example, we have had the opportunity to "average over" texture and noise. Thus we might get good results going back down the tree and imposing an absolute criteria for judging when to "stop". E.g. if the subregions of which the region is composed are within some threshold, stop, if not consider the subregions in turn.

All problems will not average out this way, however, particularly factors that can vary among scenes, e.g. lighting. Some opportunity for global decisions will be welcome, if only to tailor thresholds to the scene.

Relative techniques can also be employed on the completed tree. For example, we can argue that R, a subregion of S, is an interesting region if the differences between R and the other subregions of S (using the measures applied in building the tree) are significantly greater than the differences between the components of R. (Of course, we have hidden another absolute decision in the word "significantly".) In fact, preliminary efforts at syntactic decisions on important regions have not been encouraging; however, we have been able to prune the tree somewhat before searching for

semantically defined regions (see below). (We can prune the tree more or less, depending on the use being made of it.)

In summary, the affinity process allows us to introduce certain relative, global, and parallel features that assist us in coping with the nonuniformities of the real visual world.

GENERATING FURTHER REGIONS

Our concern here is in generating regions as arguments for suggested statements about the scene. We can distinguish "two-dimensional" and "one-dimensional" methods. We should also note that generation normally profits from context. This can direct where to look and what to look for. Properties that the sought object should have can help us find it. Methods may also differ in what they require to constitute generation or verification of a sought region. A complete boundary description, for example, may not be necessary. These issues are discussed elsewhere. We note here briefly that the locus of application and the functions employed in the methods described below could be set by the scene context using SEER mechanisms.

Several of our methods use iterative or recursive scanning procedures to focus in on regions. Such scan techniques have attracted interest for two reasons. First they seem to offer hope of reducing the enormous scanning raster size and focusing attention where it will serve the best purpose. Second there is a natural bias in many LISP users for recursive methods.

The difficulty with the standard division into squares recursive scan is that we arrive in the final state with data base in disarray. We have a large number of bits and pieces which have been assembled in some inscrutable recursive order, and face the task of putting them together.

The techniques discussed in this chapter eliminate this drawback. Their practical usefulness remains to be proven; however, at the least they offer some mathematical interest and some potential. Again I remind you that a discussion of other region finders is contained in chapter ten; various points of similarity with methods in this chapter can be found.

This section concludes with a reference to a region finding technique developed by Tomas Lozano-Perez at the M.I.T. A.I. Laboratory. His approach is nicely designed to benefit from initial "advice". SEER is able to supply advice to his program and use it to verify regions.

TWO-DIMENSIONAL METHODS

Of course, we can run the affinity process again in a specific location, adjusting cell size and function. Other region "growing" techniques can be applied. We present two such techniques, one bottom up, another top down.

SQUARING

This method is very "region oriented". The boundary is expanded based on some appropriate region based predicate, until the true border is reached. The region grows dynamically from seed.

Start with a square of size n lying within the potential region. Label the four boundary lines "unstable". Take any unstable boundary line. Determine whether a square of size n on the outside of the line should be added to the region. (For example, compare the average light intensities of a square on either side of the line.) If so replace the boundary line with this new square. Three new boundary lines will result, all to be labelled "unstable".

If we determine not to add the new square, change the label of the line to "stable".

Clearly, as the region grows, adding a square will not always be so straightforward in bookkeeping terms. However, all this is kept quite simple actually by the use of a vector representation for boundary lines. Vectors in opposite directions cancel and flush the line.

It is an interesting exercise to determine an appropriate order for proceeding around the body in the growth process, and what patterns of growth result. It turns out that most orderly approaches get hung up in some strange case or another. In fact the most arbitrary approach seems best. We simply ask each time for any unstable line, deal with it, then ask again for any unstable line, until none remain. Note that we know precisely the boundary segments at any stage of the process.

If we wish we can half the square size when the boundary becomes stable and proceed. This involves dividing each boundary line in two and relabelling them all unstable once again. We could also allow attempts at "subtracting" squares, similar to the process in the triangulation procedure below.

DIVIDE AND CONQUER

Consider the following scheme. Follow first the horizontal bissector of the scene. If we run into something interesting, stop and track around it or employ one of the above methods on it. If we find nothing, track along the vertical bissector. Again, if we find something, stop and find its contour.

If both passes yielded nothing, we have divided the scene into four quadrants. Employ the above procedure recursively on the four quadrants.

If we did find something circumscribe a rectangle around it. Now divide the scene into rectangular pieces around the initial rectangle. Recurse on each of the empty rectangles.

Of course, this process may still allow us to miss significant groupings. To be really careful then we would have to have some understanding perhaps of the shape of the contours we find, in any case circumscribe the figures more accurately. (Shirai has suggested one method for doing this.)

Recursion stops when the block size gets sufficiently small. We can wait to recurse lower until we have dealt further with the initial objects found, of course. (The "divide and conquer" method was not programmed.)

This technique permits us to divide an area into distinct interest groupings. In [Shirai 1972], Shirai is faced with the problem of finding separate groups of blocks to begin contour processing. He uses a process which involves exhaustive scan and checking flags and thresholds. Setting up a separate coarse array for initial processing makes this more bearable.

LINEAR METHODS

We can look for a region along specific dimensions. For example, if we expect a handle in a certain location we can look for a surface flat along a line, or set of lines, parallel to the expected long axis, and smoothly curved along a "cut" across the width.

We can look at the profile of a function applied at points along a specified line; we can "push along" a line dynamically, seeking the limits of a region. In the latter case there are several strategies. For example, we can compute a function on each succeeding point and continue as long as the value stays within a threshold of an initial value. Or we can continue as long as it stays within a threshold of the previous value. This first approach can find uniform surfaces, the second allows smooth changes (e.g. smooth shading of intensity values.)

It may suffice for some purposes to verify that a plane of specified shape exists along several axes. E.g. if we have the expected center of a conjectured flat circular plane, verifying that "diameters" extend in several directions, flat and for equal distances, may suffice as an identification

(description and recognition).

AFFINITY

A function along a line presents the basic problems of noise and significance we find throughout vision. Surfaces need not be uniform. We can apply the relative techniques of the affinity process along a line to deal with these problems in one dimension, as we did initially in two. Affinity on a line merges segments and produces a "segment tree".

Affinity along a line can be used to distinguish planes and plane shapes (see below). We can obtain affinity lines that verify or segment regions. An affinity line along the length axis of a cylinder, for example, can distinguish two distinct "flat" planes, the side and an end. Several of these parallel to the length axis, together with width profiles across the side and the end, might satisfy a search for the parts of a cylinder.

The process of extending lines to determine the extent of a region along given axes can be generalized to more fully determine the extent of the region. The following algorithm can be used in that connection.

TRIANGULATION

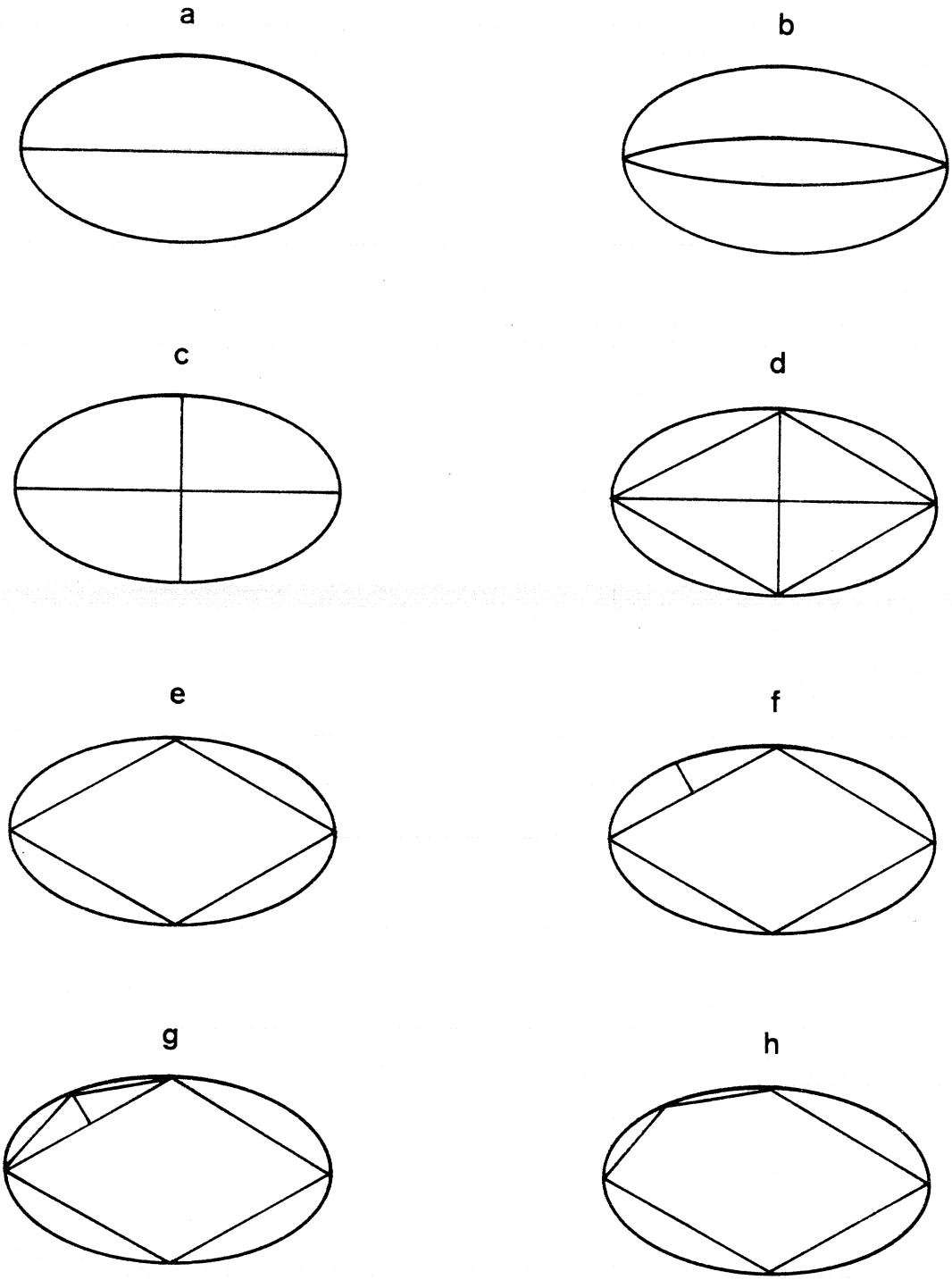
To initiate the process draw any line spanning the region (from one border to another inside the region). See figure 5 - 8 , part a. Consider this line as two lines enclosing a first approximation to the region. (This first approximation has zero area--a very thin lens shape, if you like, see part b of the figure.)

Now for every line in the approximation of the region (two at the start) we execute the following procedure. Extend a normal line outward from the midpoint until we leave the region (part c). Stop, and connect the endpoints of the line to the end of the normal (d). (Assuming these connecting lines are found to lie entirely within the region or on its boundary; if not see below.) Replace the approximating line by the two new connecting lines, in the approximating figure (e). Recurse on each line (f-g). Stop when the normal is of zero length (i. e. approximating line is true boundary segment), or below a given threshold; also stop if the total area to be added to the region by connecting to the normal is below a given threshold.

If the figure is convex we never run outside the region in adding connecting lines. (Indeed

FIGURE 5 - 8

TRIANGULATION



this is a functional definition of "convex".) If we have a concave figure the procedure is slightly more complex. Suppose we want to add connecting line AD to the approximating figure but discover that between B and C it leaves the region (figure 5 - 9 , part a). Then instead of adding AD we add AB, BC, and CD as distinct segments. We note BC is an "outside" segment and when it comes turn to recurse on it, We push inward until we hit the region again and replace BC with the appropriate connecting lines, moving in closer to the true boundary (part b).

The process is shown for two regions in figure 5 - 10 . It is a simple theorem that this procedure will converge on the true boundary of the region. Also note that at each point we have at hand the approximating boundary in a sequence of successive segments. The sequential list of boundary points is available trivially by moving transitively from one segment to the next in the data base.

In the case of polygonal regions, another theorem should establish that all lines of the region are found. That is, approximating lines are found which are segments of all the true boundary lines. (Found within some maximal number of steps, I suppose, an interesting exercise.)

In practice we can test if the extensions to intersection of alternating approximating segments lie within the region. This would allow us to avoid further recursion once we find segments of successive true boundary lines (figure 5 - 11). (Even if this process leads to new lines which are not yet true boundary lines we can use them. The process recurses on the approximating lines; the neat criss-cross pattern need not be maintained.)

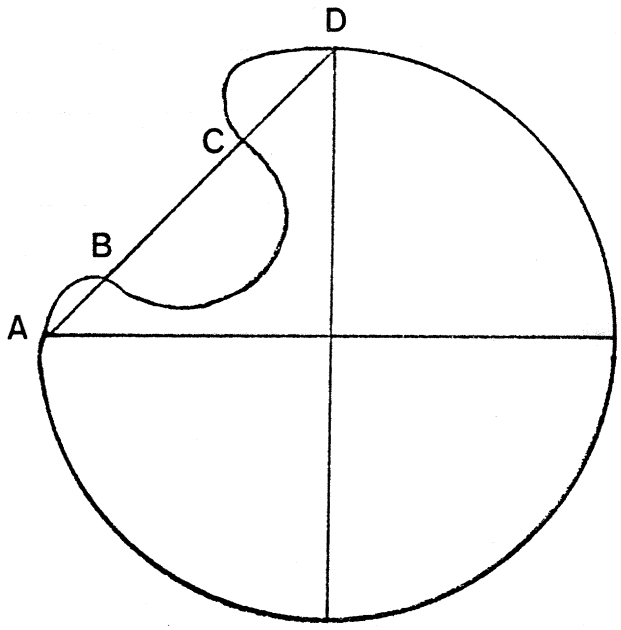
For non-polygonal regions, this process yields a polygonal approximation to any desired accuracy.

Clearly the predicates which make the crucial inside/outside decisions can be of any sort. Further their nature or support can be regulated as we recurse down. We can look at a whole triangular segment or simply track along the normal and connecting lines. If we wish, this process allows us to bite off large segments of the region and add them to the fold without scanning them at all.

FIGURE 5-9

CONCAVITY

a



b

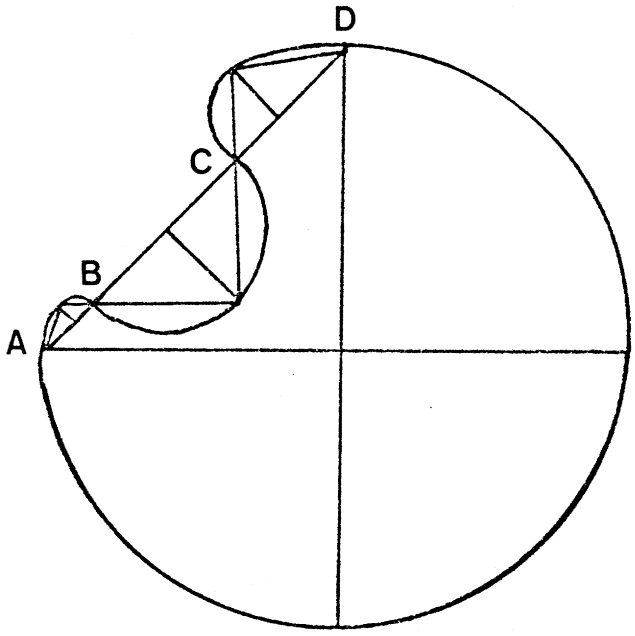


FIGURE 5 - 10

TRIANGULATION EXAMPLES

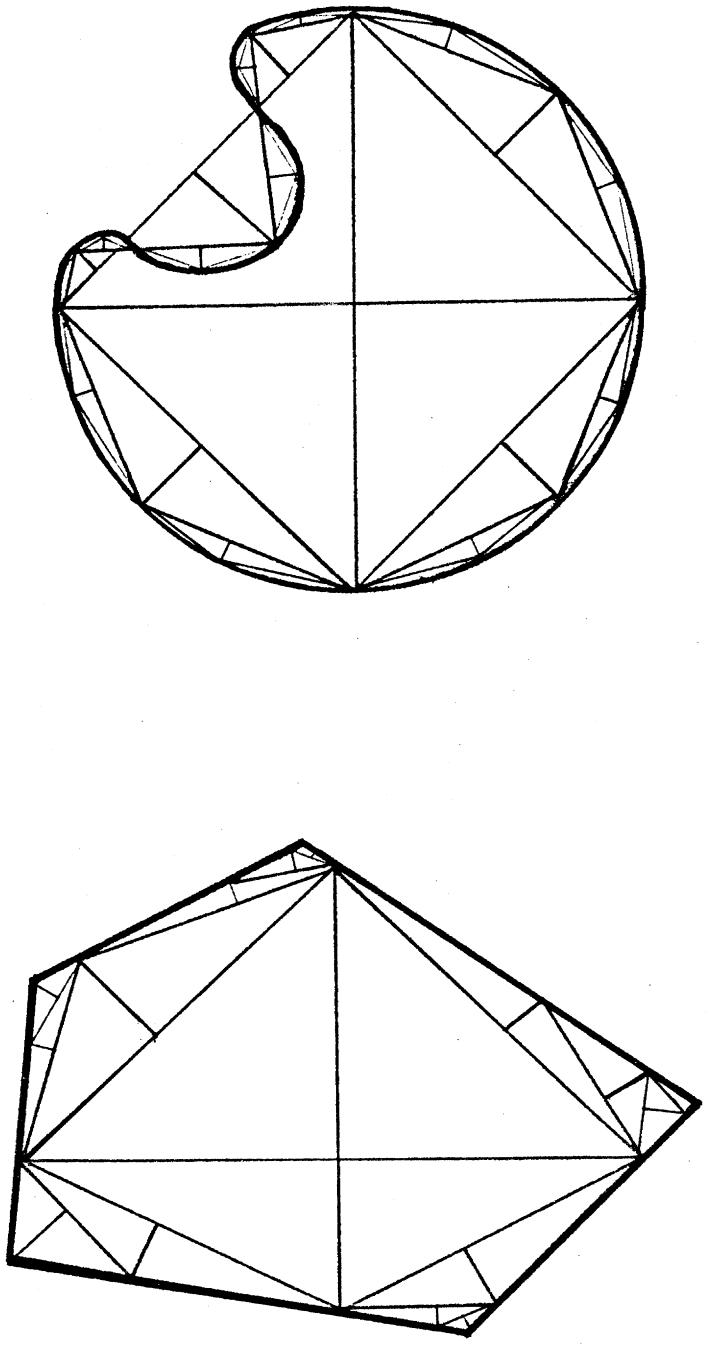
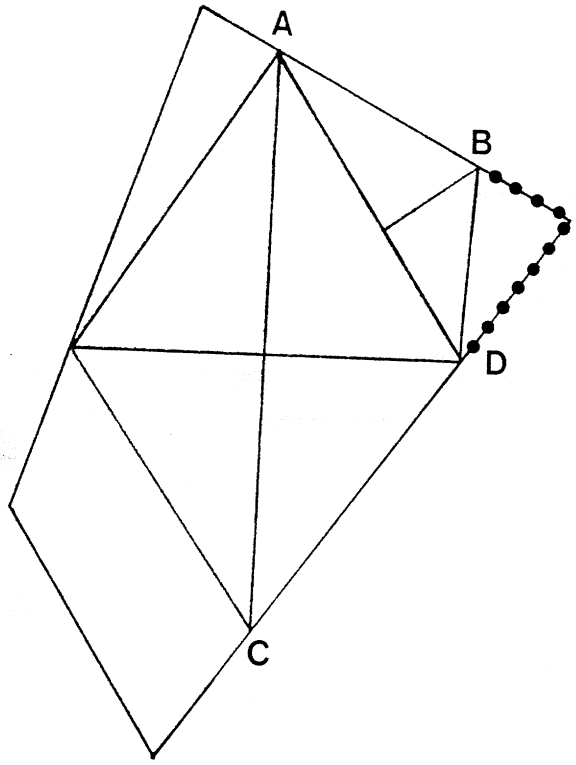


FIGURE 5-11

SUCCESSIVE BOUNDARY LINES FOUND



REGION PROFILES

Tomas Lozano-Perez has developed general techniques for defining the profile of a function applied along a line across a region [Lozano-Perez 1975]. He is able to define profiles in a "grammar". Thus he can easily accept new possibilities. SEER can tell his programs what to look for and give advice on where to look. See the case study of the ball peen hammer in the next chapter.

DESCRIPTIVE ELEMENTS

BOUNDARY SHAPE

The boundary of a region presents the usual problems of significance and noise, especially in an image created from digitized cells.

SEER employs a simple iterative process to provide a structural analysis of the boundary. It is conservative, it does not throw away information, but leaves more symbolic decisions for more intelligent or better informed processing.

Adjacent convex points on the boundary can be connected to produce a new boundary. (The boundary is originally segmented into steps corresponding to the initial cell sides in the affinity process.) This process can be iterated to produce a sequence of boundaries that define the structure of the concave incursions in the boundary. We call this a "vexity structure analysis". A similar process, "cavity structure analysis", connecting concavities, defines the protrusions in the shape. The final step in both cases produces a totally convex shape. The end result of the vexity structure analysis is a "convex hull" for the object.

The hull can be used to judge the convexity of a region. A "smoothed convex hull" can be obtained by removing minor protrusions from the convex hull. This can be used as a description of convex areas.

John Hollerbach has completed a Master's Thesis [Hollerbach 1975] which addresses many boundary shape issues. Advances in descriptive techniques, as in other areas, should easily integrate into SEER.

SURFACE SHAPE

An intensity profile along a line across a surface gives us some information on the surface shape in that direction. Running an affinity process across the line can help deal with the usual problems of variation and significance. We can move back down the affinity segment tree to look for flat planes, where subsegments are colinear. We can use a vexity structure analysis of the intensity profile to try to identify smooth convex or concave shapes.

SYMMETRY

Symmetry should prove useful as a property in itself, and as a "context" which can advise simplified techniques, e.g. for shape description and recognition. SEER uses a simple formula on its digitized boundary to measure symmetry and determine symmetric axes. The affinity process provides boundaries divided into points equidistant along the boundary. For a boundary of n points, consider the following sum taken from j = 1 to j = n/2:

S (Di-j - Di+j) , where Di is the distance of the i'th boundary point from the center of gravity of the region.

A point Pi for which the above sum is less than a threshold yields an axis of symmetry, defined by Pi and the center of gravity. We only need test n/2 consecutive boundary points, obviously. We may find that there is no symmetric axis, that the region is radially symmetric, or that there are one or more distinct axes. The point for which the above sum is minimum defines the "axis of maximum symmetry" (regardless of whether it actually qualifies as a symmetric axis).

AXES

In particular symmetry may help determine axes of orientation. There are cases where simple definitions of major axis fail, e.g. longest diameter or linear fit of boundary points. The axis of maximum symmetry, as determined by the above formula, offer some hope of capturing our intuitive feeling here. However, I have problems with this formalization as well. Fortunately, this only serves to provide me with an example of the need for specialized techniques and case by case definitions. In the context of a "bar", our most interesting shape at the moment, an axis can be more easily defined.

PROPERTIES AND RELATIONSHIPS

SEER has, of course, a set of the obvious properties, e.g. area, center of gravity. There are also a set of relationships, like "touching", "perpendicular to".

VISUAL PREDICATES

I would like to develop the concept of visual predicates like "in there" that we may be overlooking in our natural attempts to verbalize visual descriptions. This is only a vague concept at present. A system predicate that leads in this direction perhaps, is one like "cover-end" which is used in determining a face for the hammer head.

We also have utility functions that help us change our axis of orientation easily to function within the coordinate system defined by the axes of individual objects.

6

EXAMPLES

I have taken a "case study" approach to developing SEER's competence. I have tried to learn how to use visual knowledge. I have studied how pieces of knowledge can cooperate to perform specific recognition tasks and deal with specific visual problems. I have not tried to restrict the outside domain, or allow variety along only a single axis of visual complexity. Our concern is dealing with visual reality. A given scene represents a realistic input, not an idealized one. However, for any one scene there will be specific problems that can be used as case studies.

The results of these studies may generalize to produce general purpose techniques. Or we may develop a set of special purpose techniques, along with SEER's facility for using them. Again, we emphasize the study of interaction and cooperation of results and processes. SEER is particularly appropriate for handling collections of alternatives when a general purpose approach is not available or a context dependent alternative is more efficient. SEER facilitates and integrates both the incremental programming of such methods, and their dynamic application in response to processing results. Our case studies inform our evaluation and design of the SEER system elements, the priority system, the relevance link options, etc. Ultimately, we may be able to implement techniques from other work in AI [Sussman 1973] [Goldstein 1974] to help SEER bootstrap itself along in expanding its area of competence.

The following examples illustrate SEER at work on several of the problem areas identified earlier. The motivation for these case studies can come from two directions. On the one hand, we can add new scenes and let the problems which arise motivate expansion of the GK or refinement of the system. I, of course, also sought out case studies that could provide examples to illustrate various points in the thesis.

These examples illustrate some of the major problem areas we face. I do not claim to solve any of these problems here, but to demonstrate a system which is useful for attacking such problems. We begin with two scenes involving different hammers, which motivate different recognition paths.

Then a simple occlusion illustrates further how SEER naturally suggests useful approaches to visual difficulties.

A SLEDGE HAMMER ON A DARK BACKGROUND

The first scene is shown in figure 6 - 1 . The dark, mottled head very nearly disappears into the background. A close up of the head is shown in figure 6 - 2 . You can see the shadow of the head on the handle.

The pictures are rather limited by the output medium, but, on the other hand, they do illustrate the problems faced by, e.g. a naive threshholding approach to the scene. The pictures (produced by programs due to Tim Finin and Tomas Lozano-Perez) are normally formed by making a rough scan of the scene to find maximum and minimum light intensity values, and then threshholding in between to assign points one of sixteen printing "characters". You can see how such a process leaves the head virtually invisible, blending into the background. In the close up of the head I forced the threshholds to concentrate on the dark end of the scale. Thus we lost all detail in the handle, but additional detail appears for the head. However, now we see just how "crufty" and non uniform the hammer head really is. There is a danger of separating it into too many meaningless regions (and we still have a problem blending into the background). We have again the basic issue of determining significance. The nature of the hammer, the background, the lighting, and the input camera device, all contribute to these problems.

Nevertheless the affinity process is still able to give us a good start on this scene. A crudely pruned affinity tree for this scene is shown in figure 6 - 3 . The regions corresponding to the handle, head and face are all in the tree and noted in the figure. Figure 6 - 4 shows these regions superimposed on the scene. The head is found, but joins with the background before the handle, which stands out higher in the tree. The seed suggestion for region 278 is investigated, and successes and proposals quickly move up to a basic bar shape recognition. This suggests a possible handle or head. Both have high interest, the handle is easier to pursue. We explore it, success proposes we look for an encompassing hammer, and advises how we might find the required head. The head region is found in the tree; the proper relationship to the handle holds, and the face is sought. After some false

FIGURE 6 - 1
SLEDGE HAMMER ON DARK BACKGROUND

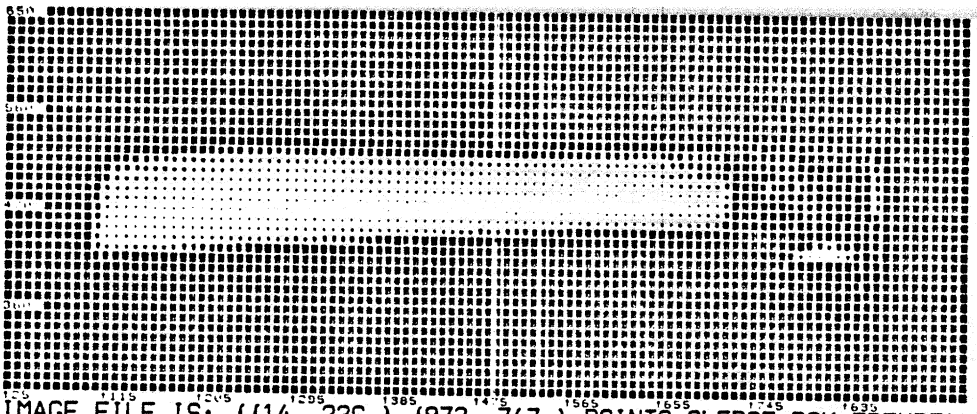


IMAGE FILE IS: ((14. 236.) (973. 747.) POINTS SLEDGE DSK FREUDE)
SELECTED WINDOW IS: ((25. 300.) (950. 650.))
RESOLUTION IS: 9.0

***** LEGEND *****

```
     -  399. :
399.  -  415. :
415.  -  431. :
431.  -  447. :
447.  -  463. :
463.  -  479. :
479.  -  495. :
495.  -  511. :
511.  -  527. :
527.  -  543. :
543.  -  559. :
559.  -  575. :
575.  -  591. :
591.  -  607. :
607.  -  623. :
623.  -       :
```

FIGURE 6 - 2
CLOSE UP OF SLEDGE HEAD

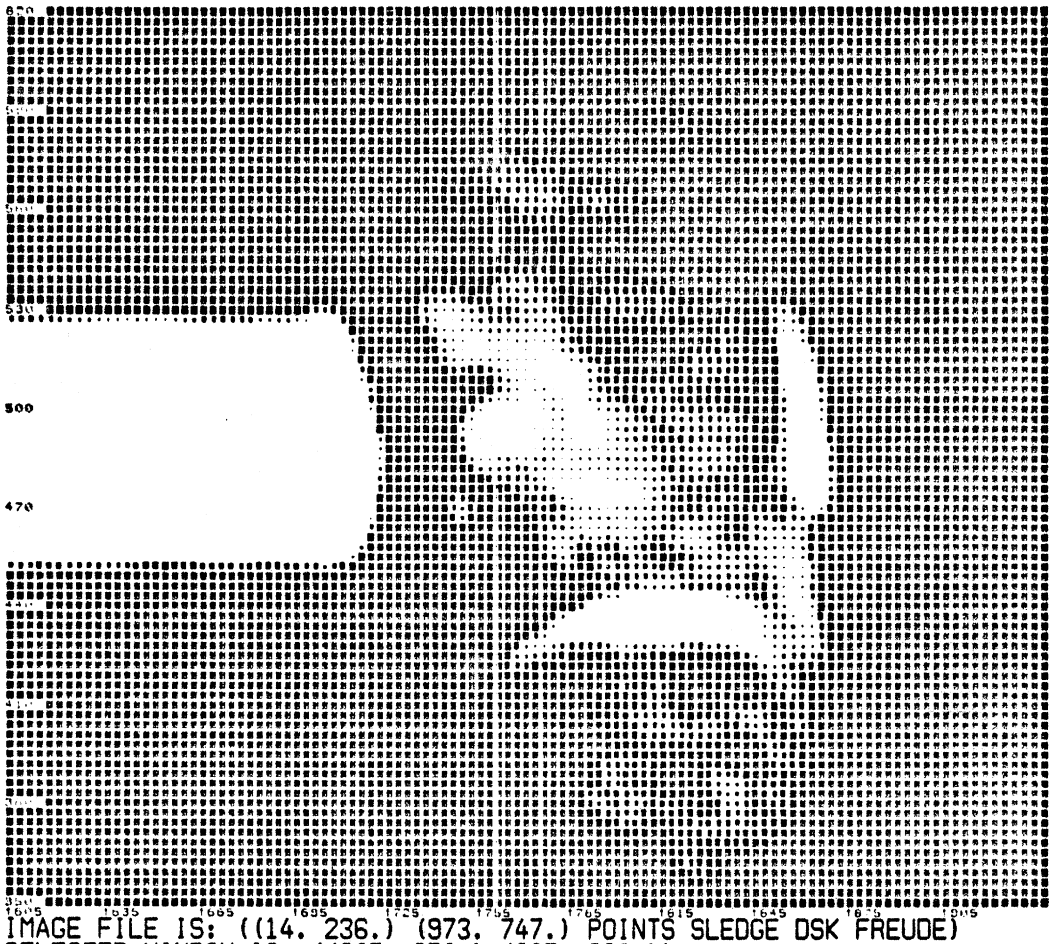


IMAGE FILE IS: ((14. 236.) (973. 747.) POINTS SLEDGE DSK FREUDE)
SELECTED WINDOW IS: ((605. 350.) (935. 620.))
RESOLUTION IS: 3.0

***** LEGEND *****

```
     -  586. :
586.  -  589. :
589.  -  592. :
592.  -  595. :
595.  -  598. :
598.  -  601. :
601.  -  604. :
604.  -  607. :
607.  -  610. :
610.  -  613. :
613.  -  616. :
616.  -  619. :
619.  -  622. :
622.  -  625. :
625.  -  628. :
628.  -       :
```

FIGURE 6 - 3

CRUDE AFFINITY TREE FOR SLEDGE

```
R279 782. 610.
   R278 123. 467. <HANDLE>
      R242 7. 583.
      R277 116. 460.
         R237 6. 583.
         R276 110. 453.
            R265 29. 405.
               R261 17. 400.
                  R256 7. 412.
                  R239 10. 392.
               R247 12. 412.
            R271 74. 462.
               R268 41. 452.
                  R266 23. 464.
                     R262 16. 456.
                     R240 7. 482.
                  R255 18. 437.
                     R225 11. 439.
                     R248 7. 433.
               R260 33. 474.
                  R254 24. 480.
                     R249 14. 485.
                        R241 8. 477.
                        R224 6. 496.
                     R235 10. 474.
                  R253 9. 456.
   R217 7. 631.
   R272 41. 606. <HEAD>
      R259 6. 591.
      R263 29. 616.
         R246 9. 622. <FACE>
         R257 20. 614.
            R245 6. 606.
            R252 14. 617.
```

FIGURE 6 - 4
REGIONS R278, R272 AND R246

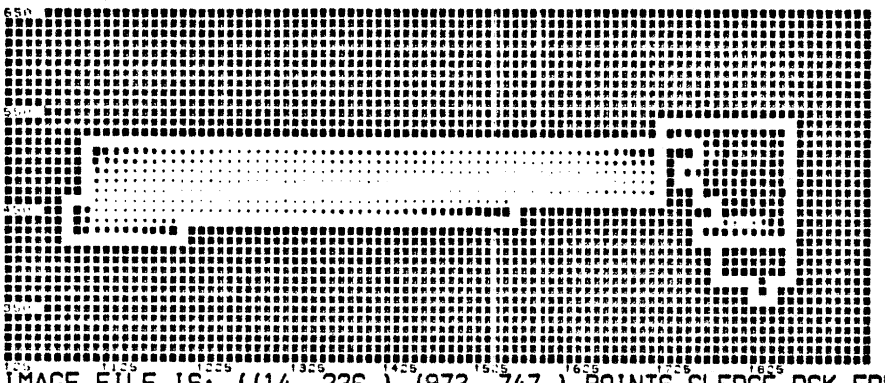


IMAGE FILE IS: ((14. 236.) (973. 747.) POINTS SLEDGE DSK FREUDE)
SELECTED WINDOW IS: ((25. 300.) (950. 650.))
RESOLUTION IS: 10.0

***** LEGEND *****

```
     -  405. :
405.  -  420. :
420.  -  435. :
435.  -  450. :
450.  -  465. :
465.  -  480. :
480.  -  495. :
495.  -  510. :
510.  -  525. :
525.  -  540. :
540.  -  555. :
555.  -  570. :
570.  -  585. :
585.  -  600. :
600.  -  615. :
615.  -       :
```

starts the correct region is found and the recognition completed.

A BALL PEEN HAMMER ON A DIRTY, WOOD GRAINED WORK BENCH

See figure 6 - 5 . The thin, sharply angled, wood grained handle is difficult for the affinity pass to conglomerate with a moderately sized square cell grid. However, the head does stand out well. A close up of the head is shown in figure 6 - 6 . (I saved some time and space by running the affinity pass over a portion of the scene, shown in figure 6 - 7 with the head region found superimposed, but I could have used the whole scene as in the previous example. The mottled background does make for more regions in the lower part of the tree; however, there is considerable background in the portion of the scene processed, and the head was still found. It would perhaps be good to make the affinity process two stage eventually: a very crude pass to find interesting areas, then a finer one in these areas.)

The head is region R196 in the affinity tree, as shown in the previous figure. Again we quickly find R196 to be a bar, and propose head and handle. In an earlier version of the GK, a bar shape was all that was required for a handle and R196 would have been established as an hp1, hammer-part-handle. A head to go with it would have then been sought in vain. We described this flow in chapter one. More recently while working on another scene the GK was improved so that more of a long and thin shape is required for a hp1 recognition. Thus the hp1 conjecture fails. Neither is R196 obviously part of an occluded handle at this point. The head conjecture is tried. A face is found; the region is shown in the less pruned affinity tree in figure 6 - 8 . The head conjecture suceeds, suggesting a hammer. An advised method for finding the handle, which employs Lozano's region profile tracking, is pursued. The head provides this method with a starting point and direction for testing, and the handle conjecture specifies the profile being sought. A region is found, a few more tests are made, and it succeeds as the sought handle, completing the head recognition.

A HAMMER OCCLUDED BY A SAW BLADE

This scene is shown in figure 6 - 9 . It is not quite a real scene; it was created from the sledge hammer in the first case study, by using an input function that did not return the stored picture

FIGURE 6 - 5
BALL PEEN HAMMER ON WORK BENCH

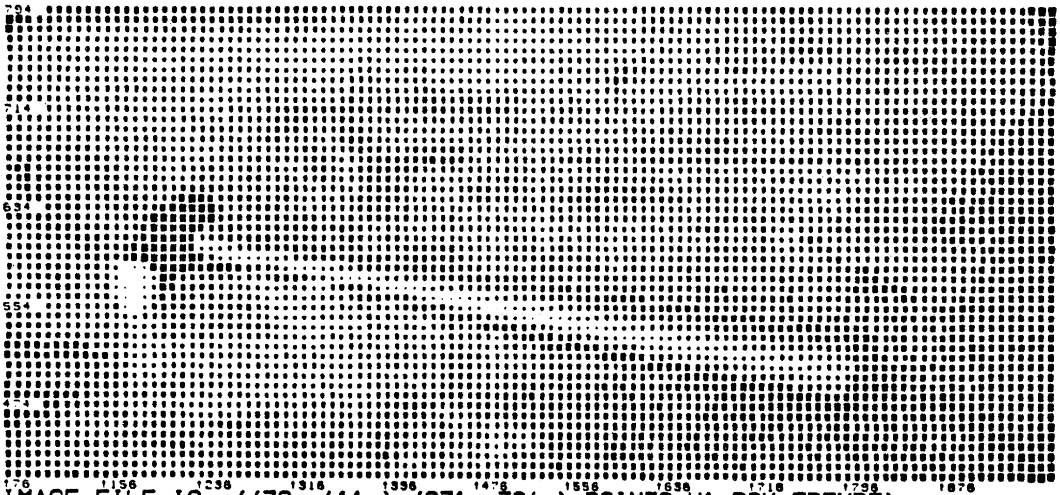


IMAGE FILE IS: ((76. 411.) (971. 794.) POINTS H1 DSK FREUDE)
SELECTED WINDOW IS: ((76. 411.) (971. 794.))
RESOLUTION IS: 8.0
***** LEGEND *****

```
     -  280. :
280.  -  299. :
299.  -  318. :
318.  -  337. :
337.  -  356. :
356.  -  375. :
375.  -  394. :
394.  -  413. :
413.  -  432. :
432.  -  451. :
451.  -  470. :
470.  -  489. :
489.  -  508. :
508.  -  527. :
527.  -  546. :
546.  -       :
```

FIGURE 6 - 6
CLOSE UP OF BALL PEEN HEAD

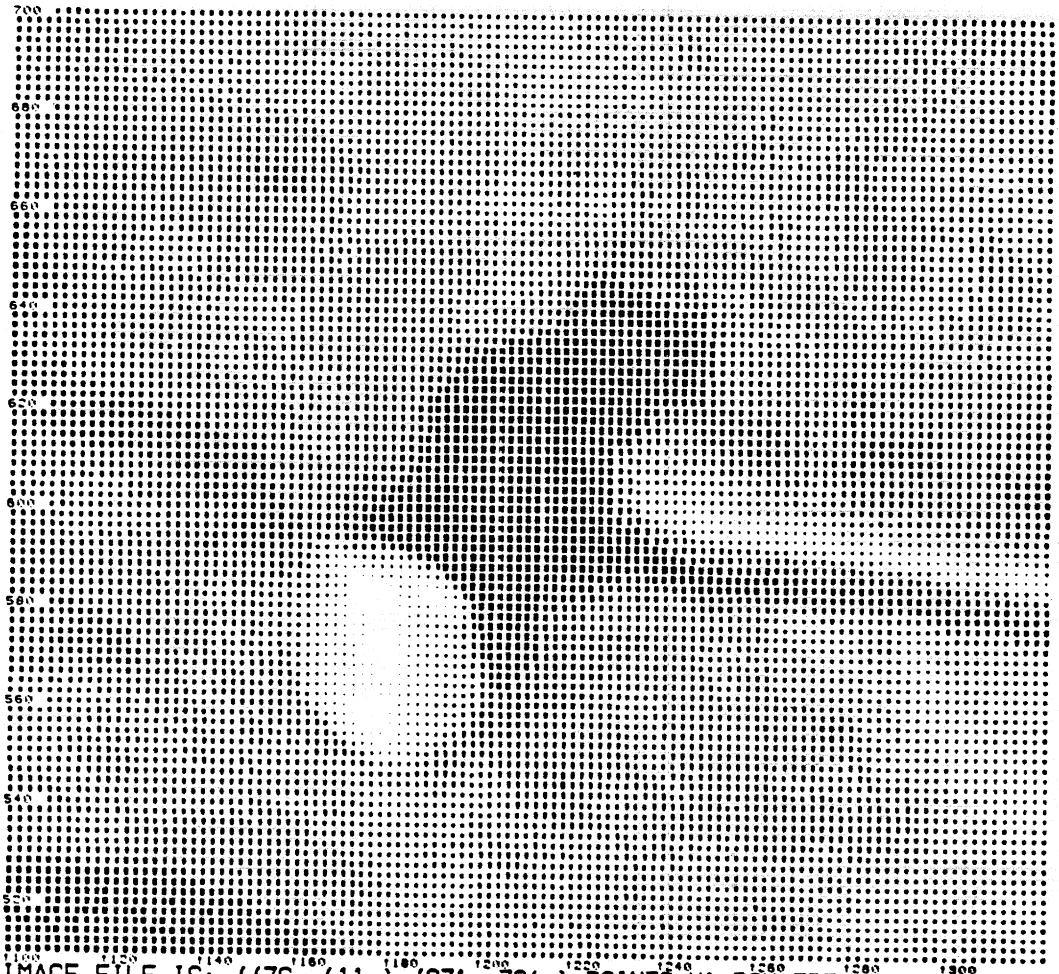


IMAGE FILE IS: ((76. 411.) (971. 794.) POINTS H1 DSK FREUDE)
SELECTED WINDOW IS: ((100. 510.) (320. 700.))
RESOLUTION IS: 2.0

***** LEGEND *****

	-	282.	:
282.	-	303.	:
303.	-	324.	:
324.	-	345.	:
345.	-	366.	:
366.	-	387.	:
387.	-	408.	:
408.	-	429.	:
429.	-	450.	:
450.	-	471.	:
471.	-	492.	:
492.	-	513.	:
513.	-	534.	:
534.	-	555.	:
555.	-		:

FIGURE 6 - 7
REGION 196

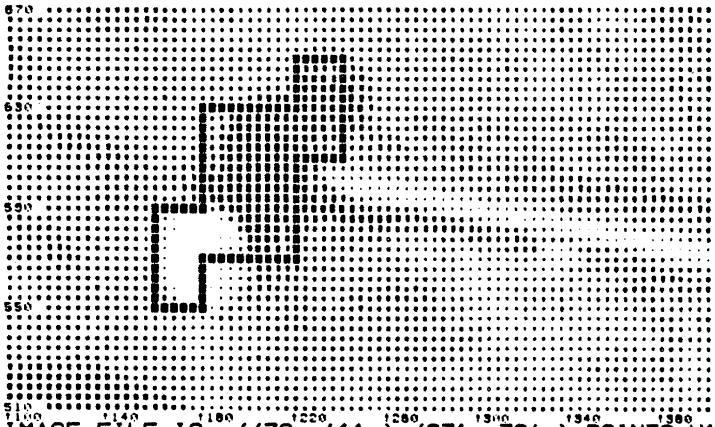


IMAGE FILE IS: ((76. 411.) (971. 794.) POINTS H1 DSK FREUDE)
SELECTED WINDOW IS: ((100. 510.) (400. 670.))
RESOLUTION IS: 4.0

***** LEGEND *****

```
     -  321. :
321.  -  341. :
341.  -  361. :
361.  -  381. :
381.  -  401. :
401.  -  421. :
421.  -  441. :
441.  -  461. :
461.  -  481. :
481.  -  501. :
501.  -  521. :
521.  -  541. :
541.  -  561. :
561.  -  581. :
581.  -  601. :
601.  -       :
```

FIGURE 6 - 8

FINE AFFINITY TREE FOR BALL PEEN HAMMER

```
R199 120. 448.
   R134 3. 423.
   R112 3. 401.
   R16 3. 451.
   R196 10. 480. <HEAD>
      R145 3. 307. <FACE>
      R169 7. 554.
         R144 3. 561.
         R142 3. 556.
   R140 6. 451.
      R51 3. 454.
   R185 6. 444.
      R158 3. 474.
      R156 3. 414.
   R191 14. 467.
      R181 12. 463.
         R175 10. 461.
            R167 8. 459.
               R72 3. 460.
   R147 3. 448.
   R120 3. 454.
   R104 3. 450.
```

FIGURE 6 - 9
OCCLUDED SLEDGE HAMMER

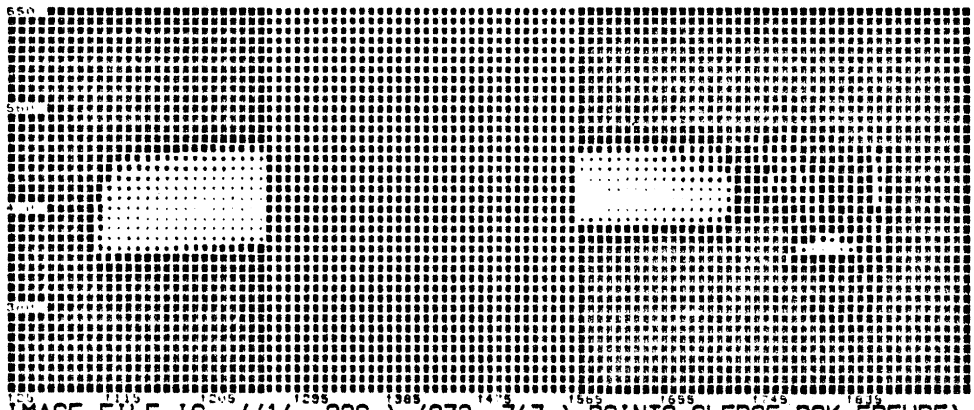


IMAGE FILE IS: ((14. 236.) (973. 747.) POINTS SLEDGE DSK FREUDE)
SELECTED WINDOW IS: ((25. 300.) (950. 650.))
RESOLUTION IS: 9.0

***** LEGEND *****

```
     -  399. :
399.  -  415. :
415.  -  431. :
431.  -  447. :
447.  -  463. :
463.  -  479. :
479.  -  495. :
495.  -  511. :
511.  -  527. :
527.  -  543. :
543.  -  559. :
559.  -  575. :
575.  -  591. :
591.  -  607. :
607.  -  623. :
623.  -       :
```

point in the central area but rather a value to simulate the occlusion. Frankly, this approach was used as an expedient to get double duty from a picture in the face of a frustrating input device that was slowly dying during the latter stages of my research. On the other hand, there is an argument for employing this technique to a limited degree. We store scenes on disk for repeatability and comparison of experiments. This approach allows us to go a step further and incrementally alter a previously studied scene by adding specified difficulties. In any case, this example is not, of course, intended as a solution to the occlusion problem, but again as merely another demonstration of how SEER works and how it can be useful for dealing with problems. In particular, we see how the suggestion mechanisms, and the organization of our GK into muturally helpful egocentric elements and alternative methods, naturally implements one approach to occlusion. (The handle is also genuinely occluded at the end by a dark cloth, and was in the first scene, simply to help fit the long object into the picture.)

Figure 6 - 10 is a crude affinity tree, and figure 6 - 11 shows how the indicated regions, representing the key elements of the scene, appear in the scene. The first region to be analyzed, R151, is a piece of the handle. It is not the right dimensions for a hammer handle possibility and there are no other occluded parts to link with. We try it out as a hammer head; this fails also. We next look at R149. This is not an hp1 either. However, the success of the bar sides search coupled with the failure of the dimensions test suggest that R149 may be an occluded handle. We look for other pieces and find R151. They seem to go together and SEER can create a new region from them, to serve as a possible hammer handle. This hp1 now can proceed to suggest and find the rest of the hammer as in the first scene. (The affinity tree for the head is different from that in the first case study, a bigger cell size was used here for one thing. SEER should be able to handle differences produced by different affinity passes.)

In the appendix of my Ph.D. thesis [Freuder 1975], I presented recognition paths and a series of annotated cycle traces and investigation state snapshots for each of the recognitions discussed. These provided a more detailed picture of the processing and examples of many of the principles and mechanisms we have discussed earlier, but I can only recommend them to those interested in a great deal of detail.

FIGURE 6 - 10

CRUDE AFFINITY TREE FOR OCCLUDED SLEDGE

```
R158 646. 608.
   R151 23. 485. <LEFT PIECE OF HANDLE>
      R131 5. 599.
      R147 18. 454.
         R143 12. 437.
         R142 6. 488.
   R149 16. 495. <RIGHT PIECE OF HANDLE>
      R126 6. 618.
      R141 10. 421.
   R140 8. 583.
   R154 35. 618. <HEAD>
      R152 20. 611.
         R134 7. 623. <FACE>
         R145 12. 603.
            R123 5. 594.
            R133 6. 605.
      R144 15. 627.
         R122 7. 634.
         R138 8. 621.
```

FIGURE 6 - 11
REGIONS R151, R149, R154, R134

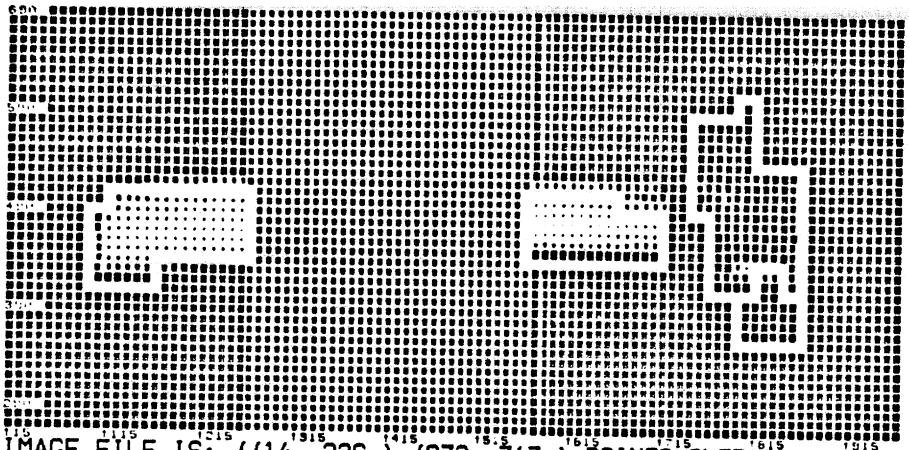


IMAGE FILE IS: ((14. 236.) (973. 747.) POINTS SLEDGE DSK FREUDE)
SELECTED WINDOW IS: ((15. 265.) (965. 690.))
RESOLUTION IS: 10.0

***** LEGEND *****

```
     -  368. :
368.  -  386. :
386.  -  404. :
404.  -  422. :
422.  -  440. :
440.  -  458. :
458.  -  476. :
476.  -  494. :
494.  -  512. :
512.  -  530. :
530.  -  548. :
548.  -  566. :
566.  -  584. :
584.  -  602. :
602.  -  620. :
620.  -       :
```

CYCLE TRACES PRESENT A MORE DETAILED PICTURE

I will present here the second example above in some greater detail, including several cycle traces. When changes or additions are made in SEER the processing flow for a scene may change, and as indicated above different successful flows were obtained on this scene. The new version is more efficient, but the older one is interesting and I present it here. I will pick out several instructive cycles in the processing of the problem scene. (Figure 6 - 12 illustrates the basic operation of a few of these in the simplified graphical format.) First we view a succession of consecutive cycles early in the analysis to get a feeling for some of the processing flow. Then we jump quickly through some of the highlights of the recognition process. We include the cycle trace and a picture of the PK state at the start of the cycle for each cycle discussed. These were printed by the program as the cycle progressed.

Here is a summary of the recognition:

Region 196 stands out in the scene. First it is thought that it might be a hammer handle, but a head cannot be found to go with it to complete a hammer recognition. Along the way another suggestion was made, however, that R196 might be a hammer head. SEER goes back and investigates this suggestion. A striking face is found and a tentative head recognition is made, which again suggests a hammer. This time, with advice from the head, a handle is found, that goes along with the head and completes the hammer recognition.

CYCLE 12: This was used as an illustration in chapter three. We are moving downward to pursue the investigation of D15, which represents the results needed to compute the smooth convex hull (SCH) of region R196.

CYCLE 13: We continue to explore downward.

CYCLE 14: An execution succeeds and is exploited. Results propagate back upwards.

CYCLE 15: Another execution. This cycle was also used as an illustration in chapter three. The success of D16 is the last result needed to complete the investigation of D15. D15 succeeds and proposes a further investigation by initiating a new datum, D19. D19 is compared with the already present investigations and chosen as the new top priority investigation. The exploitiation of D9 also affects datums which are not in the D15 investigation, but are elsewhere in the PK: D8 and D6.

FIGURE 6-12

CYCLES C12 - C15

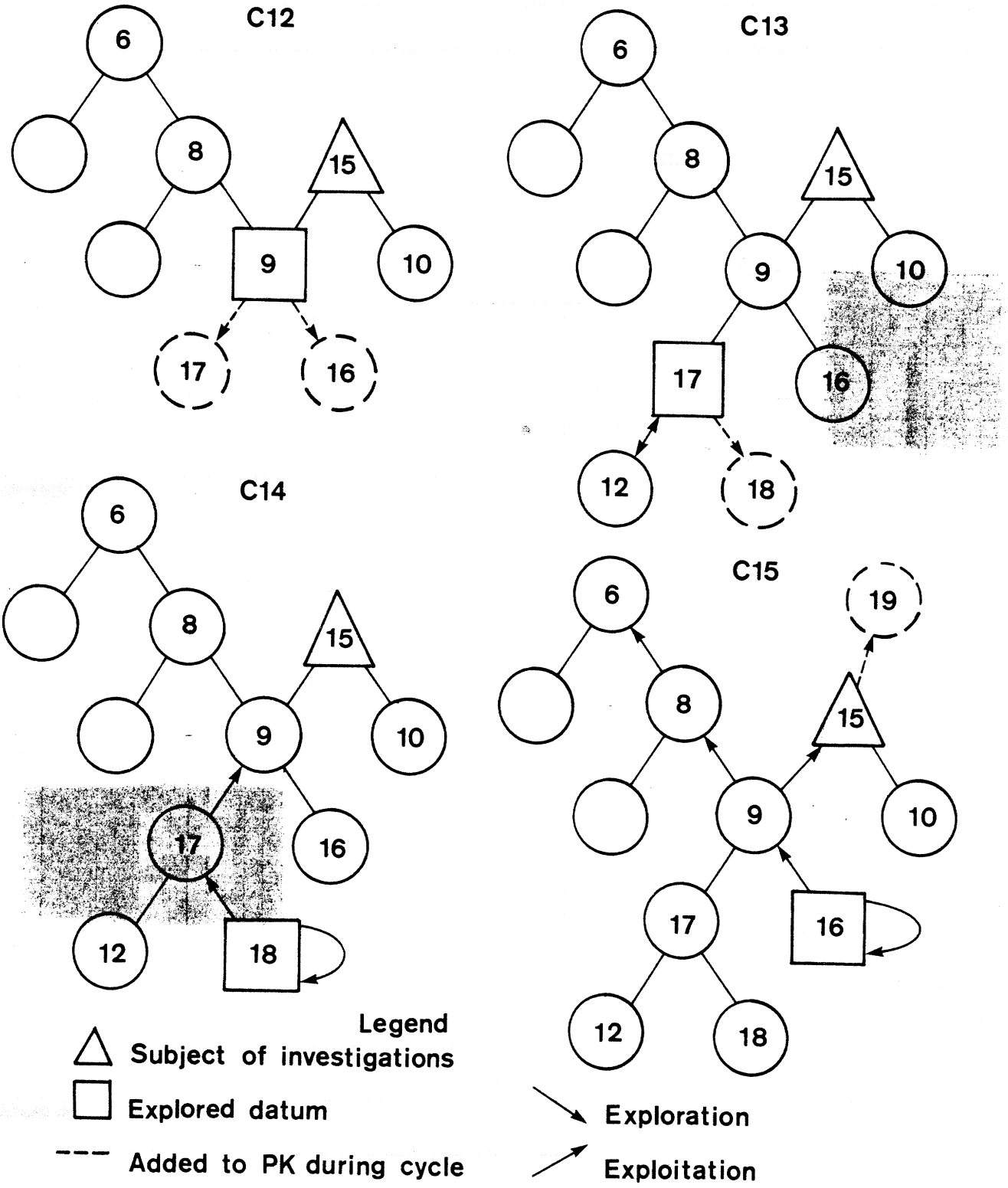


FIGURE 6 - 13

```
CYCLE C12
INVESTIGATING D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
AT: D9 (CONVEX-HULL R196)


STATE OF INVESTIGATION
OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
AT START OF CYCLE C12

-> # D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) # CONJECTURE
 -> $$ D9 (CONVEX-HULL R196) $$ CONJECTURE <- (D8)
 -> D10 (AREA R196) SUCCESS <- (D8)

EXPLORE D9
---> * D17 (VEX-STR-ANA R196)
     ENTERED: (PUI SI SE A)
     FIRED:
---> * D16 (CONVEX-HULL-COMPUTATION R196)
     ENTERED: (PUI SI SE A)
     FIRED:
```

FIGURE 6 - 14

```
                    CYCLE C13
                    INVESTIGATING D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT: D17 (VEX-STR-ANA R196)


                    STATE OF INVESTIGATION
                    OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT START OF CYCLE C13

-> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) CONJECTURE
 -> # D9 (CONVEX-HULL R196) # CONJECTURE <- (D8)
  -> $$ D17 (VEX-STR-ANA R196) $$ CONJECTURE
  -> D16 (CONVEX-HULL-COMPUTATION R196) CONJECTURE
 -> D10 (AREA R196) SUCCESS <- (D8)

EXPLORE D17
---> D12 (O-BDRY R196)
     ENTERED: (PUI SI SE A)
     FIRED: SE
     DIFF -> 0.64 LIKE -> 1.0
---> * D18 (VEX-STR-ANA-COMPUTATION R196)
     ENTERED: (PUI SI SE A)
     FIRED:
```

FIGURE 6 - 15

```
                    CYCLE C14
                    INVESTIGATING D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT: D18 (VEX-STR-ANA-COMPUTATION R196)


                    STATE OF INVESTIGATION
                    OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT START OF CYCLE C14

-> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) CONJECTURE
 -> D9 (CONVEX-HULL R196) CONJECTURE <- (D8)
  -> # D17 (VEX-STR-ANA R196) # CONJECTURE
   -> D12 (C-BDRY R196) SUCCESS <- (D10)
   -> $$ D18 (VEX-STR-ANA-COMPUTATION R196) $$ CONJECTURE
  -> D16 (CONVEX-HULL-COMPUTATION R196) CONJECTURE
 -> D10 (AREA R196) SUCCESS <- (D8)

EXPLORE D18
EXECUTED: 1.51 SECONDS 2.93 REALTIME
STATUS -> SUCCESS
EXPLOIT D18
<--- D17 (VEX-STR-ANA R196)
     ENTERED: NIL
     FIRED: SE
     STATUS -> SUCCESS
     EXPLOIT D17
     <--- D9 (CONVEX-HULL R196)
          ENTERED: NIL
          FIRED: SE
          DIFF -> 0.03 LIKE -> 1.0
```

FIGURE 6 - 16

```
                    CYCLE C15
                    INVESTIGATING D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT: D16 (CONVEX-HULL-COMPUTATION R196)


                    STATE OF INVESTIGATION
                    OF D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
                    AT START OF CYCLE C15

-> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) CONJECTURE
 -> D9 (CONVEX-HULL R196) CONJECTURE <- (D8)
  -> D17 (VEX-STR-ANA R196) SUCCESS
  -> $$ D16 (CONVEX-HULL-COMPUTATION R196) $$ CONJECTURE
 -> D10 (AREA R196) SUCCESS <- (D8)

EXPLORE D16
EXECUTED: 0.05 SECONDS 0.06 REALTIME
STATUS -> SUCCESS
EXPLOIT D16
<--- D9 (CONVEX-HULL R196)
     ENTERED: NIL
     FIRED: SE
     STATUS -> SUCCESS
     EXPLOIT D9
     <--- D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
          ENTERED: NIL
          FIRED: SE
          -INVES
          STATUS -> SUCCESS
          EXPLOIT D15
          <--- * D19 (SMOOTHED-CONVEX-HULL R196)
               +INVES
               ENTERED: (PUI SI SE A)
               FIRED: SE
               DIFF -> 1.0 LIKE -> 1.0
               PRIORITY -> 0.99
     <--- D8 (CONVEX-NEEDS R196)
          ENTERED: NIL
          FIRED: SE
          STATUS -> SUCCESS
          EXPLOIT D8
          <--- D6 (CONVEX R196)
               ENTERED: NIL
               FIRED: SE
               DIFF -> 0.17

NEW TOP PRIORITY INVESTIGATION: D19 (SMOOTHED-CONVEX-HULL R196)
```

FIGURE 6 - 17

```
                    CYCLE C16
                    INVESTIGATING D19 (SMOOTHED-CONVEX-HULL R196)
                    AT: D19 (SMOOTHED-CONVEX-HULL R196)


                    STATE OF INVESTIGATION
                    OF D19 (SMOOTHED-CONVEX-HULL R196)
                    AT START OF CYCLE C16

-> $$ D19 (SMOOTHED-CONVEX-HULL R196) $$ CONJECTURE
 -> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) SUCCESS

EXPLORE D19
---> * D20 (SMOOTHED-CONVEX-HULL-COMPUTATION R196)
     ENTERED: (PUI SI SE A)
     FIRED:
---> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196)
     ENTERED: NIL
     FIRED:
```

FIGURE 6 - 18

```
                    CYCLE C17
                    INVESTIGATING D19 (SMOOTHED-CONVEX-HULL R196)
                    AT: D20 (SMOOTHED-CONVEX-HULL-COMPUTATION R196)


                    STATE OF INVESTIGATION
                    OF D19 (SMOOTHED-CONVEX-HULL R196)
                    AT START OF CYCLE C17

-> # D19 (SMOOTHED-CONVEX-HULL R196) # CONJECTURE
 -> $$ D20 (SMOOTHED-CONVEX-HULL-COMPUTATION R196) $$ CONJECTURE
 -> D15 (SMOOTHED-CONVEX-HULL-NEEDS R196) SUCCESS

EXPLORE D20
EXECUTED: 0.38 SECONDS 4.2 REALTIME
STATUS -> SUCCESS
EXPLOIT D20
<--- D19 (SMOOTHED-CONVEX-HULL R196)
     ENTERED: NIL
     FIRED: SE
     -INVES
     STATUS -> SUCCESS
     EXPLOIT D19
     <--- * D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196)
          +INVES
          ENTERED: (PUI SI SE A)
          FIRED: SE
          DIFF -> 0.96
          PRIORITY -> 0.51

NEW TOP PRIORITY INVESTIGATION: D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196)
```

FIGURE 6 - 19

```
                    CYCLE C18
                    INVESTIGATING D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196)
                    AT: D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196)


                    STATE OF INVESTIGATION
                    OF D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196)
                    AT START OF CYCLE C18

-> $$ D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196) $$ CONJECTURE
 -> D19 (SMOOTHED-CONVEX-HULL R196) SUCCESS

EXPLORE D21
---> D6 (CONVEX R196)
     ENTERED: (PUI SI SE A)
     FIRED:
---> D19 (SMOOTHED-CONVEX-HULL R196)
     ENTERED: NIL
     FIRED:
```

FIGURE 6 - 20

```
                    CYCLE C19
                    INVESTIGATING D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196)
                    AT: D7 (CONVEX-COMPUTATION R196)


                    STATE OF INVESTIGATION
                    OF D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196)
                    AT START OF CYCLE C19

-> # D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196) # CONJECTURE
 -> D6 (CONVEX R196) CONJECTURE <- (D2)
  -> D8 (CONVEX-NEEDS R196) SUCCESS
  -> $$ D7 (CONVEX-COMPUTATION R196) $$ CONJECTURE
 -> D19 (SMOOTHED-CONVEX-HULL R196) SUCCESS

EXPLORE D7
EXECUTED: 0.21 SECONDS 0.33 REALTIME
STATUS -> SUCCESS
EXPLOIT D7
<--- D6 (CONVEX R196)
     ENTERED: NIL
     FIRED: SE
     STATUS -> SUCCESS
     EXPLOIT D6
     <--- D21 (SMOOTHED-CONVEX-HULL-AS-DESCRIPTION R196)
          ENTERED: NIL
          FIRED: SE
          -INVES
          STATUS -> SUCCESS
          EXPLOIT D21
          <--- * D22 (HAS-BAR-SIDES-METHOD1-NEEDS R196)
               +INVES
               ENTERED: (PUI SI SE A)
               FIRED: SE
               DIFF -> 0.02
               PRIORITY -> 16.93
     <--- D2 (SEED R196)
          ENTERED: NIL
          FIRED: SE
          -INVES
          STATUS -> SUCCESS
          EXPLOIT D2

NEW TOP PRIORITY INVESTIGATION: D22 (HAS-BAR-SIDES-METHOD1-NEEDS R196)
```

FIGURE 6 - 21

```
                    CYCLE C33
                    INVESTIGATING D41 (BAR-DIMENSIONS R196)
                    AT: D42 (BAR-DIMENSIONS-COMPUTATION R196)


                    STATE OF INVESTIGATION
                    OF D41 (BAR-DIMENSIONS R196)
                    AT START OF CYCLE C33

-> # D41 (BAR-DIMENSIONS R196) # CONJECTURE
 -> $$ D42 (BAR-DIMENSIONS-COMPUTATION R196) $$ CONJECTURE
 -> D39 (BAR-DIMENSIONS-NEEDS R196) SUCCESS

EXPLORE D42
EXECUTED: 0.04 SECONDS 0.1 REALTIME
STATUS -> SUCCESS
EXPLOIT D42
<--- D41 (BAR-DIMENSIONS R196)
     ENTERED: NIL
     FIRED: SE
     -INVES
     STATUS -> SUCCESS
     EXPLOIT D41
     <--- D28 (IS-A-BAR-METHOD1 R196)
          ENTERED: (PUI SI SE A)
          FIRED: SE
          -INVES
          STATUS -> SUCCESS
          EXPLOIT D28
          <--- * D43 (IS-A-BAR R196)
               +INVES
               ENTERED: (SE SI PUI A)
               FIRED: SE
               -INVES
               STATUS -> SUCCESS
               EXPLOIT D43
               <--- * D45 (IS-A-HAMMER-PART-HANDLE-METHOD1 R196)
                    +INVES
                    ENTERED: (PUI SI SE A)
                    FIRED: SE
                    -INVES
                    STATUS -> SUCCESS
                    EXPLOIT D45
                    <--- * D46 (IS-A-HAMMER-PART-HANDLE R196)
                         +INVES
                         ENTERED: (SE SI PUI A)
                         FIRED: SE
                         -INVES
                         STATUS -> SUCCESS
                         EXPLOIT D46
                         <--- * D47 (HAS-A-HAMMER-PART-HANDLE DR1)
                              +INVES
```

```
                                        ENTERED: (SE SI)
                                        FIRED: SE
                                        -INVES
                                        STATUS -> SUCCESS
                                        EXPLOIT D47
                                        <--- * D49 (IS-A-HAMMER-METHOD1 OR1)
                                             +INVES
                                             ENTERED: (PUI SI SE A)
                                             FIRED: SE
                                             DIFF -> 417.51
                                             PRIORITY -> 119.75
                                        <--- * D48 (HAS-A-HAMMER-PART-HEAD-METHOD1 OR1)
                                             +INVES
                                             ENTERED: (PUI SI SE A)
                                             FIRED: SE
                                             DIFF -> 267.51
                                             PRIORITY -> 1.86E-3
                              <--- * D44 (IS-A-HAMMER-PART-HEAD-WHOLE R196)
                                   +INVES
                                   ENTERED: (PUI SI SE A)
                                   FIRED: SE
                                   -INVES
                                   STATUS -> SUCCESS
                                   EXPLOIT D44
                                   <--- * D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)
                                        +INVES
                                        ENTERED: (PUI SI SE A)
                                        FIRED: SE
                                        DIFF -> 6.92
                                        PRIORITY -> 72.15

NEW TOP PRIORITY INVESTIGATION: D49 (IS-A-HAMMER-METHOD1 OR1)
```

FIGURE 6 - 22

```
CYCLE C50
INVESTIGATING D49 (IS-A-HAMMER-METHOD1 DR1)
AT: D67 (TOUCH-METHOD1 R196 R140)


STATE OF INVESTIGATION
OF D49 (IS-A-HAMMER-METHOD1 DR1)
AT START OF CYCLE C50

-> D49 (IS-A-HAMMER-METHOD1 DR1) CONJECTURE
 -> D51 (HAS-A-HAMMER-PART-HEAD DR1) CONJECTURE
  -> D48 (HAS-A-HAMMER-PART-HEAD-METHOD1 DR1) CONJECTURE
   -> D52 (HAS-A-HAMMER-PART-HEAD-METHOD1-GT DR1) CONJECTURE
    -> D54 (HAS-A-HAMMER-PART-HEAD-METHOD1-G DR1 A1) SUCCESS
    -> D53 (HAS-A-HAMMER-PART-HEAD-METHOD1-T DR1) CONJECTURE
     -> D59 (HAS-THE-HAMMER-PART-HEAD DR1 R140) CONJECTURE
      -> D60 (HAS-THE-HAMMER-PART-HEAD-METHOD1 DR1 R140) CONJECTURE
       -> D62 (H-REL-P2 DR1 R140) CONJECTURE
        -> D63 (H-REL-P1-P2 R196 R140) CONJECTURE
         -> D64 (T-JOINED R196 R140) CONJECTURE
          -> # D66 (TOUCH R196 R140) # CONJECTURE
           -> $$ D67 (TOUCH-METHOD1 R196 R140) $$ CONJECTURE
          -> D65 (PERP R196 R140) CONJECTURE
       -> D61 (IS-A-HAMMER-PART-HEAD R140) CONJECTURE
   -> D47 (HAS-A-HAMMER-PART-HANDLE DR1) SUCCESS <- (D49)
 -> D47 (HAS-A-HAMMER-PART-HANDLE DR1) SUCCESS <- (D48)

EXPLORE D67
---> * D69 (TOUCH-METHOD1-NEEDS R196 R140)
     ENTERED: (PUI SI SE A)
     FIRED:
---> * D68 (TOUCH-METHOD1-COMPUTATION R196 R140)
     ENTERED: (PUI SI SE A)
     FIRED:
```

FIGURE 6 - 23

```
                    CYCLE C200
                    INVESTIGATING D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)
                    AT: D234 (HAS-A-HAMMER-PART-HEAD-PART-FACE-METHOD1 R196)


                    STATE OF INVESTIGATION
                    OF D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)
                    AT START OF CYCLE C200

-> D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196) CONJECTURE
 -> # D233 (HAS-A-HAMMER-PART-HEAD-PART-FACE R196) # CONJECTURE
  -> $$ D234 (HAS-A-HAMMER-PART-HEAD-PART-FACE-METHOD1 R196) $$ CONJECTURE
 -> D44 (IS-A-HAMMER-PART-HEAD-WHOLE R196) SUCCESS

EXPLORE D234
---> * D235 (HAS-THE-HAMMER-PART-HEAD-PART-FACE-METHOD1 R196 R145)
     ENTERED: (SE PUI A)
     FIRED:
```

FIGURE 6 - 24

```
CYCLE C225
INVESTIGATING D263 (IS-A-HAMMER-METHOD1 DR4)
AT: D269 (HAS-A-HAMMER-PART-HANDLE-METHOD2-TOMAS DR4)


STATE OF INVESTIGATION
OF D263 (IS-A-HAMMER-METHOD1 DR4)
AT START OF CYCLE C225

-> D263 (IS-A-HAMMER-METHOD1 DR4) CONJECTURE
 -> D264 (HAS-A-HAMMER-PART-HANDLE DR4) CONJECTURE
  -> D266 (HAS-A-HAMMER-PART-HANDLE-METHOD1 DR4) CONJECTURE
  -> D262 (HAS-A-HAMMER-PART-HANDLE-METHOD2 DR4) CONJECTURE
   -> # D270 (HAS-A-HAMMER-PART-HANDLE-METHOD2-NEEDS DR4) # SUCCESS
   -> $$ D269 (HAS-A-HAMMER-PART-HANDLE-METHOD2-TOMAS DR4) $$ CONJECTURE
   -> D268 (HAS-A-HAMMER-PART-HANDLE-METHOD2-DUM DR4) CONJECTURE
   -> D267 (HAS-A-HAMMER-PART-HANDLE-METHOD2-V DR4) CONJECTURE
   -> D261 (HAS-A-HAMMER-PART-HEAD DR4) SUCCESS <- (D263)
  -> D265 (HAS-A-HAMMER-PART-HANDLE-METHOD3 DR4) CONJECTURE
 -> D261 (HAS-A-HAMMER-PART-HEAD DR4) SUCCESS <- (D262)

EXPLORE D269
EXECUTED: 0.26 SECONDS 1.13 REALTIME
STATUS -> SUCCESS
EXPLOIT D269
<--- D262 (HAS-A-HAMMER-PART-HANDLE-METHOD2 DR4)
     ENTERED: NIL
     FIRED: SE
     DIFF -> 18.42
```

FIGURE 6 - 25

```
                    CYCLE C262
                    INVESTIGATING D263 (IS-A-HAMMER-METHOD1 DR4)
                    AT: D267 (HAS-A-HAMMER-PART-HANDLE-METHOD2-V DR4)


                    STATE OF INVESTIGATION
                    OF D263 (IS-A-HAMMER-METHOD1 DR4)
                    AT END OF CYCLE C262

-> D263 (IS-A-HAMMER-METHOD1 DR4) CONJECTURE
 -> D264 (HAS-A-HAMMER-PART-HANDLE DR4) CONJECTURE
  -> D266 (HAS-A-HAMMER-PART-HANDLE-METHOD1 DR4) CONJECTURE
  -> D262 (HAS-A-HAMMER-PART-HANDLE-METHOD2 DR4) CONJECTURE
   -> D270 (HAS-A-HAMMER-PART-HANDLE-METHOD2-NEEDS DR4) SUCCESS
   -> D269 (HAS-A-HAMMER-PART-HANDLE-METHOD2-TOMAS DR4) SUCCESS
   -> D268 (HAS-A-HAMMER-PART-HANDLE-METHOD2-DUM DR4) SUCCESS
   -> $$ D267 (HAS-A-HAMMER-PART-HANDLE-METHOD2-V DR4) $$ CONJECTURE
   -> D261 (HAS-A-HAMMER-PART-HEAD DR4) SUCCESS <- (D263)
  -> D265 (HAS-A-HAMMER-PART-HANDLE-METHOD3 DR4) CONJECTURE
 -> D261 (HAS-A-HAMMER-PART-HEAD DR4) SUCCESS <- (D262)

EXPLORE D267
EXECUTED: 0.06 SECONDS 0.23 REALTIME
STATUS -> SUCCESS
EXPLOIT D267
<--- D262 (HAS-A-HAMMER-PART-HANDLE-METHOD2 DR4)
     ENTERED: NIL
     FIRED: SE
     STATUS -> SUCCESS
     EXPLOIT D262
     <--- D264 (HAS-A-HAMMER-PART-HANDLE DR4)
          ENTERED: NIL
          FIRED: SE
          STATUS -> SUCCESS
          EXPLOIT D264
          <--- * D316 (HAS-A-HAMMER-PART-HEAD-METHOD1 DR4)
               +INVES
               ENTERED: (PUI SI SE A)
               FIRED: SE
               DIFF -> 267.51
               PRIORITY -> 1.86E-3
          <--- D263 (IS-A-HAMMER-METHOD1 DR4)
               ENTERED: NIL
               FIRED: SE
               -INVES
               STATUS -> SUCCESS
               EXPLOIT D263
               <--- * D317 (IS-A-HAMMER DR4)
                    +INVES
                    ENTERED: (SE SI PUI A)
                    FIRED: SE
                    -INVES
```

```
                STATUS -> SUCCESS
                EXPLOIT D317
                <--- * D318 (RECOGNITION DR4)
                     +INVES
                     ENTERED: (SE SI)
                     FIRED: SE
                     -INVES
                     STATUS -> SUCCESS
                     EXPLOIT D318
***** RECOGNIZED DR4 AS A HAMMER *****
```

CYCLE 16: The priority system has rated D19 to be the highest priority top level datum at this point, and we begin to pursue its investigation. D15 is already present, and established immediately below it. D20 is now initiated.

CYCLE 17: D20 is executed, D19 established, D21 suggested.

CYCLE 18: Back down again. D19 was a "serial" datum; we needed the results under D15 to compute upon in D20. D21 is a "parallel" datum. We need two properties. They can be acquired in whichever order responds best to the scene. In this case, we already have established D19. The exploration of D21 also pursues D6. This datum is already present the PK but has not yet been linked to D21; the link is now entered.

CYCLE 19: D6 is also not yet established. However, there is already a datum structure below it, some of which has been established. Recall D8 was established in cycle 15. The investigation of D6 had earlier been abandoned for more promising pursuits. Now we return to it in the context of a larger investigation. Thus when the monitor looks within the D21 investigation for the next suggestion to pursue it moves all the way down to D7. And the success of D7 is all that remained to complete the investigation.

Now we are going to skip briefly through some high spots in the course of the recognition. If you have about had enough detail and the size of cycle 33 seems burdensome, all you really need do is read the following description, glance at the figure, and observe that one exhibits a formal implementation of the other. The basic suggestion and advice operations can be formalized in exploration and exploitation, combining PK results with GK relationships to expand and massage the PK towards a recognition of the hammer.

CYCLE 33: At this point SEER has found that region R196 is bar shaped. This is sufficient to establish it crudely as a hammer handle possibility. By itself, this would not constitute a recognition of a hammer handle, but in the context of a hammer recognition supported by other features as well, it would be sufficient, and it is enough to suggest looking for the inclusive hammer. Thus we say we have a "hammer-part-handle", i.e. "hammer part: handle", rather than a "hammer handle". A bar shape also suggests the possibility that the region is a hammer head, but we require more of even a crude head recognition, "hammer-part-head", before venturing to act upon it (we will want to observe a face

also). The hammer-part-handle recognition not only suggests that a hammer is present, but provides advice on a method for finding the necessary head that will use the previously acquired handle.

The hammer recognition now becomes top priority and SEER pursues it. We already have the hammer-part-handle; we look for a hammer-part-head; the method advised is employed.

CYCLE 50: A region, R140, has been generated as a hammer-part-head candidate. There are several properties to be established. SEER chooses to test the relationship of R196 to R140 before making the considerable effort to establish that R140 is a hammer-part-head.

The relationship in fact fails to hold. Other candidates are tried but all fail. We do not have strong enough evidence to warrant a really extensive effort to salvage the conjecture.

CYCLE 200: SEER now goes back to pursue the suggestion that R196 is an hammer-part-head. We need to find a striking face for the head (hammer-part-head-part-face). Region R145 is suggested.

The hammer-part-head conjecture eventually succeeds, and in turn suggests a hammer and methods for finding the handle.

CYCLE 225: SEER is seeking a hammer-part-handle for DR4. We are employing here a method for finding regions developed by Tomas Lozano-Perez. His method looks for a region with a specified surface profile. The profile, the starting point, and a starting direction must be provided. This constitutes an excellent example of SEER's ability as a system to integrate results in individual areas. The hammer-part-head recognition permits precisely the "advice" that Lozano's program needs. SEER invokes the Lozano method ("method2") and transmits the required specifications. This interaction at processing time and extensibility at programming time is SEER's forte.

The method does find a region meeting the specified conditions. Some further testing is then done to see if it suffices as a hammer-part-handle.

Cycle 262: The Tomas method has won and the hammer recognition has been completed.

We may also get a broader view by looking at various kinds of "paths" through pieces of the finished PK structure. Figure 6 - 26 illustrates the order in which datums were established. (The number 280 after datum D317 indicates that this was the 280th datum initiated. C262 is the cycle during which it was established. D263, is-a-hammer-method1 DR4, was the datum being investigated when D317 was established. D267 was the datum explored to start off cycle C262.) Figure 6 - 27

FIGURE 6 - 26

```
                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D317 (IS-A-HAMMER DR4)
                    TO DEPTH 2.
                    SHOWING THE ESTABLISHMENT PATH

  -> D317 (IS-A-HAMMER DR4)
     280. C262 (D263 (IS-A-HAMMER-METHOD1 DR4)) (D267 (HAS-A-HAMMER-PART-HANDLE-
METHOD2-V DR4))
    -> D263 (IS-A-HAMMER-METHOD1 DR4)
       279. C262 (D263 (IS-A-HAMMER-METHOD1 DR4)) (D267 (HAS-A-HAMMER-PART-HANDLE-
METHOD2-V DR4))
     -> D264 (HAS-A-HAMMER-PART-HANDLE DR4)
        278. C262 (D263 (IS-A-HAMMER-METHOD1 DR4)) (D267 (HAS-A-HAMMER-PART-
HANDLE-METHOD2-V DR4))
     -> D261 (HAS-A-HAMMER-PART-HEAD DR4)
        230. C220 (D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)) (D250 (COVER-END-
COMPUTATION R196 R145))
```

FIGURE 6 - 27

```
PK STATE
AT END OF CYCLE C262
BELOW D317 (IS-A-HAMMER DR4)
TO DEPTH 2.
SHOWING THE INITIATION PATH

  -> D317 (IS-A-HAMMER DR4)
     317. C262 (D263 (IS-A-HAMMER-METHOD1 DR4)) (D267 (HAS-A-HAMMER-PART-HANDLE-
METHOD2-V DR4))
    -> D263 (IS-A-HAMMER-METHOD1 DR4)
       263. C220 (D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)) (D250 (COVER-END-
COMPUTATION R196 R145))
     -> D264 (HAS-A-HAMMER-PART-HANDLE DR4)
        264. C221 (D263 (IS-A-HAMMER-METHOD1 DR4)) (D263 (IS-A-HAMMER-METHOD1
DR4))
     -> D261 (HAS-A-HAMMER-PART-HEAD DR4)
        261. C220 (D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196)) (D250 (COVER-END-
COMPUTATION R196 R145))
```

indicates the order in which datums were initiated (created). Figure 6 - 28 reflects the exploration sequence. (NIL indicates that the node was never explored; it may nevertheless have been established through exploitiation of nodes below.) Figure 6 - 29 reflects the exploration sequence (actuation path) to a greater depth, printing only the ordinal sequence numbers.

Figure 6 - 30 is a partial recognition path for the sledge hammer scene discussed first above, indicating the order in which the results were established; figure 6 - 31 is an abbreviated exploration path for more of the PK. Comparing them with the corresponding figures for the scene just discussed we see that the handle of the sledge was recognized first, and processing then proceeded in the opposite direction to the processing of the ball peen hammer scene.

FIGURE 6 - 28

```
PK STATE
AT END OF CYCLE C262
BELOW D317 (IS-A-HAMMER DR4)
TO DEPTH 2.
SHOWING THE ACTUATION PATH

  -> D317 (IS-A-HAMMER DR4)
     NIL
   -> D263 (IS-A-HAMMER-METHOD1 DR4)
      219. C221 (D263 (IS-A-HAMMER-METHOD1 DR4)) (D263 (IS-A-HAMMER-METHOD1 DR4))
    -> D264 (HAS-A-HAMMER-PART-HANDLE DR4)
       220. C222 (D263 (IS-A-HAMMER-METHOD1 DR4)) (D264 (HAS-A-HAMMER-PART-HANDLE
DR4))
    -> D261 (HAS-A-HAMMER-PART-HEAD DR4)
       NIL
```

FIGURE 6 - 29

```
                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D317 (IS-A-HAMMER DR4)
                    TO DEPTH 7.
                    SHOWING THE ACTUATION PATH

-> D317 (IS-A-HAMMER DR4) NIL
 -> D263 (IS-A-HAMMER-METHOD1 DR4) 219.
  -> D264 (HAS-A-HAMMER-PART-HANDLE DR4) 220.
   -> D266 (HAS-A-HAMMER-PART-HANDLE-METHOD1 DR4) NIL
   -> D262 (HAS-A-HAMMER-PART-HANDLE-METHOD2 DR4) 221.
    -> D270 (HAS-A-HAMMER-PART-HANDLE-METHOD2-NEEDS DR4) 222.
     -> D31 (L-AXIS R196) NIL
      -> D29 (L-AXIS-METHOD1 R196) 22.
       -> D30 (L-AXIS-METHOD1-COMPUTATION R196) 23.
       -> D26 (L-AXIS-METHOD1-NEEDS R196) 21.
     -> D38 (WIDTH R196) NIL
      -> D36 (WIDTH-METHOD1 R196) 26.
       -> D37 (WIDTH-METHOD1-COMPUTATION R196) 27.
       -> D35 (W-AXIS R196) NIL
    -> D269 (HAS-A-HAMMER-PART-HANDLE-METHOD2-TOMAS DR4) 223.
    -> D268 (HAS-A-HAMMER-PART-HANDLE-METHOD2-DUM DR4) 224.
     -> D271 (BAR-DIMENSIONS NR1) 225.
      -> D273 (BAR-DIMENSIONS-NEEDS NR1) 226.
       -> D275 (LENGTH NR1) 227.
       -> D274 (WIDTH NR1) 249.
      -> # D272 (BAR-DIMENSIONS-COMPUTATION NR1) # 259.
    -> $$ D267 (HAS-A-HAMMER-PART-HANDLE-METHOD2-V DR4) $$ 260.
    -> D261 (HAS-A-HAMMER-PART-HEAD DR4) NIL
     -> D260 (IS-A-HAMMER-PART-HEAD R196) NIL
      -> D50 (IS-A-HAMMER-PART-HEAD-METHOD1 R196) 196.
       -> D233 (HAS-A-HAMMER-PART-HEAD-PART-FACE R196) 197.
       -> D44 (IS-A-HAMMER-PART-HEAD-WHOLE R196) NIL
  -> D265 (HAS-A-HAMMER-PART-HANDLE-METHOD3 DR4) NIL
 -> D261 (HAS-A-HAMMER-PART-HEAD DR4) NIL
```

FIGURE 6 - 30

```
PK STATE
AT START OF CYCLE C233
BELOW D272 (IS-A-HAMMER DR1)
TO DEPTH 2.
SHOWING THE ESTABLISHMENT PATH

  -> D272 (IS-A-HAMMER DR1)
     246. C233 (D50 (IS-A-HAMMER-METHOD1 DR1)) (D255 (COVER-END-COMPUTATION R272
R246))
    -> D50 (IS-A-HAMMER-METHOD1 DR1)
       245. C233 (D50 (IS-A-HAMMER-METHOD1 DR1)) (D255 (COVER-END-COMPUTATION R272
R246))
     -> D52 (HAS-A-HAMMER-PART-HEAD DR1)
        244. C233 (D50 (IS-A-HAMMER-METHOD1 DR1)) (D255 (COVER-END-COMPUTATION
R272 R246))
     -> D48 (HAS-A-HAMMER-PART-HANDLE DR1)
        43. C33 (D46 (IS-A-HAMMER-PART-HANDLE-METHOD1 R278)) (D43 (BAR-DIMENSIONS-
COMPUTATION R278))
```

FIGURE 6 - 31

```
                    PK STATE
                    AT START OF CYCLE C233
                    BELOW D272 (IS-A-HAMMER DR1)
                    TO DEPTH 7.
                    SHOWING THE ACTUATION PATH

-> D272 (IS-A-HAMMER DR1) NIL
 -> D50 (IS-A-HAMMER-METHOD1 DR1) 32.
  -> D52 (HAS-A-HAMMER-PART-HEAD DR1) 33.
   -> D49 (HAS-A-HAMMER-PART-HEAD-METHOD1 DR1) 34.
    -> D53 (HAS-A-HAMMER-PART-HEAD-METHOD1-GT DR1) 93.
     -> D115 (HAS-A-HAMMER-PART-HEAD-METHOD1-G DR1 A2) 94.
      -> D117 (HAS-A-HAMMER-PART-HEAD-METHOD1-GN DR1 A2) 95.
       -> D119 (HAS-A-HAMMER-PART-HEAD-METHOD1-GNN DR1 A2) 96.
       -> D118 (HAS-A-HAMMER-PART-HEAD-METHOD1-GNC DR1 A2) 97.
      -> D116 (HAS-A-HAMMER-PART-HEAD-METHOD1-GC DR1 A2) 98.
     -> D114 (HAS-A-HAMMER-PART-HEAD-METHOD1-T DR1) 99.
      -> D120 (HAS-THE-HAMMER-PART-HEAD DR1 R272) 100.
       -> D121 (HAS-THE-HAMMER-PART-HEAD-METHOD1 DR1 R272) 101.
     -> D55 (HAS-A-HAMMER-PART-HEAD-METHOD1-G DR1 A1) 36.
      -> D57 (HAS-A-HAMMER-PART-HEAD-METHOD1-GN DR1 A1) 37.
       -> D59 (HAS-A-HAMMER-PART-HEAD-METHOD1-GNN DR1 A1) 38.
       -> D58 (HAS-A-HAMMER-PART-HEAD-METHOD1-GNC DR1 A1) 39.
      -> D56 (HAS-A-HAMMER-PART-HEAD-METHOD1-GC DR1 A1) 40.
     -> D54 (HAS-A-HAMMER-PART-HEAD-METHOD1-T DR1) 41.
      -> D60 (HAS-THE-HAMMER-PART-HEAD DR1 R264) 42.
       -> D61 (HAS-THE-HAMMER-PART-HEAD-METHOD1 DR1 R264) 43.
    -> D48 (HAS-A-HAMMER-PART-HANDLE DR1) NIL
     -> D47 (IS-A-HAMMER-PART-HANDLE R278) NIL
      -> D46 (IS-A-HAMMER-PART-HANDLE-METHOD1 R278) NIL
       -> D44 (IS-A-BAR R278) NIL
 -> D48 (HAS-A-HAMMER-PART-HANDLE DR1) NIL
```

7

ACTIVE KNOWLEDGE CONTROL

THE BASIC QUESTIONS THE CONTROL STRUCTURE MUST DEAL WITH ARE: WHAT, HOW, AND WHEN

What do we do, how and when do we do it? The answers need to be effective and efficient. The first issue is whether SEER can perform its recognition tasks at all. Proficiency needs to be accompanied by reasonable efficiency to be practical.

WHAT: SEER generates tasks based on the given scene. "What do we do" breaks down into two basic questions: "what do we look at" and "what do we look for". Where do we look? What regions do we investigate? Which properties, objects and relationships do we attempt to determine? In SEER, we distinguish initial generation of regions and tasks, that "prime the pump", from subsequent suggestions that are developed by active knowledge.

HOW: Once a task is generated, how is it investigated? What methods are applied? Which alternatives are pursued? How much "proof" is required?

WHEN: In what order are tasks performed? Which suggestions are in fact carried out? What do we do next? What is most promising? When do we continue on the current investigation? For how long? When do we leave to try something else? When and how do we return? Within an investigation, how do we order alternative methods or requirements? When do we move from one alternative to another? There will be many places at which to begin an investigation, and different paths along which to proceed. We want to find the "winning line" to the correct identification. We want proceed in the most efficient order, avoid getting stuck in a losing line of inquiry, doing unnecessary tasks. We want to establish or refute hypotheses quickly.

Clearly the distinction between what, how and when is not absolute. Advice on how to accomplish a task affects the ordering of alternative methods. Advice itself is explicitly viewed by SEER as suggestions about what to do. Basically, I have been presenting general problems in process control. These problems have a particular character and emphasis in vision research, of course. The "generation" problem, what to look at, is particularly acute in vision. There is such a mass of parallel

sensory input. There is a great deal of redundant and irrelevant information. What stands out (where to start), what approach to use, how much investigation is required, varies from scene to scene for the same object or property. Advice on how to proceed and flexibility in the order of processing is vital, and must, in particular, be motivated by the specific scene.

CONTROL IS THE CENTRAL CONCERN OF THE SEER SYSTEM.

Chapter two described our interactive knowledge structures: levels and areas of knowledge communicate. Chapter three described an interactive control structure: investigations and processes communicate. Interaction in the knowledge structures is reflected in control structure interaction. Active knowledge, as explained before, is the cooperation of the GK and PK knowledge structures to guide the control structure. PK intermediate results consult GK relationships to effect processing operations.

The knowledge structure was designed to embody ideas on how having x can help get y during processing. The knowledge structure consists of means of acquiring descriptions, performing recognitions. The GK was analyzed for potentially useful interactions that can help guide processing. Control structure mechanisms employ these interactions to effect active knowledge. The GK holds the potential. PK results trigger it. This chapter will discuss further the types of direction AK can provide and the specific mechanisms that SEER employs to effect AK. Our basic aim is to analyze, formalize and institutionalize, as specific mechanisms, in a system, vague ideas about interaction and heterarchy. We work down to pinpoint "natural" processes like "suggestion" and "advice", analyze and implement their components.

SEER'S BASIC ADVANTAGE IS ITS ABILITY TO RESPOND FLEXIBLY AND APPROPRIATELY TO THE SCENE

There are an indefinite variety of PK contexts possible for different scenes and processing states. The input "problem", the real visual world, can not be well constrained. It is an impossible programming task to specifically account for and organize all possible optimal processing sequences in response to all scenes and all sequences of results of operating on those scenes. One cannot write a "precompiled" monitor: do a, if you find b and have c and d; do e, if you find f and have c and not d;

We emphasize here the role of conjecture and success, particularly partial success, in generating suggestions. Failure, of course, can also be "exploited", in particular to initiate suggestions.

The generation of what to do has two aspects: what to look for and what to look at. These correspond to the predicate and the arguments in a suggestion. The initial region pass provides us with some regions of interest. The seed suggestions supply some predicates to apply to these regions. Subsequent suggestions are provided by AK.

INDEPENDENT VERSUS INFLUENCED PROCESSING

Suggestions may be characterized as "independent" or "influenced". The seed suggestions are made independently of the scene. Later suggestions are pursued or proposed by other datums, reflecting previous processing influence. The investigation of the suggestion may also be characterized as independent or influenced, depending on whether how and when it is developed is affected by previous results. The issue arises: what is the best relative balance of independent vs influenced activity?

This is not quite a "pass oriented" vs "heterarchical" argument. The independent activity need not be confined to coherent "pass". This argument over passes does arise in SEER with reference to the initial region acquisition pass. The independent activity need not all be performed initially before influenced processing begins. We may return to independent activity or at least independently initiated suggestions. What are the issues?

On the one hand, we would like to confine the independent processing to as little as possible, and get it out of the way early on. We want to move quickly on to influenced activity, to take advantage in SEER of active knowledge processing. Our basic bias is that "influence" is beneficial. On the other hand, we clearly need some independent processing to get started. We can not begin by asking if there is a hammer at input grid point (369,440). Moreover, other needs for independent processes may be argued.

There is some danger of "bootstrapping" ourselves up the garden path. What we expect to see should influence visual processing; however, we do not want to just "see what we want to see". We do not want to ignore contradictions. Independent processing may provide needed checks on our

do d, if ... . Long "cond" clauses, will at best reflect local, ad hoc programming decisions.

SEER puts together an appropriate program or programming sequence from the GK. The suggestion based control structure, priority system and monitor all contribute. The cycle structure permits global response to the PK state, while individual chunks of knowledge still enjoy local control of exploration and exploitation. The alternative control flows possible permit a flexible response to the variety of reality.

WHAT DO WE DO? -- SUGGESTIONS PROVIDE CANDIDATES

Suggestions have a specific definition in SEER. The suggestion process is implemented first by initiation of datums. "Resuggestion" of already present conjectures is reflected in priority values and est count. Broader senses of suggestion, covering actions without initiation, are encompassed by the term "exploitation". Suggestions, pursued or proposed, have a uniform role in the suggestion pool. Suggestions are not commands. SEER chooses to execute or ignore them, as appropriate. Suggestions are based on relationships between visual phenomena.

The key motivation is that behind proposals. Heuristically, we feel that when we learn a fact it may be useful for something:

It may help compute something else.
It may be part of the "definition" of a larger concept: a piece, a property.
It may hold relationships to other scene elements.
It may supply a context which makes other results more or less likely.

There are so many phenomena we could investigate. It makes sense to suggest or encourage those we already know something about, where a result has already provided partial success or other support. SEER formalizes this concept. First a relevance analysis forms phenomena into components and provides establishment and priority links between them. Then initiation links are formed based on the establishment and priority relationships. Links are "fired" to exploit results.

Pursued phenomena are also handled as suggestions. This has many advantages. Basically it permits a uniform "little man" and "suggestion pool" organization. Rather than pursue an investigation to its conclusion immediately, each stage of the investigation, as it is pursued downward, must compete in the suggestion pool. Initiation links used in pursuing suggestions are also based on the establishment and priority structures.

We emphasize here the role of conjecture and success, particularly partial success, in generating suggestions. Failure, of course, can also be "exploited", in particular to initiate suggestions.

The generation of what to do has two aspects: what to look for and what to look at. These correspond to the predicate and the arguments in a suggestion. The initial region pass provides us with some regions of interest. The seed suggestions supply some predicates to apply to these regions. Subsequent suggestions are provided by AK.

INDEPENDENT VERSUS INFLUENCED PROCESSING

Suggestions may be characterized as "independent" or "influenced". The seed suggestions are made independently of the scene. Later suggestions are pursued or proposed by other datums, reflecting previous processing influence. The investigation of the suggestion may also be characterized as independent or influenced, depending on whether how and when it is developed is affected by previous results. The issue arises: what is the best relative balance of independent vs influenced activity?

This is not quite a "pass oriented" vs "heterarchical" argument. The independent activity need not be confined to coherent "pass". This argument over passes does arise in SEER with reference to the initial region acquisition pass. The independent activity need not all be performed initially before influenced processing begins. We may return to independent activity or at least independently initiated suggestions. What are the issues?

On the one hand, we would like to confine the independent processing to as little as possible, and get it out of the way early on. We want to move quickly on to influenced activity, to take advantage in SEER of active knowledge processing. Our basic bias is that "influence" is beneficial. On the other hand, we clearly need some independent processing to get started. We can not begin by asking if there is a hammer at input grid point (369,440). Moreover, other needs for independent processes may be argued.

There is some danger of "bootstrapping" ourselves up the garden path. What we expect to see should influence visual processing; however, we do not want to just "see what we want to see". We do not want to ignore contradictions. Independent processing may provide needed checks on our

progress. If we have found ourselves led off into a wrong track, without advice on recovery, it is nice to have some place to begin afresh.

Or when we do succeed in an investigation, there may still be unaccounted for portions of the scene which remain "in the dark", and where we need to begin again "priming the suggestion pump". We do not want to do all these independent computations in an initial pass, many may prove unnecessary. However, they should be there to return to when we have worked our way "up", successfully or otherwise, as far as we can in one area of the scene. (Of course, results in one area provide general context, occlusion, etc. that influences further portions of the scene.) Ideally the independent seed suggestions should be "complete", in the sense that if we eventually execute them all we will have triggered all possible analyses that SEER can make of the scene. We may want to execute additional independent processes to begin several influenced investigations running in parallel.

The force of these arguments and the amount of independent processing they imply depends somewhat on one's confidence in the success of influenced processing and the processing flow of one's system in operation. There are two basic arguments. Independent options should be available when needed; hardly arguable, the question is how much will they be needed. Independent processing should play a larger role to assist or check upon "influenced" processing.

As usual the extremes are questionable. The strictly pass oriented approach that computes all properties independent of intermediate results or higher level advice fails. A system like Wizard [Winston 1972b] which does almost no independent processing is not very flexible.

SEER has the initial independent region acquisition pass and the "seed suggestions". These latter are the primitive uncertain possibilities, thus fairly "complete", eventually leading up to all uncertain phenomena. The regions are not intended to be complete. It is meaningless to think of all possible regions, and we do not wish to choose the right ones this early.

Accounting for all regions of the scene, if it is required, could provide some check on hasty identifications of scene portions. Verification of conjectures should help guard against hasty hypotheses, and alternatives remain available. Seed suggestions are returned to as appropriate: when called for from above, or when nothing "better" (higher priority) is in the suggestion pool.

do d, if ... . Long "cond" clauses, will at best reflect local, ad hoc programming decisions.

SEER puts together an appropriate program or programming sequence from the GK. The suggestion based control structure, priority system and monitor all contribute. The cycle structure permits global response to the PK state, while individual chunks of knowledge still enjoy local control of exploration and exploitation. The alternative control flows possible permit a flexible response to the variety of reality.

WHAT DO WE DO? -- SUGGESTIONS PROVIDE CANDIDATES

Suggestions have a specific definition in SEER. The suggestion process is implemented first by initiation of datums. "Resuggestion" of already present conjectures is reflected in priority values and est count. Broader senses of suggestion, covering actions without initiation, are encompassed by the term "exploitation". Suggestions, pursued or proposed, have a uniform role in the suggestion pool. Suggestions are not commands. SEER chooses to execute or ignore them, as appropriate. Suggestions are based on relationships between visual phenomena.

The key motivation is that behind proposals. Heuristically, we feel that when we learn a fact it may be useful for something:

It may help compute something else.
It may be part of the "definition" of a larger concept: a piece, a property.
It may hold relationships to other scene elements.
It may supply a context which makes other results more or less likely.

There are so many phenomena we could investigate. It makes sense to suggest or encourage those we already know something about, where a result has already provided partial success or other support. SEER formalizes this concept. First a relevance analysis forms phenomena into components and provides establishment and priority links between them. Then initiation links are formed based on the establishment and priority relationships. Links are "fired" to exploit results.

Pursued phenomena are also handled as suggestions. This has many advantages. Basically it permits a uniform "little man" and "suggestion pool" organization. Rather than pursue an investigation to its conclusion immediately, each stage of the investigation, as it is pursued downward, must compete in the suggestion pool. Initiation links used in pursuing suggestions are also based on the establishment and priority structures.

THERE ARE MANY WAYS IN WHICH THE MANNER IN WHICH A PHENOMENON IS ESTABLISHED MAY DIFFER

We may generate arguments in different ways. In particular there are alternative sources of regions as candidates for hypothesized objects. There are different means of establishing identifications or features. In particular, how hard do we look, how easily should we be convinced, how much "proof" do we need? In the context of a hammer recognition, for example, a crude handle recognition suffices. A hammer handle by itself requires subtle shape distinctions for recognition.

GUIDANCE ON HOW TO PROCEED MAY BE BASED ON RESULTS OR ON HYPOTHESES

The PK normally represents a dynamic context, the scene being processed. (It could contain specific context statements, like "the lighting is from the right", but does not at present.) A global context is provided by the established results. A local context is provided by hypotheses, when working "down in context" to verify conjectures. A hammer handle result can be exploited upwards to advise the location of the head. Exploring downward in the context of a hammer handle conjecture, we do not need to run a general purpose boundary shape finder, but can look for a bar shape. A bar shape conjecture can look specifically for long straight sides parallel to the axis.

IT IS EASIER TO SEE SOMETHING IF YOU KNOW WHAT YOU ARE LOOKING FOR

In particular, after proposing a conjecture, we can use verification methods to establish it. It may be easier to ask "is it x?", than "what is it?" For example, "is it straight?" as opposed to "what shape is it?" (Of course, we cannot simply ask "is it x1, is it x2,...?" Thus we have appropriate suggestion mechanisms to tell us what to look for, and the priority system to pick out the most promising possible perceptions.)

Special techniques are available within hypothesized contexts, or given previous results. Consider the boundary finding problem, for example. To answer the question--what is the boundary?--we have to apply general methods for corner finding segmentation, chain encoding or whatever. It may be difficult to judge which features are significant, and which are noise or secondary effects.

However, if we ask--does it have two long parallel straight sides like a "bar" or "cylinder"?--

there are specialized verification techniques we can apply, and previous results may give us some idea of significant dimensions. Consider the outer boundary of a cylinder. (An early version of SEER used cylindrical dowels as a domain, in preparation for hammer handles.) Our "straw man" passive type of processing would act on this boundary in the following manner.

First the feature points of the scene would be found and organized into lines. Then the boundary line would be segmented, into corners, straight lines, "uniformly" curved segments perhaps, or some more complex description of curved boundaries. Then possible predicates would be tried on the pieces to determine possible relations. The two straight lines would be found to be parallel and of the same length, the two curved segments of the same shape and size and oppositely directed. Finally the built up description of the cylinder boundary might be matched against models of object boundaries. All these operations, from line finding on up, are difficult to execute in a general context. We may have serious failures along the line.

In SEER it is likely that when we go looking for a cylinder boundary, we are pursuing the conjecture that some region is a cylinder. We also have, or will look for, symmetry in the region. The two symmetric axes constitute the length and width axes of the region. If the region has a cylinder boundary, it will have a straight side parallel to the length axis, and just a little shorter, running through the endpoint of the width axis.

With this position pointed out, we can proceed to "verify" the line in several ways. We can employ a standard line verification or tracking technique. In this case we can in fact use a trick involving the boundary of the region. A general method might search for local maxima and minima of boundary points about the center of gravity, with the attendant problems of a local process. Instead we find all boundary points close to the hypothesized line, and use this set to find the endpoints and verify the existence of the line. (This trick is supported by another property of a cylinder-like region, convexity. We may already have established convexity, and it may have "advised" this trick, or we can pursue convexity now.)

Now that we have one side, symmetry gives us the other. Symmetry tells us the sides have equal length; and the existence of two symmetric axes implies that the sides are parallel. We already know that they are parallel to the major symmetric axis; that is how we found them. So the relations

are taken care of. The endpoints of the two lines specify the remaining sides. We can quickly verify that one of these is a simple "single" curve, e.g. it has no concavities. The presence, shape and relative orientation of the other end is implied by the symmetry again.

THE MODULAR, INCREMENTAL ASPECTS OF SEER ARE PARTICULARLY WELL SUITED TO HANDLING MULTIPLE CASES

We should avoid the "one, two, infinity fallacy" that may lead us to give up too soon when a few alternatives prove insufficient. A reasonable finite number of methods may still suffice. Under intelligent direction such specialized techniques can succeed where they may be impossible or innefficient to apply a priori. Employing specialized methods is not merely a question of efficiency. Generalized methods or dragnet syntactic searches may simply not work. A general method tends to either miss subtleties or get confused by them, e.g. find too little or too much. Such is the problem with line finding methods. This dichotomy may make a general solution impossible. It makes sense that the proper choice from a set of special methods will often be more effective and efficient than a general solution. The problem has been to provide effective, efficient means of integrating such collections in a uniform, motivated system (as opposed to an ad hoc collection of "hacks") and applying them to specific cases.

SEER IMPLEMENTS ADVICE AS SUGGESTED METHODS

A common metaphor for advice involves "messages". Unfortunately, this approach can lend itself to handwaving. When you come right down to it, advice on how to do something suggests a method for doing it. SEER makes these methods explicit. They provide modular alternatives. They can be handled with the same machinery that deals with other suggestions. Thus when we find a hammer handle we propose a method for finding the head (figure 7 - 1).

A specialized method in SEER can be advised by the appropriate context of previous results or seek such results. The priority system can choose among methods in a uniform manner. Methods can use previous results to assist computation or justify less stringent requirements. Methods can be developed to deal with specific problems and added as required.

FIGURE 7 - 1

```
                    CYCLE C33 <Excerpts>
                    INVESTIGATING D41 (BAR-DIMENSIONS R196)
                    AT: D42 (BAR-DIMENSIONS-C R196)

EXPLORE D42
EXECUTED: 0.04 SECONDS 0.13 REALTIME
STATUS -> S
EXPLOIT D42
                .
                .
                .
                     <--- * D46 (IS-A-HP1 R196)
                          +INVES
                          ENTERED: (SE SI PUI A)
                          FIRED: SE
                          -INVES
     <Handle established:>
                          STATUS -> S
                          EXPLOIT D46
                          <--- * D47 (HAS-A-HP1 DR1)
                               +INVES
                               ENTERED: (SE SI)
                               FIRED: SE
                               -INVES
                               STATUS -> S
                               EXPLOIT D47
                               <--- * D49 (IS-A-H-M1 DR1)
                                    +INVES
                                    ENTERED: (PUI SI SE A)
                                    FIRED: SE
                                    DIFF -> 417.51
                                    PRIORITY -> 119.75
     <Head method proposed:>
                               <--- * D48 (HAS-A-HP2-M1 DR1)
                                    +INVES
                                    ENTERED: (PUI SI SE A)
                                    FIRED: SE
                                    DIFF -> 267.51
                                    PRIORITY -> 1.86E-3
                .
                .
                .
```

These methods provide alternatives. Some alternatives may be easier than others. They may work in different circumstances. There may be versions of differing precision, requiring various levels of verification. The appropriate method may depend on the use to which it is put, what other results are available or being sought to go with it. A crude method may be sufficient to initiate a further investigation or verify a final step. A hammer handle alone requires a subtle shape recognition; in the context of a hammer, a crude bar shape is a sufficient identification. Pseudo objects allow crude observations to make suggestions and serve as recognitions within an appropriate context. On the other hand if necessary a pseudo object could propose a more positive independent recognition requiring finer observations, e.g. "hp1" could propose "hammer handle". The methods chosen may vary with the entry point and direction of processing flow within an investigation. The GK may have alternatives available for identifying a handle alone, to initiate an investigation of a hammer, to verify the part given a hammer hypothesis, to suggest an occluded handle, etc. (These methods can share subpieces, of course.) There may be conditions to meet before a method is tried. Methods may incorporate other PK knowledge. While alternatives are held in the "suggestion pool", relevant information can build up. The method may be advised again, its priority may increase (it can even be established). The more we know, the more we can help guide alternatives or suggest superior ones.

BY FILTERING UP AND SEEKING THE WINNING LINE SEER FOCUSES ON SUCCESS AND AVOIDS UNNECESSARY OBSERVATIONS

SEER only seeks the minimum needed to verify results in the context of hypotheses and previous results. This allows simpler methods and fewer requirements. Briefly SEER will only see what it needs to see in a scene. Suppose, for example, we are trying to recognize the side of a cylinder, having already identified a region, R, as having the overall properties of a cylinder "as a whole".

Now the side of a cylinder has a number of necessary properties. If we were simply given some region and asked to decide if it was a cylinder-side, "in vacuo", these properties would all have to be checked out. However, some of these properties are implied by the properties of R that we have found already. For example, in finding that the boundary of R is cylinderlike, we have already established relevant information about three sides of the cylinder-side boundary.

This knowledge may not be in easily "shared" form. Data base "assertions" that involve R, as a cylinder, may not simply "match" needed facts about a cylinder-side. Some "higher level" sharing is required. We could have a conjecture access a method capable of "deducing" an assertion from such related facts, when a simple data base match failed. In SEER the presence of the established facts about R advise (suggest) a method for establishing a cylinder-side for R. This method assumes what is already available, and merely seeks to verify a few more details.

We already have a "profile" of R, segmenting it into two (three-dimensionally) straight surfaces lengthwise. We take additional profiles on either side. Some may be affected by noise, or fall along a line where intensities on side and face are equal. However, say two out of three, should coincide, to enable us to verify a distinct side of uniform length and curvature.

Suggestion and advice, specialized methods, in context processing, verification, crude definitions, flexible techniques, all attempt the "easy way out". We can "look for a bar shape" rather than "find the shape", we can use crude pseudo objects. We can move quickly to a winning hypothesis in simple situations. I believe people operate similarly. If a scene is familiar or easy to parse, many properties and details will not be perceived.

SOURCES OF ADVICE

Sources of advice are based on the relationships that motivate suggestions generally. Again we implement the basic heuristic aim: "have x, help..." The PK context as it develops interacts with GK knowledge of how items relate and can help each other. If we have some establishing features of a phenomena, it may advise which other features are needed or a method to establish them. For example, if we know that an object is symmetric, we only need to verify "half" its boundary shape. In figure 7 - 2 we see a method for determining if two regions touch, which allows a perpendicularity context to advise a simple method of computation.

If we have some parts of an object, this can advise how much we need to verify for the remainder. If we have a conjectured relationship and one member of it, this can help find the other. Special attention is given to using relationships to help direct attention and locate regions. The most obvious example is the help that the head region can give in restricting a basic search for the face;

FIGURE 7 - 2

```
                    CYCLE C51
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D69 (TOUCH-M1-N R196 R140)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C51

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> D53 (HAS-A-HP2-M1-T DR1) C
     -> D59 (HAS-THE-HP2 DR1 R140) C
      -> D60 (HAS-THE-HP2-M1 DR1 R140) C
       -> D62 (H-REL-P2 DR1 R140) C
        -> D63 (H-REL-P1-P2 R196 R140) C
         -> D64 (T-JOINED R196 R140) C
          -> D66 (TOUCH R196 R140) C
           -> # D67 (TOUCH-M1 R196 R140) # C
            -> $$ D69 (TOUCH-M1-N R196 R140) $$ C
            -> D68 (TOUCH-M1-C R196 R140) C
          -> D65 (PERP R196 R140) C
       -> D61 (IS-A-HP2 R140) C
  -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D69
---> D65 (PERP R196 R140)
     ENTERED: (PUI SI SE A)
     FIRED:
```

we are not testing regions all over the picture. If we are seeking to establish a hammer head by finding the striking face of the head, the location and dimensions of the region sought are constrained. We might also use the features of the object we seek to help generate a candidate region. E.g. we have a region finding method that specializes in looking for flat regions (like hammer head faces). It is easier to find a significant region when you can use requirements to guide the generation process: Lozano's methods make use of profile "templates", in a sense. The larger context, e.g. lighting, location, may modify further investigation.

SEER DECIDES WHICH SUGGESTIONS TO EXPLORE AND IN WHAT ORDER

SEER "code" does not specify: "do this, then this..." We provide a network of possible programs in the GK. The PK puts together the "program" for a scene from this raw material. Even when SEER chooses "what" to do, these are not immediate orders, but "suggestions" for what might be done, when appropriate. The "suggestion pool" mechanisms permit us to hold many "competing" suggestions, waiting while results accumulate that reinforce or discourage them. Promising conjectures will "rise to the top" of the pool. SEER is "success driven". We will discuss the priority system and the monitor that try to find the "winning line" in the PK network. First we look at the processing sequences that they produce.

FLEXIBLE PROCESSING SEQUENCES RESPOND TO THE VARIETY OF REALITY

As Simon has observed, simple internal structures, when environment driven, can produce complex seeming behavior [Simon 1969]. The idea is to be flexible enough to deal with obscurity or take advantage of clarity in a given scene. To this end, the GK provides many potential recognition routes whose instantiation and execution is dependent on active knowledge interaction.

INVESTIGATIONS ARE "MULTIENTRY" AND "MULTIDIRECTIONAL"

SEER can enter an investigation from "above" conjecturing the existence of the whole, or from "below" discovering a piece or property and proposing the next step up. And, of course, we can enter at an intermediate level. SEER can then proceed in the investigation through various choices of nodes,

in various sequences; various initiation, exploration and establishment paths result. Figure 7 - 3 presents establishment paths for two hammer recognitions, on two different scenes. In the first the head was found first, in the second, the handle. Figures 7 - 4 and 7 - 5 compare actuation paths. We established the head conjecture after entry from below in one case, above in another.The computations appropriate to different paths differ. Recognition of a hammer handle "from below", by itself, will require precise shape determination. In the context of a complete hammer, any bar or cylinder shaped recognition will serve to verify the handle.

THE ASPECTS OF A SCENE WHICH "STAND OUT" OR ARE EASY TO RECOGNIZE CAN BE IDENTIFIED FIRST AND HELP TO MAKE FURTHER WORK EASIER OR UNNECESSARY

Shirai [Shirai 1972] showed how useful is the simple principle of doing the easy things first, and letting them help in the harder investigations. SEER allows more general use of the principle, and in a less restricted domain. A scene in which a head blends into the shadowy background and the handle is shiny, need not be treated in the same uniform way as one in which the wooden handle blends with the workbench, while the dark head stands out. SEER exploits the likelihood that in any "distortion" there will be some aspect that remains straightforward, that stands out easily. If one feature is obscure another will be clear, one part distorted, another obvious, one object disguised, a suggestive context present. SEER is prepared to work from whatever entry point is convenient, toward a complete recognition, using partial results to facilitate or obviate more difficult tasks. Even where a feature is not simple and straightforward by itself, a specialist may make it so. We conjecture that a reasonable number of distortion specialists, together with a manageable set of simple "canonical" appearances, will enable us to get started on a wide variety of instantiations of a feature or object. SEER can then bootstrap its way along using previous results to assist further efforts.

ACTIVE KNOWLEDGE DIRECTS PROCESSING AMONG ALTERNATIVE PATHS WITHIN AN INVESTIGATION AND AMONG ALTERNATIVE INVESTIGATIONS

It coroutines alternatives, if you like. SEER can start pursuing one processing branch, stop, pursue some alternative method or investigation, return. The current state of any branch is available

FIGURE 7 - 3

```
                    HEAD FOUND FIRST

                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D317 (IS-A-H DR4)
                    TO DEPTH 2.
                    SHOWING THE E PATH

-> D317 (IS-A-H DR4)
   280. C262 (D263 (IS-A-H-M1 DR4)) (D267 (HAS-A-HP1-M2-V DR4))
 -> D263 (IS-A-H-M1 DR4)
    279. C262 (D263 (IS-A-H-M1 DR4)) (D267 (HAS-A-HP1-M2-V DR4))
  -> D264 (HAS-A-HP1 DR4)
     278. C262 (D263 (IS-A-H-M1 DR4)) (D267 (HAS-A-HP1-M2-V DR4))
  -> D261 (HAS-A-HP2 DR4)
     230. C220 (D50 (IS-A-HP2-M1 R196)) (D250 (COVER-END-C R196 R145))

                    HANDLE FOUND FIRST

                    PK STATE
                    AT START OF CYCLE C233
                    BELOW D272 (IS-A-H DR1)
                    TO DEPTH 2.
                    SHOWING THE E PATH

-> D272 (IS-A-H DR1)
   246. C233 (D50 (IS-A-H-M1 DR1)) (D255 (COVER-END-C R272 R246))
 -> D50 (IS-A-H-M1 DR1)
    245. C233 (D50 (IS-A-H-M1 DR1)) (D255 (COVER-END-C R272 R246))
  -> D52 (HAS-A-HP2 DR1)
     244. C233 (D50 (IS-A-H-M1 DR1)) (D255 (COVER-END-C R272 R246))
  -> D48 (HAS-A-HP1 DR1)
     43. C33 (D46 (IS-A-HP1-M1 R278)) (D43 (BAR-DIMENSIONS-C R278))
```

FIGURE 7 - 4

```
                    HEAD FOUND FIRST

                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D317 (IS-A-H DR4)
                    TO DEPTH 7.
                    SHOWING THE A PATH

-> D317 (IS-A-H DR4) NIL
 -> D263 (IS-A-H-M1 DR4) 219.
  -> D264 (HAS-A-HP1 DR4) 220.
   -> D266 (HAS-A-HP1-M1 DR4) NIL
   -> D262 (HAS-A-HP1-M2 DR4) 221.
    -> D270 (HAS-A-HP1-M2-N DR4) 222.
     -> D31 (L-AXIS R196) NIL
      -> D29 (L-AXIS-M1 R196) 22.
       -> D30 (L-AXIS-M1-C R196) 23.
       -> D26 (L-AXIS-M1-N R196) 21.
     -> D38 (WIDTH R196) NIL
      -> D36 (WIDTH-M1 R196) 26.
       -> D37 (WIDTH-M1-C R196) 27.
       -> D35 (W-AXIS R196) NIL
    -> D269 (HAS-A-HP1-M2-TOMAS DR4) 223.
    -> D268 (HAS-A-HP1-M2-DUM DR4) 224.
     -> D271 (BAR-DIMENSIONS NR1) 225.
      -> D273 (BAR-DIMENSIONS-N NR1) 226.
       -> D275 (LENGTH NR1) 227.
       -> D274 (WIDTH NR1) 249.
      -> # D272 (BAR-DIMENSIONS-C NR1) # 259.
   -> $$ D267 (HAS-A-HP1-M2-V DR4) $$ 260.
   -> D261 (HAS-A-HP2 DR4) NIL
    -> D260 (IS-A-HP2 R196) NIL
     -> D50 (IS-A-HP2-M1 R196) 196.
      -> D233 (HAS-A-HP2P1 R196) 197.
      -> D44 (IS-A-HP2-W R196) NIL
  -> D265 (HAS-A-HP1-M3 DR4) NIL
 -> D261 (HAS-A-HP2 DR4) NIL
```

FIGURE 7 - 5

HANDLE FOUND FIRST

PK STATE
AT START OF CYCLE C233
BELOW D272 (IS-A-H DR1)
TO DEPTH 7.
SHOWING THE A PATH

```
-> D272 (IS-A-H DR1) NIL
 -> D50 (IS-A-H-M1 DR1) 32.
  -> D52 (HAS-A-HP2 DR1) 33.
   -> D49 (HAS-A-HP2-M1 DR1) 34.
    -> D53 (HAS-A-HP2-M1-GT DR1) 93.
     -> D115 (HAS-A-HP2-M1-G DR1 A2) 94.
      -> D117 (HAS-A-HP2-M1-GN DR1 A2) 95.
       -> D119 (HAS-A-HP2-M1-GNN DR1 A2) 96.
       -> D118 (HAS-A-HP2-M1-GNC DR1 A2) 97.
      -> D116 (HAS-A-HP2-M1-GC DR1 A2) 98.
     -> D114 (HAS-A-HP2-M1-T DR1) 99.
      -> D120 (HAS-THE-HP2 DR1 R272) 100.
       -> D121 (HAS-THE-HP2-M1 DR1 R272) 101.
     -> D55 (HAS-A-HP2-M1-G DR1 A1) 36.
      -> D57 (HAS-A-HP2-M1-GN DR1 A1) 37.
       -> D59 (HAS-A-HP2-M1-GNN DR1 A1) 38.
       -> D58 (HAS-A-HP2-M1-GNC DR1 A1) 39.
      -> D56 (HAS-A-HP2-M1-GC DR1 A1) 40.
     -> D54 (HAS-A-HP2-M1-T DR1) 41.
      -> D60 (HAS-THE-HP2 DR1 R264) 42.
       -> D61 (HAS-THE-HP2-M1 DR1 R264) 43.
    -> D48 (HAS-A-HP1 DR1) NIL
     -> D47 (IS-A-HP1 R278) NIL
      -> D46 (IS-A-HP1-M1 R278) NIL
       -> D44 (IS-A-BAR R278) NIL
  -> D48 (HAS-A-HP1 DR1) NIL
```

in the PK for reference (sharing results) or return. The investigation state snapshot in figure 7 - 6 provides an illustration. The irregular shape of the investigation, the several "indentations" in the figure, reflect the fact that several branches of the investigation have been partially pursued at this point. The "#" previously explored datum and the "$" currently explored datum, indicate that the investigation has shifted from one branch to another in the current cycle.

Decisions on which of several OR methods or AND properties to pursue respond to the dynamic PK state; the monitor continually reassesses the situation. E.g. if the easy method for establishing A fails, another exploration may seem more promising than the remaining methods, at least for a while; or a positive result can improve the outlook in several areas and reorder our interest.

Response to the scene can be viewed in terms of a single investigation or over the whole of the PK, among investigations. Within an investigation, SEER helps choose the most effective, efficient processing sequence by determining in what order to explore properties, what methods to apply, and thus how much information is needed, where to look, etc. Among investigations, SEER focuses attention on the most interesting and promising questions and areas of the scene.

PROCESSING IS "NON PAROCHIAL", OR "HOMOGENEOUS"

We do not get stuck in a "black box". Results established while working on one processing branch can affect other branches. They may even cause a shift of control to another branch. Information flow is not strictly on a "who called?" basis. The paths in figures 7 - 7 , 7 - 8 and 7 - 9 illustrate this by showing which datum was being investigated when the various datums were initiated, explored, and established.

Partial successes, even a losing cause, can remain in the data base for reference, and may suggest or encourage other possibilities. A region which is bar shaped but fails to have handle dimensions may suggest an occluded handle. SEER depends on interaction to make its "when" decisions. We know that levels and areas of knowledge interact. "Interactive processing" also involves investigations interacting dynamically through homogeneous processing. These interactions occur over time, through accumulating exploitation and exploration.

FIGURE 7 - 6

STATE OF INVESTIGATION
OF D66 (IS-A-H-M1 DR1)
AT START OF CYCLE C129

```
-> D66 (IS-A-H-M1 DR1) C
 -> D67 (HAS-A-HP2 DR1) C
  -> D65 (HAS-A-HP2-M1 DR1) C
   -> D68 (HAS-A-HP2-M1-GT DR1) C
    -> D84 (HAS-A-HP2-M1-G DR1 A2) S
    -> D83 (HAS-A-HP2-M1-T DR1) C
     -> D89 (HAS-THE-HP2 DR1 R272) C
      -> D90 (HAS-THE-HP2-M1 DR1 R272) C
       -> D92 (H-REL-P2 DR1 R272) C
        -> D115 (H-REL-P1-P2 R278 R272) C
         -> D116 (T-JOINED R278 R272) C
          -> D118 (TOUCH R278 R272) C
           -> D119 (TOUCH-M1 R278 R272) C
            -> D121 (TOUCH-M1-N R278 R272) C
             -> D117 (PERP R278 R272) C <- (D116)
              -> D123 (PERP-N R278 R272) C
               -> D124 (L-AXIS R272) C <- (D156 D149)
                -> D125 (L-AXIS-M1 R272) C
                 -> D127 (L-AXIS-M1-N R272) C
                  -> D128 (HAS-BAR-SIDES R272) C <- (D156 D144)
                   -> D129 (HAS-BAR-SIDES-M1 R272) C
                    -> D131 (HAS-BAR-SIDES-M1-N R272) S
                    -> $$ D130 (HAS-BAR-SIDES-M1-C R272) $$ C
                  -> D112 (O-BDRY R272) S <- (D139 D131 D105)
                 -> D126 (L-AXIS-M1-C R272) C
               -> D45 (L-AXIS R278) S <- (D88 D74 D41 D46)
              -> D122 (PERP-C R278 R272) C
            -> D120 (TOUCH-M1-C R278 R272) C
           -> D117 (PERP R278 R272) C <- (D121)
       -> D91 (IS-A-HP2 R272) C
        -> D93 (IS-A-HP2-M1 R272) C
         -> D95 (IS-A-HP2-W R272) C
          -> D143 (IS-A-BAR R272) C
           -> D144 (IS-A-BAR-M1 R272) C
            -> D128 (HAS-BAR-SIDES R272) C <- (D156 D127)
            -> D145 (BAR-DIMENSIONS R272) C
             -> D147 (BAR-DIMENSIONS-N R272) C
              -> D149 (LENGTH R272) C
               -> D124 (L-AXIS R272) C <- (D156 D123)
               -> D150 (LENGTH-C R272) C
              -> D148 (WIDTH R272) C
               -> D151 (WIDTH-M1 R272) C
                -> D153 (W-AXIS R272) C
                 -> D154 (W-AXIS-M1 R272) C
                  -> # D156 (W-AXIS-M1-N R272) # C
                   -> D124 (L-AXIS R272) C <- (D149 D123)
                   -> D128 (HAS-BAR-SIDES R272) C <- (D144 D127)
                  -> D155 (W-AXIS-M1-C R272) C
```

```
                    -> D152 (WIDTH-M1-C R272) C
                 -> D146 (BAR-DIMENSIONS-C R272) C
              -> D94 (HAS-A-HP2P1 R272) C
               -> D96 (HAS-A-HP2P1-M1 R272) C
                -> D97 (HAS-THE-HP2P1-M1 R272 R269) C
                 -> D99 (HAS-THE-HP2P1-M1-CE R272 R269) F
                 -> D98 (HAS-THE-HP2P1-M1-CF R272 R269) C
    -> D70 (HAS-A-HP2-M1-G DR1 A1) S
    -> D69 (HAS-A-HP2-M1-T DR1) F
  -> D64 (HAS-A-HP1 DR1) S <- (D66)
-> D64 (HAS-A-HP1 DR1) S <- (D65)
```

FIGURE 7 - 7

```
PK STATE
AT END OF CYCLE C262
BELOW D6 (CONVEX R196)
TO DEPTH 5.
SHOWING THE I PATH

-> D6 (CONVEX R196)
   6. C3 (D2 (SEED R196)) (D2 (SEED R196))
 -> D8 (CONVEX-N R196)
    8. C4 (D2 (SEED R196)) (D6 (CONVEX R196))
  -> D10 (AREA R196)
     10. C5 (D2 (SEED R196)) (D8 (CONVEX-N R196))
   -> D12 (O-BDRY R196)
      12. C6 (D2 (SEED R196)) (D10 (AREA R196))
    -> D14 (BDRY R196)
       14. C7 (D2 (SEED R196)) (D12 (O-BDRY R196))
    -> D13 (O-BDRY-C R196)
       13. C7 (D2 (SEED R196)) (D12 (O-BDRY R196))
   -> D11 (AREA-C R196)
      11. C6 (D2 (SEED R196)) (D10 (AREA R196))
  -> D9 (CONVEX-HULL R196)
     9. C5 (D2 (SEED R196)) (D8 (CONVEX-N R196))
   -> D17 (VEX-STR-ANA R196)
      17. C12 (D15 (SCH-N R196)) (D9 (CONVEX-HULL R196))
    -> D12 (O-BDRY R196)
       12. C6 (D2 (SEED R196)) (D10 (AREA R196))
    -> D18 (VEX-STR-ANA-C R196)
       18. C13 (D15 (SCH-N R196)) (D17 (VEX-STR-ANA R196))
   -> D16 (CONVEX-HULL-C R196)
      16. C12 (D15 (SCH-N R196)) (D9 (CONVEX-HULL R196))
 -> D7 (CONVEX-C R196)
    7. C4 (D2 (SEED R196)) (D6 (CONVEX R196))
```

FIGURE 7 - 8

```
                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D6 (CONVEX R196)
                    TO DEPTH 5.
                    SHOWING THE A PATH

-> D6 (CONVEX R196)
   2. C4 (D2 (SEED R196)) (D6 (CONVEX R196))
 -> D8 (CONVEX-N R196)
    3. C5 (D2 (SEED R196)) (D8 (CONVEX-N R196))
  -> D10 (AREA R196)
     4. C6 (D2 (SEED R196)) (D10 (AREA R196))
   -> D12 (O-BDRY R196)
      5. C7 (D2 (SEED R196)) (D12 (O-BDRY R196))
    -> D14 (BDRY R196)
       6. C8 (D2 (SEED R196)) (D14 (BDRY R196))
    -> D13 (O-BDRY-C R196)
       7. C9 (D2 (SEED R196)) (D13 (O-BDRY-C R196))
   -> D11 (AREA-C R196)
      8. C10 (D2 (SEED R196)) (D11 (AREA-C R196))
  -> D9 (CONVEX-HULL R196)
     10. C12 (D15 (SCH-N R196)) (D9 (CONVEX-HULL R196))
   -> D17 (VEX-STR-ANA R196)
      11. C13 (D15 (SCH-N R196)) (D17 (VEX-STR-ANA R196))
    -> D12 (O-BDRY R196)
       5. C7 (D2 (SEED R196)) (D12 (O-BDRY R196))
    -> D18 (VEX-STR-ANA-C R196)
       12. C14 (D15 (SCH-N R196)) (D18 (VEX-STR-ANA-C R196))
   -> D16 (CONVEX-HULL-C R196)
      13. C15 (D15 (SCH-N R196)) (D16 (CONVEX-HULL-C R196))
 -> D7 (CONVEX-C R196)
    17. C19 (D21 (SCH-AS-DESC R196)) (D7 (CONVEX-C R196))
```

FIGURE 7 - 9

```
PK STATE
AT END OF CYCLE C262
BELOW D6 (CONVEX R196)
TO DEPTH 5.
SHOWING THE E PATH

-> D6 (CONVEX R196)
   16. C19 (D21 (SCH-AS-DESC R196)) (D7 (CONVEX-C R196))
 -> D8 (CONVEX-N R196)
    12. C15 (D19 (SCH R196)) (D16 (CONVEX-HULL-C R196))
  -> D10 (AREA R196)
     6. C10 (D2 (SEED R196)) (D11 (AREA-C R196))
   -> D12 (O-BDRY R196)
      4. C9 (D2 (SEED R196)) (D13 (O-BDRY-C R196))
    -> D14 (BDRY R196)
       2. C8 (D2 (SEED R196)) (D14 (BDRY R196))
    -> D13 (O-BDRY-C R196)
       3. C9 (D2 (SEED R196)) (D13 (O-BDRY-C R196))
   -> D11 (AREA-C R196)
      5. C10 (D2 (SEED R196)) (D11 (AREA-C R196))
  -> D9 (CONVEX-HULL R196)
     10. C15 (D15 (SCH-N R196)) (D16 (CONVEX-HULL-C R196))
   -> D17 (VEX-STR-ANA R196)
      8. C14 (D15 (SCH-N R196)) (D18 (VEX-STR-ANA-C R196))
    -> D12 (O-BDRY R196)
       4. C9 (D2 (SEED R196)) (D13 (O-BDRY-C R196))
    -> D18 (VEX-STR-ANA-C R196)
       7. C14 (D15 (SCH-N R196)) (D18 (VEX-STR-ANA-C R196))
   -> D16 (CONVEX-HULL-C R196)
      9. C15 (D15 (SCH-N R196)) (D16 (CONVEX-HULL-C R196))
 -> D7 (CONVEX-C R196)
    15. C19 (D21 (SCH-AS-DESC R196)) (D7 (CONVEX-C R196))
```

SEER'S APPROACH IS BASICALLY PARALLEL

Alternatives are laid out, pursued or proposed, together, and parallel options remain available to the monitor. This parallelism is a side effect, to an degree, of an organization designed originally to facilitate the suggestion and advice interaction of active knowledge. Admittedly, this may not be your usage of the term "parallel", and of course a serial processing order is chosen; we only employ a single processor.

It is not unreasonable to experiment with parallel structures in vision. There are motivations for expecting that the visual domain is receptive to parallel approaches. We start, of course, with the unorganized, non linear nature of the input. At the "lowest" levels, there are, of course, neurophysiological motivations for studying parallel structures in vision. B.K.P. Horn's recent results in perceiving "lightness", suggest a parallel implementation [Horn 1974]. He worked with David Marr, who has neurophysiological theories on lightness with some parallel features [Marr 1974]. My work in region finding, and other work before me, has a parallel flavor.

On an even higher level, we have Dave Waltz's efforts [Waltz 1972]. I feel that his program can be viewed as carrying forward all possible conjectures, about his class of scene predicates, in parallel. Most notably, this did not result in an exponential explosion. Rather as more of the scene was viewed, and more types of scene predicates, more classes of knowledge were understood and pursued, the process converged to the correct analysis.

SEER AMELIORATES BUT DOES NOT SOLVE THE PROBLEM OF COMBINATORIAL EXPLOSION

Many important choices are to be made within even a single investigation. SEER is I hope clearly of use in this regard. Among investigations, one does reach a point where the bogeyman of combinatorial explosion raises its head. This may not occur as quickly as one might think. Exploitation only occurs one level up at a time. Results combine to cut down some possibilities and clearly enhance others as they accumulate. There will hopefully be a "Waltz Effect" with larger amounts of interacting knowledge and results. (Notice how SEER allows investigations to develop and interact in "parallel", analogously to the Waltz algorithm.) Hardware implementations of linking mechanisms could stave off the "population bomb" a bit longer. At worst, we are still better off than the build a description and

match with models passive approach, described in chapter one, which demands a complete scene graph.

While it may seem very wasteful to make a lot of suggestions in parallel, this passive approach would really in some sense have to compute everything. All possible properties and relationships have to be computed since we do not know which might be needed to match some model. At least in SEER, we have some hope of not computing everything. Consider a worst case simple model of the system in operation.

A region R has property p1. This proposes that the region satisfies predicates in the set S. So we now have to pursue all the properties P of all the members of S. But these are not necessarily all the properties that could be tried for R; and would be tried in our straw man passive system. They at least have some likelihood of success. There may be properties Q that nowhere coexist with property p1; why check for them?

Further, remember these properties P are not really pursued all at once in parallel. They are pursued according to a priority and monitor system. And our hierarchical structure is organized so that we proceed upward in appropriate small chunks, and discriminate quickly among large sets of possibilities.

As we determine more properties of R, more suggestions are made; but we really cut down our possibilites, in a Waltz-like way, as we learn more and proceed upward. We proceed upward in stages: p1 and p2 establish q1; q1 proposes r1, which pursues q2; q1 and q2 establish r1;... A jagged procession, two steps forward, one step back.

But in many cases we intersect, contradict and halt. "Round" suggests a lot of things, "red" suggests a lot of things, but only items in their intersection, e.g. apples not fire-engines, remain. Fruit bowls, not fires, will be proposed next. Furthermore the conjecture that a round, red region is an apple will have a higher priority than a conjecture that it is a fire engine, simply because we have two pieces of evidence for it.

Eventually, however, we will have to employ larger "chunking" of knowledge. As Marvin Minsky has cogently pointed out, we will require "frames" [Minsky 1974]. Frames [Minsky 1974] provide a means of organizing "context quanta". Such contexts might be implemented by employing SEER mechanisms at higher levels. The exploitation of a PK result, for example, could activate a

workbench "frame", by altering the GK in "immediate memory" or changing a priori likelihoods.

We have the beginnings of a frame system already, in some sense. We work now with low level elements: bar, hammer, convex. We could move up: tools, workbench, house. SEER is more "homogeneous" than some frame theories. Its egocentric elements refuse to confine themselves to one "frame". Their non parochial activity helps direct the monitor among frames and suggest new ones. At a higher level we may need to accumulate outside references before hauling in new frames.

In SEER the GK reflects an initial static context. The priority factors in particular embody the a priori context. However, the structure and contents of the GK as a whole could be modified initially or dynamically to reflect circumscribed contexts: expectations or interests. A different GK would cause different PK initiation and linking. The PK in turn could alter the GK dynamically, pulling sections (frames?) out of "deeper" memory, when called for.

In summary, SEER is clearly of assistance in handling multiple posssibilities up to a point. These include a variety of visual phenomena, methods and processing sequences in response to varied scenes and PK states. Beyond this, we need large knowledge chunking mechanisms. SEER may be useful at this level; however, this is not a focus of the current research.

SPECIFICALLY SERIAL MECHANISMS ARE OVERLAID ON THE PARALLEL BASE

We can initiate, pursue or propose, alternatives serially, or require that a set of alternatives below a node be established in order. It is instructive to consider when and why serial mechanisms are required. For example, when the cost of actually generating a candidate is large, we may want to dispose of one before generating the next. When a computation acts upon certain facts, these facts must be ascertained before the computation can be performed.

Often, however, what appears to be a need for serial requirements is really a desire to constrain the system to employ the most efficient or effective processing order. The issue in other words is specifying "strategy". Among the possible paths, the parallel alternatives, that we have alluded to above, which should SEER choose? We may be given a phenomena with four independent features, for example, and feel compelled to require that these be determined in some specific order. When we examine this compulsion we might find that feature B is very hard to determine. We feel

unwilling to waste effort on it when a very easy check for feature A exists that could give a negative result and terminate the investigation.

ACTIVE KNOWLEDGE ADAPTS A PK PROCESSING SEQUENCE TO THE SCENE FROM PARALLEL GK POSSIBILITIES

In other words strategy is a priority issue. In convential black box programming we tend to specify processing sequences on an ad hoc, local basis. We may attempt to provide alternatives, based on conditional checks. However, we eventually will be unable to account for the variety of global contexts (i.e. PK states) that may exist to influence our sequencing choices. At any rate the results will not be very modular. We will simply be unable eventually to program or understand the system adequately on such an individually coded basis. SEER has a global priority system. The issues that affect sequencing decisions can be studied explicitly and generally in this system. Global influences are accounted for.

In conventional "one dimensional" programming, the linear order of our coding statements makes us think naturally in a serial fashion. If four features must be found, or four alternatives are available, we have to ask for them in a specific order. We may devote some effort to determining what is the best order. We may even provide some alternatives.

In SEER, we "program" in the GK where the four features or options are organized "two dimensionally" below the node requiring them. If one truly requires the results of another, a serial order can be imposed. Otherwise the organization is naturally "parallel". SEER can choose the most promising to explore first, based on knowledge available about the given scene at the time of choice. Furthermore, we do not have to completely "compute" one choice before going on to the next. We move from one to another, as results dictate.

Figure 7 - 10 compares sequencing decisions in standard programming with those in SEER. Basically SEER can adapt to the scene more flexibly than we could program by hand, given the indefinite variety of possible scenes and PK states at decision time and the large body of GK options. When we add the problem of adding new options and expanding the domain incrementally, SEER's advantages mount. Winograd described similar problems in large language systems [Winograd 1971].

FIGURE 7 - 10

SEQUENCING DECISIONS

NAIVE:
- AD HOC
- LOCAL
- INDIVIDUAL
- FIXED
- FINAL
- LIMITED
- INACCESSIBLE

SEER:
- RESPOND TO PK STATE AT DECISION TIME AND TO GENERAL KNOWLEDGE
- SYSTEM MODULE
- GLOBAL
- FLEXIBLE
- PARALLEL
- EXTENSIBLE
- OBSERVABLE

We see this adaptation at work at a variety of levels: Small subtle sharing of results and ordering of investigations can occur at the low level. Special methods are chosen in the context of results or conjectures, e.g. bar shape computatation using symmetry or looking for specific bar features. Investigations are chosen and pursued based on a global evaluation of current results and alternatives. Broad recognition strategies depend on what stands out in the scene.

Even when one result will not necessarily help another, there may be a preferred order. We found that relationships between a handle and a proposed head failed before entering the more involved process of determining if there was in fact a head, with a striking face, etc. The a priori obvious order is not always the best, however. Results may partially establish one option or easy methods may fail for another. Some of these results can occur before the initial decision is made on which branch to pursue, others can occur as the investigation proceeds. Others may occur while off working on another investigation. At the beginning of each cycle, the monitor can take these results into account.

WHAT DOES THE RESULTING PROCESSING LOOK LIKE?

The sequence of successive PK's gives us one picture. The various "paths" give us others. Within an investigation SEER tends to proceed upward in a "zigzag" fashion: establish A, move up to propose B, move down to pursue B, establish B, move up to propose C. However, we also can skip between branches of an investigation, e.g. alternative methods. And we can jump between investigations. We may obtain results relevant to one investigation while pursuing another. Thus we can have several entry points and have started several paths before the investigation is finally resolved. You may want to review the cycles shown in the examples chapter at this point. Recall, for example, how we saw SEER make a "false start", considering the possibility that a region was a hammer handle and looking for a head. It then did an about face and succeeded with the opposite flow: the region was a head and helped to find the handle.

The specific processing sequence will depend on the GK organization, the PK results, and the priority and monitor systems. These sequences will have certain broad structural features. For example, SEER could spend a lot of time exploring low level phenomena, or it could jump quickly to

higher level hypotheses (figure 7 - 11). Issues like these bear on the general model of visual processing that SEER embodies. By experimenting with the elements of the system, and specific scenes, we can distinguish and evaluate alternatives.

One feature of the processing flow that we hope SEER will evidence is "convergence". We hope that the use of additional knowledge to guide the processing will act in a filtering, contraining fashion to direct us to the winning lines in the PK. Note that the order of processing can affect what is seen as well as when we see it. In an ambiguous scene with alternate interpretations we may just see the first one. The amount of detail seen depends on the path to the recognition, also the amount of "irrelevant" material noticed before finding the winning line.

THE PRIORITY SYSTEM PROVIDES THE DATA WITH WHICH THE MONITOR CAN DECIDE WHAT TO DO NEXT

The idea is to maximize effectiveness and efficiency of operation. SEER's priority system is crude, but the important point is that there is a slot in the SEER system for a priority module. A uniform, global priority system has advantages over ad hoc local decisions, as we have discussed. It also permits us to experiment with alternatives. It provides a natural opportunity for learning (though admittedly there are dangers; one can expect too much of oversimplified learning mechanisms). We use it to help develop models of visual processing.

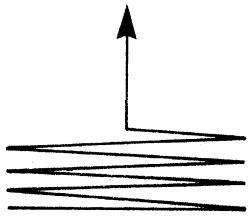
The priority factors correspond to factors in our model. "What we expect to see" is reflected by the "likelihood" factor: a priori in the GK, and conditional, changing with the PK context. Effort is measured by the difficulty factor. Interest factors reflect goals and approaches (e.g. work on high level hypotheses early or wait until lower level results accumulate).

Some factors are intrinsic to the nodes and constant. Others are dependent dynamically on the PK. The factors are combined by the monitor to determine processing sequence. There are, of course, many alternative factors and priority analyses possible; SEER has experimented with some.

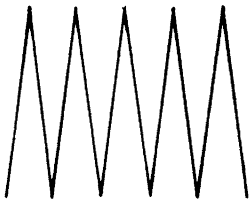
There are two ways to view use of priority. We can try to choose and use factors to reflect a general model of how visual processing is motivated. We can try to set factors "objectively", e.g. likelihood, through statistical learning. Then we can test if our priority formulas work well, produce efficient, effective results. We can experiment. On the other hand, we can try manipulating the

FIGURE 7-11

FLOW SAMPLES



Grub around a lot
before moving up.



Jump quickly to
high level hypotheses.

priority system to make processing proceed in individual cases as we would like. The former approach is preferable. It is more likely to lead to a clean general "theory". Note that a good theory will likely still admit of some "floundering".

The priority factors can be set by learning. Difficulty and likelihood in particular can be learned in obvious statistical fashion. Such learning has been discredited in AI due to misuse, but there should be places for it. The learning SEER does now is admittedly very simpleminded. These factors can be also set in ranges corresponding to heuristic judgments. The interest factor, in any case, has to take up the slack.

Updating dynamic factors is handled by active knowledge through the priority links. Updating is elementary at the moment. SEER could try to update based on partial results, or push updating through nodes not yet present. These possibilities compound the difficulties of acting on priority links. SEER did this to a degree for the difficulty factor at one time. For likelihood the problems of modelling such partial results without a formal Bayesian framework are particulary difficult. I decided to start with a simple first order system and evaluate its performance first.

A Bayesian model would be very helpful for maintaining full fledged conditional probabilities. However, the requirements of a mutually exclusive exhaustive set of phenomena are not present in our system. We have dealt with combined, cumulative effects in a heuristic ad hoc fashion.

THE MONITOR USES A TWO STAGE PROCESS TO CHOOSE THE DATUM TO EXECUTE AT EACH PROCESSING CYCLE

First the most "promising" investigation is chosen. This is basically a cost/benefit analysis. Next SEER seeks the most effective, efficient means of continuing the investigation. It prefers an unlikely, easy to compute feature that will tell it quickly if it is not on the winning line.

There are of course many other possible strategies. SEER could attempt to execute datums that promise to distinguish best among the investigations (e.g. decision theory techniques). It could favor lower nodes that most increased the likelihood of the subject datums of investigations. Other priority factors and other combinations can be employed.

Some procedures may be difficult to implement in SEER, since SEER does not enter a complete

"decision tree" into the PK at once. (Updating factors based on partial results, for example, can have distorted effects, unable to act on datums not present.) In SEER's problem domain, generating the possibilities, e.g. generating region candidates, is a task in itself. In any case, SEER chooses to motivate possibilities before cluttering its considerations with them. SEER lets the scene generate hypotheses dynamically.

Keep in mind that if the priority and monitor process becomes too involved it defeats its purpose. There is no sense in taking more effort to determine when to do things than we save by doing them in a good order. The proper balance point in this trade off must be found.

The monitor, like the priority system is just a simple first approximation. My intention was not to devote too much attention to any part of the system, but rather to emphasize interaction, and build an original system that can be bootstrapped. The important thing again is to have a place for a global monitor. It can be experimented with and evaluated; of course, we have already gone through some alternatives.

The monitor is the top level link in our chain of mechanisms for global control and non parochial processing. On the other hand, the monitor does not impose a monolithic a priori strategy. Temporary control is granted to individual knowledge elements. Active knowledge provides the dynamic basis for monitor decisions.

THE USE OF NUMBERS IN THE PRIORITY SYSTEM RAISES SOME QUESTIONS

Can the numbers capture adequate heuristic content, or should we use more sophisticated "reasons"? Are the numerical distinctions "significant"? Is this a chaotic neural net which these numbers can only pretend to understand?

Let me begin by placing in perspective my concerns in this work. Most significant is the ability of SEER to make suggestions and give advice. It is important that it can do so in a manner which does not demand immediate action, but permits these suggestions to wait, to be reinforced, and to be explored as appropriate for the scene. Next it is significant that there are global system modules that permit these suggestions to be compared. The system modules help implement a theory of visual processing in a manner subject to experimentation and review.

Now we come to the specific monitor/priority elements in use at the moment. Recall that I have avoided going too deeply into any one aspect of the system. These elements have tried to drag me into the mire of specialization as strongly as any. A number of alternatives have been experimented with. As I said, one of the features of the system is that I can run the system with one monitor, see the kinds of decisions being made in the flow of processing, learn why changes are necessary, and implement them globally. However, I am not overly pleased with the present systems, and am quite prepared to substitute others. If some of these rely less on numbers, fine. The rest of the system can support that easily.

The point is that at present the numbers only serve to summarize formalized heuristic interactions. Initial GK organization reflects these. Suggestion, advice, suggestion from several sources, acting on the relevance links, these have all contributed to the state of the PK. The numbers provide a convenient comparison of some of these results, but the interactions are not lost. The nodes and links are still available for more sophisticated analysis if desired, or further information can be transmitted when links are acted upon. Individual monitors could overlay specialized decision making on the basic system. However, we should be a little careful before we add more sophistication. To begin with we should be sure it is really required. It makes no sense for the decision making to cost more than the savings it produces. Simple systems should be understood first.

In any case, I suspect that some simple uniform system will remain useful as a basis at least, to preserve the virtures of flexibility, extensibility and modularity cited earlier. There are after all reasons why the real number system is so nice to work with. As the unique complete ordered field it has some uniquely nice properties. Whatever sophisticated heuristics we add, at some point a decision must be made: A is better than B. If we add a new element to our knowledge base, and at some point of processing it is instantiated with priority 86, this can be easily compared to the other priorities present with the standard ordering on the number system: "better" is "greater". If, however, this new element is to be supported by 86 reasons, we must define how these 86 reasons compare to the 86 reasons for every element already in our knowledge base. We must define "better" anew. The programming task and probably the processing calculation is much less straightforward. In fact, the effort is likely to degenerate into handwaving at this point, without any formal implementation. In

summary, there should be room for sophisticated comparisons, but a uniform system that easily assimilates new interactions is our first concern.

People are also concerned that insignificant numerical distinctions will be claimed to support significant decisions. Am I claiming that a priority of 86.002 is any reason for preferring an investigation over one with priority 86.001? Of course not. There may be times when such numbers turn up; they simply mean that it does not matter much which one is chosen at that time.

However, there are significant distinctions of the sort normally made in an ad hoc local fashion by programmers, that are also embodied ultimately in these numbers. We need to show A, B and C. If we have A already, B is easiest to get next, otherwise C is best to start with. SEER can be attuned to more subtle distinctions than a program might ordinarily bother with: smooth convex hull requires convex and convex hull, if we have some partial results for convex, is convex hull better to start with or ... . This is an advantage up to a point. When it passes that point and the distinctions become insignificant, there is no need to claim otherwise. Most importantly there are straightforward mechanisms for handling important decisions. Should we attempt to establish the relationship of head to handle now or determine first if it has the overall shape of a head?

Finally, we might ask whether SEER is some perceptron like chaos that the numbers are expected magically to manipulate? No. There is a strong balance here between the local interactions and the continuing global monitor review at the start of every cycle. The interactions are not random; they represent knowledge based understanding of relevance relationships. They formalize heuristic notions of suggestion and advice. Uniformity, automation and extensibility are not bought at the cost of semantic content. Special purpose and context dependent methods are encouraged. SEER does provide a uniform structure for integrating these to avoid the chaos produced by unstructured collections of "hacks", trying to achieve "distributed knowledge" within a coherent system. SEER tries to assume some of the better aspects of both the "special purpose knowledge, common sense reasoning" and the "formal system grinding" schools.

IN SUMMARY, SUGGESTIONS TELL US WHAT TO EXPECT, ADVICE HOW TO INVESTIGATE IT, AND THE PRIORITY SYSTEM WHEN TO CONSIDER IT

The easy tasks for the given scene can be done first, results can accumulate in the suggestion pool until harder tasks become promising, unnecessary computation avoided, and specialized methods handle individual cases.

8

A SYSTEMATIC APPROACH

SEER functions as a programming system, specifically as a vision system, as opposed to an ad hoc collection of programs. As a system, we want SEER to be "realizable": It should be "programmable"; some large systems become dense and unwieldy. It should be "efficient". More fundamental is the issue of "effectiveness", finding the right answer at all. However, efficiency is required for practical development and application. The system should be "institutionalized": useful operations should be formalized for the user. Finally we want an "experimental" system, that can serve as a framework for further research.

SEER FACILITATES THE TRANSFORMATION OF HEURISTIC INSIGHTS INTO FORMAL MECHANISMS

In dealing with individual recognition problems SEER encourages us to employ active knowledge analysis, using suggestion and advice. I have also found that when new system mechanisms are required they are easy to develop within the SEER context.

The system must help with several tasks. To begin with we must prepare basic GK data. Later we will want to add to it easily. The recognitions must be debugged and run. Finally we will want to experiment, use the system as a laboratory, and ultimately as a model for visual processing. SEER has several properties that should help. It is modular, explicit, and incremental, while retaining an overall framework and interactive formalism.

SEER IS EXPLICIT

SEER makes it clear what and where the knowledge is. In GK and PK the pieces are accessible, the interactions are explicit. In older heterarchical systems the higher level assumptions were often largely implicit. The system might make hypotheses based on parallelipiped assumptions but how could we add "wedge" hypotheses? In SEER the egocentric elements are modular and each knows its place in the whole scheme. There is the larger scheme to fit into and mechanisms for

specifying the connections.

These features can be seen in the static GK and in the PK processing record. The PK state is incremented explicitly in the data base, not hidden in "tags" or "closures". Modular partial results are available for sharing or exploitiation. There are various system aids to debugging our knowledge in action. Paths and traces are helpful. Statically, the GK organization makes organization and options explicit. What we do and how we do it are embodied as nodes. Reading the "code" does not require us to extract elements and interactions from monolithic programs. Suggestion and advice are clear from the links.

SEER "INSTITUTIONALIZES" PROCESSING FOR THE USER BY AUTOMATING FUNCTIONS AND DEFINING FORMS

A system provides different levels of services. Some actions are taken automatically without user concern. Exploitation is the prime example. The system provides functions that the user can employ. Standard forms of organizing and specifying the GK implement other basic user options. Again we see aids to developing the static GK, and system provided processing of the dynamic PK, e.g. monitor decisions on processing sequences. These institutional features help particularly in our initial coding of the GK. The "little man" organization and link structure formalize the transfer of intuitive relationships to program specifications.

THE KNOWLEDGE AND CONTROL STRUCTURES ARE ORGANIZED AROUND NODES

These nodes represent visual phenomena. A node "knows about" its relationships to other nodes, how to establish or exploit itself. Thus I use the term egocentric elements, for the knowledge "chunks", datums (or quantums). Actually, I like the term "little man" processing element. However, the little man concept has been developed more fully by others (e.g. Papert, Hewitt, Kay see the background discussion in [Hewitt and Smith 1975]) and already has connotations. SEER datums may not be quite the independent processes that the little man term could imply to some.

The "little man " or "egocentric element" metaphor encourages the design of a modular knowledge structure and uniform processing activity. Each piece of knowledge functions as a

descriptive element and a processing element, statically as a quantum in the GK, then dynamically, instantiated as a datum in the PK. The most general model is a mathematical or automata theory one where a datum D can initiate or change the state of another datum D'. In practice we have two sets of interactions. When D is executed we have "exploration"; when D is established we have "exploitation".

The exploration and exploitation processes are roughly symmetric. They involve the three basic interactions between datums; initiating, encouraging (affecting priority) and establishing (or affecting the est count). These interactions are effected by "firing" links and acting upon datums.

The exploration and exploitation processes work in a complementary fashion. Care must be taken or some interactions will not be effected and other effects will be duplicated. For example, if D is E-related to D', then an established D' should affect the est count of D. Whether this occurs while exploiting D' or exploring D will depend on the nature of the links between D' and D and the state of PK at exploitation or exploration time. Also SEER must avoid adding to the est count twice.

The various link type combinations that can occur between nodes complicate the exploration and exploitation options. In particular, the absence of an I-link, and possibility of only one node knowing about a relationship, can complicate matters. Then there are the various PK states to consider, the different sequences in which datums may appear or be acted upon. There are also a variety of PK possibilities to which we must respond. Nodes and links may be present or not, established or not, fired or not, when pursued or proposed.

Not all these options are possible or useful. Relationships among the link types dictate some organization and simplification. If D is E-linked to D' it ought to be P-linked as well. We even formalized our notion of I-linkage by dependence on E and P links. Nevertheless some options are useful for technical or semantic reasons or to allow for varied processing flows. Consider, for example, the seed structure illustrated earlier, for a technical use of options to reflect specialized relationships and processing requirements.

A few examples here: In figure 8 - 1 D29 was present when proposed D31 was not, yet D31 was, while D33 again was not. Exploitation "filled in a hole". Also notice how while D27 was present, the link between D31 and D27 was not, and was entered by the exploitation. In figure 8 - 2 both D9 and D10 are present when pursued in exploring D15. D10 is also already linked to D15, while D9 is not,

FIGURE 8 - 1

```
                    CYCLE C25
                    INVESTIGATING D29 (L-AXIS-M1 R196)
                    AT: D30 (L-AXIS-M1-C R196)


                    STATE OF INVESTIGATION
                    OF D29 (L-AXIS-M1 R196)
                    AT START OF CYCLE C25

-> # D29 (L-AXIS-M1 R196) # C
 -> $$ D30 (L-AXIS-M1-C R196) $$ C
 -> D26 (L-AXIS-M1-N R196) S

EXPLORE D30
EXECUTED: 0.29 SECONDS 0.39 REALTIME
STATUS -> S
EXPLOIT D30
<--- D29 (L-AXIS-M1 R196)
     ENTERED: NIL
     FIRED: SE
     -INVES
     STATUS -> S
     EXPLOIT D29
     <--- * D31 (L-AXIS R196)
          +INVES
          ENTERED: (SE SI PUI A)
          FIRED: SE
          -INVES
          STATUS -> S
          EXPLOIT D31
          <--- D27 (W-AXIS-M1-N R196)
               ENTERED: (PUI SI SE A)
               FIRED: SE
               -INVES
               STATUS -> S
               EXPLOIT D27
               <--- * D33 (W-AXIS-M1 R196)
                    +INVES
                    ENTERED: (PUI SI SE A)
                    FIRED: SE
                    DIFF -> 1.0 LIKE -> 1.0
                    PRIORITY -> 1.0
          <--- * D32 (LENGTH R196)
               +INVES
               ENTERED: (PUI SI SE A)
               FIRED: SE
               DIFF -> 1.0 LIKE -> 1.0
               PRIORITY -> 1.0
NEW TOP PRIORITY INVESTIGATION: D33 (W-AXIS-M1 R196)
```

FIGURE 8 - 2

```
                    CYCLE C11
                    INVESTIGATING D15 (SCH-N R196)
                    AT: D15 (SCH-N R196)


                    STATE OF INVESTIGATION
                    OF D15 (SCH-N R196)
                    AT START OF CYCLE C11

-> $$ D15 (SCH-N R196) $$ C
 -> D10 (AREA R196) S <- (D8)

EXPLORE D15
---> D9 (CONVEX-HULL R196)
     ENTERED: (PUI SI SE A)
     FIRED:
---> D10 (AREA R196)
     ENTERED: NIL
     FIRED:
```

and a link is entered. In moving down from D60 in figure 8 - 3 , we need to initiate both D62 and D61. In exploring D19 in figure 8 - 4 , D15 has already succeeded and in fact proposed D19 from below, so we naturally find it present and only need initiate D20. In figure 8 - 5 , in exploring D71 we find D31 is present but is not linked to D71, and we not only enter the link but fire it, as D31 is already established.

Allowing for some generality accounts for the more contorted passages in the exploration and exploitation descriptions. You may have noticed as well that the description given would be inefficient if implemented directly. The descriptions were chosen for pedagogical elegance, implementational ease and flexibility. The implementation, for example, does not always need to enter a link, then check to see if it is there, then act on it. A relationship may be acted on immediately, and no link may be needed for future reference.

The exploration and exploitation sections of SEER were reorganized many times. First it is necessary to identify the distinct relationships, the distinct ways in which one datum can act upon another. Then their interactions and the various options and sequences have to be considered.

It is important to first distinguish the distinct relationships and then explore their interactions. Because of these interactions, e.g. between initiation and establishment, identifying the "atomic" relationships may be difficult. Inadequate analysis will lead to confusion about the options available, in expressing relationships between datums, in proper "bookkeeping" of relationships under various sequences of initiation, execution and establishment. Initiation plays a special role. If D is P or E related to D' but unwilling to initiate it, this is an added complication.

Exploitation reflects the basic heuristic aims of SEER, in using active knowledge to benefit from acquired results. Exploration avoids handing control to any single investigation for any great space. It brings "downward" processing into the same format as exploitation, ties all processing into the egocentric element and suggestion pool mechanisms, and permits constant reassessment of global strategy for control flow decisions.

It is important to remember that while these egocentric elements have a rather independent existence, none of them retains "control" for very long. The strategy that produces the processing sequence remains under global control. These elements also interact globally. The uniform node

FIGURE 8 - 3

```
                    CYCLE C45
                    INVESTIGATING D49 (IS-A-H-M1 DR1)
                    AT: D60 (HAS-THE-HP2-M1 DR1 R140)


                    STATE OF INVESTIGATION
                    OF D49 (IS-A-H-M1 DR1)
                    AT START OF CYCLE C45

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> D53 (HAS-A-HP2-M1-T DR1) C
     -> # D59 (HAS-THE-HP2 DR1 R140) # C
      -> $$ D60 (HAS-THE-HP2-M1 DR1 R140) $$ C
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D60
---> * D62 (H-REL-P2 DR1 R140)
     ENTERED: (PUI SI SE A)
     FIRED:
---> * D61 (IS-A-HP2 R140)
     ENTERED: (SE PUI A)
     FIRED:
```

FIGURE 8 - 4

```
                    CYCLE C16
                    INVESTIGATING D19 (SCH R196)
                    AT: D19 (SCH R196)


                    STATE OF INVESTIGATION
                    OF D19 (SCH R196)
                    AT START OF CYCLE C16

-> $$ D19 (SCH R196) $$ C
 -> D15 (SCH-N R196) S

EXPLORE D19
---> * D20 (SCH-C R196)
     ENTERED: (PUI SI SE A)
     FIRED:
---> D15 (SCH-N R196)
     ENTERED: NIL
     FIRED:
```

FIGURE 8 - 5

```
CYCLE C53
INVESTIGATING D49 (IS-A-H-M1 DR1)
AT: D71 (PERP-N R196 R140)


STATE OF INVESTIGATION
OF D49 (IS-A-H-M1 DR1)
AT START OF CYCLE C53

-> D49 (IS-A-H-M1 DR1) C
 -> D51 (HAS-A-HP2 DR1) C
  -> D48 (HAS-A-HP2-M1 DR1) C
   -> D52 (HAS-A-HP2-M1-GT DR1) C
    -> D54 (HAS-A-HP2-M1-G DR1 A1) S
    -> D53 (HAS-A-HP2-M1-T DR1) C
     -> D59 (HAS-THE-HP2 DR1 R140) C
      -> D60 (HAS-THE-HP2-M1 DR1 R140) C
       -> D62 (H-REL-P2 DR1 R140) C
        -> D63 (H-REL-P1-P2 R196 R140) C
         -> D64 (T-JOINED R196 R140) C
          -> D66 (TOUCH R196 R140) C
           -> D67 (TOUCH-M1 R196 R140) C
            -> D69 (TOUCH-M1-N R196 R140) C
             -> # D65 (PERP R196 R140) # C <- (D64)
              -> $$ D71 (PERP-N R196 R140) $$ C
              -> D70 (PERP-C R196 R140) C
            -> D68 (TOUCH-M1-C R196 R140) C
          -> # D65 (PERP R196 R140) # C <- (D69)
       -> D61 (IS-A-HP2 R140) C
   -> D47 (HAS-A-HP1 DR1) S <- (D49)
 -> D47 (HAS-A-HP1 DR1) S <- (D48)

EXPLORE D71
---> * D72 (L-AXIS R140)
     ENTERED: (SE PUI A)
     FIRED:
---> D31 (L-AXIS R196)
     ENTERED: (SE PUI A)
     FIRED: SE
     DIFF -> 4.81
```

structure facilitates sharing. The datums function as a data base of "assertions". Results are not limited to responding to the investigation or datum which provoked them. They are free to exploit themselves in other contexts.

SEER IS EXTENSIBLE

SEER legitimizes "hacking" in some sense. A special method for boundary finding, for bars, or employing symmetry, is a hack in a system that implicitly deals only with an environment of bars or symmetric objects. However, in SEER special knowledge can be integrated into a general, extensible system. It is particularly necessary to be able to add to the GK incrementally. Methods and paths through the GK need not be discovered and specified all at once. The modular, explicit nature of the GK organization and the straightforward link formalism for interactions permits incremental development.

For example, I first encoded the path that led from a handle recognition on to a hammer recognition. Only later did I expand the hammer GK to allow a recognition path that found the head first. This essentially just involved putting in exploitation links to generate a hammer conjecture from the head datum. However, as there was also difficulty in getting the handle from the affinity tree at that point, I also added a new method for finding a handle, which easily linked to the handle quantum.

SEER also organizes higher level decisions into system priority and monitor modules. The user cannot program into every element or investigation an ability to direct the processing sequence appropriate to the scene. "Modular" is a slippery word. SEER's extraction of priority and monitor functions modularizes decision making at the system level. However, it makes each element less modularly independent (though individual monitor modules are a possible overlaid mechanism). SEER, here as elsewhere, tries to take the best from conflicting good ideas. It implements modularity at a low level when this helps make the system incremental and explicit; it provides general system functions, when they help further that goal.

The incremental nature of SEER permits us to add new advances. The control requirements that guide the GK structure should permit other advances in descriptive formalisms to be implemented as well. At the lowest level new visual functions and predicates can easily be assimilated. New

methods and interactive possibilities are possible. The top level modularity of system functions facilitates experimentation with issues like priority and learning.

SEER is well suited to function as a laboratory. SEER provides the interactive formalism. Students can study individual visual problems and still have a means of organizing the knowledge they develop to fit into a working system. This permits study and use of the potential of cooperating areas of knowledge. I was easily able to use Lozano's region finder, for example, providing it with just the "advice" it needed for input.

IN OPERATION, THE SYSTEM TRIES TO TAILOR THE PROGRAM TO THE SCENE

We discussed this point as a control issue in the previous chapter. Basically it comes down to intelligent organization. If we try to write one big program that will recognize hammers, under all contingencies, the complexities will overwhelm us, and any individual results will not be properly available to a general system. A system formalism can tell us how to organize the hammer knowledge and help us use it. SEER allows us to specify interactions explicitly and then operates on them and evaluates the results automatically to dynamically direct processing. We have seen SEER's flexible response to the scene in operation at several levels from low level property acquisition to broad structural flows. In figures 8 - 6 and 8 - 7 we see two recognition paths for two conjectures about a hammer-part-head. These were taken from the example in chapter one. In one we proposed the conjecture from below after establishing part of it, then pursued the rest, but failed. In the other we pursued the conjecture from above, then moved downward to establish it.

SEER EMBODIES MODELS FOR VISUAL PROCESSING, WHICH PROVIDE A CONTEXT FOR OUR WORK

It is important to have a system where one can experiment with and evaluate such models. They provide a coherent structure. Processing can be motivated by general principles, rather than ad hoc individual decisions. We need a "theory". Ideas like heterarchy provide general intelligent processing metaphors. We also need specifically visual models.

"SEER" proceeds by "looking ahead". PK, as acquired, triggers and reinforces possibilities; it hypothesizes expectations based on contextual and relational information imbedded in GK. The order in

FIGURE 8 - 6

```
                    PK STATE
                    AT END OF CYCLE C262
                    BELOW D260 (IS-A-HP2 R196)
                    TO DEPTH 8.
                    SHOWING THE A PATH

-> D260 (IS-A-HP2 R196) NIL
 -> D50 (IS-A-HP2-M1 R196) 196.
  -> D233 (HAS-A-HP2P1 R196) 197.
   -> D234 (HAS-A-HP2P1-M1 R196) 198.
    -> D235 (HAS-THE-HP2P1-M1 R196 R145) 199.
     -> D237 (HAS-THE-HP2P1-M1-CE R196 R145) 200.
      -> D240 (COVER-END R196 R145) 209.
       -> D251 (COVER-END-N R196 R145) 210.
        -> D245 (O-BDRY R145) 204.
        -> D12 (O-BDRY R196) 5.
        -> D243 (AREA R145) 203.
        -> D31 (L-AXIS R196) NIL
       -> D250 (COVER-END-C R196 R145) 218.
      -> D239 (CONVEX R145) 211.
       -> D248 (CONVEX-N R145) 212.
        -> D253 (CONVEX-HULL R145) 213.
        -> D243 (AREA R145) 203.
       -> D252 (CONVEX-C R145) 217.
      -> D238 (HAS-THE-HP2P1-M1-CE1-M2 R196 R145) 201.
       -> D242 (HAS-THE-HP2P1-M1-CE1-M2-N R196 R145) 202.
        -> D243 (AREA R145) 203.
        -> D10 (AREA R196) 4.
       -> D241 (HAS-THE-HP2P1-M1-CE1-M2-C R196 R145) 208.
     -> D236 (HAS-THE-HP2P1-M1-CF R196 R145) NIL
 -> D44 (IS-A-HP2-W R196) NIL
  -> D43 (IS-A-BAR R196) NIL
   -> D28 (IS-A-BAR-M1 R196) NIL
    -> D41 (BAR-DIMENSIONS R196) 30.
     -> D42 (BAR-DIMENSIONS-C R196) 31.
     -> D39 (BAR-DIMENSIONS-N R196) NIL
      -> D32 (LENGTH R196) 28.
       -> D40 (LENGTH-C R196) 29.
       -> D31 (L-AXIS R196) NIL
      -> D38 (WIDTH R196) NIL
       -> D36 (WIDTH-M1 R196) 26.
    -> D25 (HAS-BAR-SIDES R196) NIL
     -> D23 (HAS-BAR-SIDES-M1 R196) 19.
      -> D24 (HAS-BAR-SIDES-M1-C R196) 20.
      -> D22 (HAS-BAR-SIDES-M1-N R196) 18.
       -> D12 (O-BDRY R196) 5.
       -> D21 (SCH-AS-DESC R196) 16.
```

FIGURE 8 - 7

PK STATE
AT END OF CYCLE C262
BELOW D61 (IS-A-HP2 R140)
TO DEPTH 8.
SHOWING THE A PATH

```
-> D61 (IS-A-HP2 R140) 73.
 -> D96 (IS-A-HP2-M1 R140) 74.
  -> D98 (IS-A-HP2-W R140) 75.
   -> D99 (IS-A-BAR R140) 76.
    -> D100 (IS-A-BAR-M1 R140) 77.
     -> D77 (HAS-BAR-SIDES R140) 55.
      -> D78 (HAS-BAR-SIDES-M1 R140) 56.
       -> D80 (HAS-BAR-SIDES-M1-N R140) 57.
        -> D81 (SCH-AS-DESC R140) 58.
        -> D76 (O-BDRY R140) 62.
       -> D79 (HAS-BAR-SIDES-M1-C R140) 86.
     -> D101 (BAR-DIMENSIONS R140) 78.
      -> D103 (BAR-DIMENSIONS-N R140) 79.
       -> D105 (LENGTH R140) 80.
        -> D72 (L-AXIS R140) 52.
        -> D106 (LENGTH-C R140) NIL
       -> D104 (WIDTH R140) 81.
        -> D107 (WIDTH-M1 R140) 82.
      -> D102 (BAR-DIMENSIONS-C R140) NIL
  -> D97 (HAS-A-HP2P1 R140) NIL
```

which SEER processes the possibilities it envisions depends on what it expects to see, what it is interested in seeing. Also on how hard things are to see, how much effort is involved.

First the most promising investigation is chosen; then SEER tries to verify or eliminate alternatives most efficiently. Results reverberate in non local ways among alternative lines of inquiry. SEER continuously makes a global reassessment of priorities to pursue winning lines. We have described the processing flow structures that can result from this model. Alternatives should be considered, and evaluated; this process has only begun, in developing SEER.

The model allows us to specify, predict or observe behavior. For example, it would seem reasonable that a good visual system, particularly one with SEER's biases, should actually "see" less in easily recognized scenes. That is, if a hammer stands out clearly in a scene, and the goal, as in SEER, is simply recognition, details of the hammer may be ignored, actually not "perceived".

We would in fact expect SEER to jump quickly to hammer hypotheses and verify them easily in this situation. In a more obscure scene SEER might spend more time at the lower level looking for evidence to suggest the hammer. Other suggestions might deserve attention. Higher level suggestions might not acquire enough confidence initially to prefer them to further low level investigations. More evidence may be required to verify the correct hypotheses. Thus details of shape, irrelevant observations on texture or stains, may be noticed.

We have seen a small demonstration chapter six. Faced with an unoccluded sledge hammer, where the handle stands out clearly, SEER quickly recognizes the handle and goes on to propose and recognize the hammer. In another scene in which the handle is occluded, a piece of the handle is studied first, but it is obviously not a handle by itself. SEER considers the possibility that it is a head, rejects that. SEER does not move up to the hammer hypothesis until later, after properties of another piece of the handle have been studied, and finally this piece combines with the first to constitute a possible handle recognition.

Precisely how SEER reacts depends on the GK and the scene, and in large measure on the priority and monitor systems. We can evaluate whether the mechanisms implement our models, and if the models function effectively.

9

RECOGNITION TASKS

SEER ATTEMPTS TO DEAL WITH THE VARIETY OF VISUAL REALITY

Its task is recognition of visually perceived phenomena, i.e. of regions, properties, relationships and ultimately objects. Visual variety effectively multiplies a single such item into a number of recognition tasks. Hammers may be black or brown, wooden or metal, ball peen or sledge. A single hammer may appear in shadow or spotlight, in side view or end on. A perverse lighting may transform a simple hammer recognition to a difficult one.

The greater the variety of appearance of a single item, the more difficult is its recognition. We must either have general definitions and mechanisms that can encompass this diversity, or a variety of specific approaches for different cases. Some specific examples of an item can be particularly difficult to handle. Or an external factor can transform an easy task into a difficult one. A straightforward example of a hammer can be obscured by heavy shadows from an unfortunate light source. Even a single item, then, can introduce the problems attendant on dealing with a large number of recognition tasks, in particular choosing which processing steps to take out of the many options available.

The converse problem to that of identifying a lot of different appearances as the same item, is the problem of distinguishing different items which appear quite similar. A paper cut out of a cylinder may appear quite similar to a cylinder, but hopefully enough surface shape or depth information can be inferred to reject the cut out as a cylinder. Again, this problem is present even in dealing with a single item. We may not know about wrenches, but still have to distinguish them from hammers. (In some ways adding knowledge of additional items makes things easier. Identification of an item as a wrench is a useful way of ruling out a hammer recognition. And, of course, choosing between a specified set of possible identifications, one of which must hold may be easier than saying yes or no to the bald question: is it that item?)

Of course, increasing the variety of the items SEER understands is important. However, our

first concern has been with increasing SEER's ability to deal with the effect of visual variety on individual items. There are clearly special problems that arise when the number of possible recognitions becomes very large. It is quite possible, however, for even a system with few items to cope with, to be unable to cope with them under realistic conditions.

The problem is not one of being unable to describe the items. It is a simple matter to describe most objects in a manner that will distinguish them from most others. The problem lies in being able to put this description to work on real data. In other words the problem is in describing the data. And here it is important that all levels of an item's description, and all other areas of knowledge, be able to cooperate at any level in analyzing the data in order to perceive the item: in other words active knowledge.

Many problems can be viewed as various forms of "distortion". A shadow, an occlusion, a difficult orientation all produce variations in a small set of "ideal images" of an object. These problems each require special knowledge, but some general techniques should apply to all of them. SEER tries to provide some of these. Viewing visual problems as a distortion "function" encourages a search for "fixed points" and heuristic or symbolic interpolation techniques between standard or known positions.

An alternative model of visual input attacks visual variety as discrete sums of alternatives. SEER facilitates both input programming of such alternatives, and processing time application. SEER integrates alternatives and responds appropriately to each "processing context". The context of a processing decision varies with the scene being studied and the state of previously acquired knowledge.

A MASS OF UNDIFFERENTIATED SENSORY DATA IS RECEIVED IN PARALLEL

The most fundamental problem faced in visual recognition is the nature of the sensory input. I compare the visual recognition task to the problem of understanding a spoken conversation at a noisy cocktail party. What do we look at? How do we segment the data; what are the "words" whose meaning is to be sought. What are the important regions and features, what deserves attention and in what order? What do we ask about?

There are the usual technical "noise" problems. Fundamentally, there is a problem of

"significance". What is noise in the data, or a real but unimportant feature of the scene, and what is a feature to which we want to pay attention? Many low level problems are "isomorphic" to a paradigm problem of determining significant segments in a "graph" of a function. Shape problems especially, surface and boundary shape, can be viewed in these terms. We can have a plot of a function computed along a line in the surface. The outer boundary is already a curve. (It can be plotted as a single valued function of distance from the center of gravity, if you like.)

It is often not possible to get proper results from low level processing without introducing a global element. It is really a question of making a low level pass at all the data before higher level intervention. In many cases we cannot truly make a limited low level observation of part of the data, without involving a certain amount of global processing.

The reason is that we cannot distinguish subordinate effects from primary ones at a local level. Consider figure 9 - 1 . Part a shows a graph of let us say intensity values along a line through the scene. Is there an edge at x? This depends first on whether the intensity increase around x is significant. But "significant" can vary from region to region or scene to scene. Suppose for example that the graph in part a is part of the extended view indicated in part b. Or the feature at x may be significant, but part of a regular textural pattern as in part c of the figure, and not indicative of a break between regions.

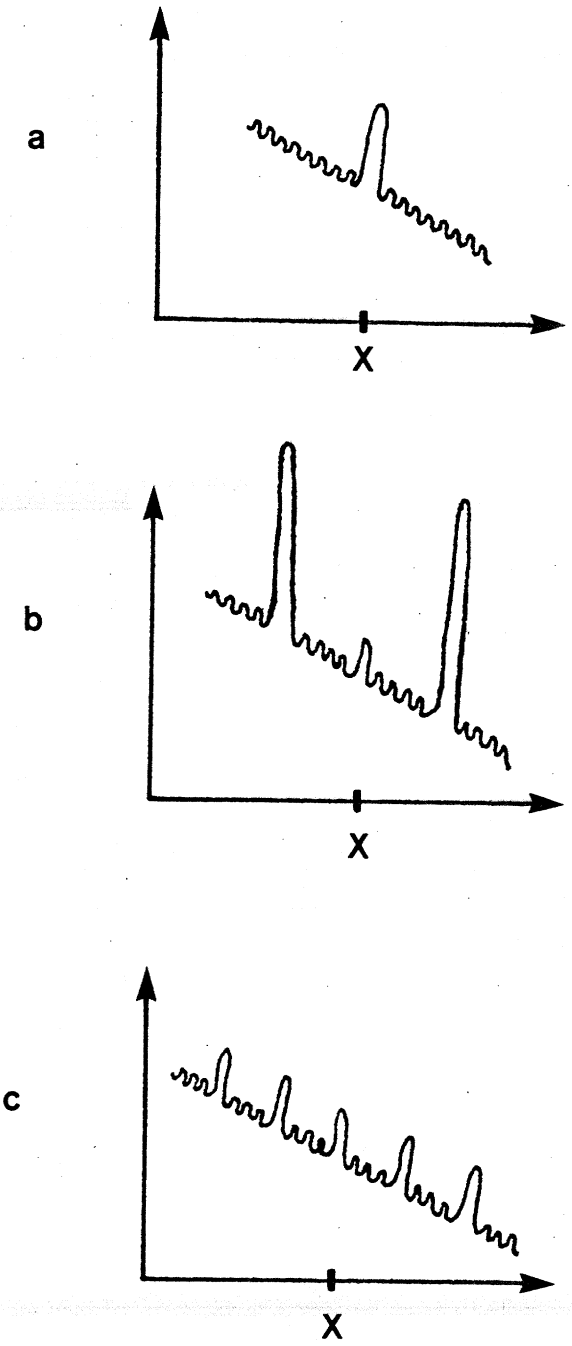
Furthermore, the feature may be large but merely indicative of noise or a reflection, or a surface imperfection. Techniques that iterate up to broader "supports" like affinity or vexity structure analysis, can be applied to different problems that can be formulated within this function graph paradigm.

THE RESPONSE TO VARIETY MAY BE TO DEVELOP GENERAL METHODS AND DEFINITIONS THAT ARE BROAD ENOUGH TO ENCOMPASS OR IGNORE DIFFERENCES

However, a general definition of hammer that is broad enough to cover sledge hammers and claw hammers may have difficulty avoiding inclusion of other objects. This general definition may still be useful in leading us to recognition of a limited class of objects. From there it may be easier to decide which of the class we have. On the other hand it is a bit surprising, perhaps, when you think of

FIGURE 9-1

SIGNIFICANCE--GLOBAL/LOCAL



it, how a very crude definition that will encompass almost any hammer, really seems to cover very little else. This may have something to do with the inherently functional nature of many definitions: one piece to hang onto and a crosswise piece at the end to bang with, suitable for a hammer and little else.

Flexible techniques play a role as well. A good technique may be able to recognize features despite a range of realistic problems. A general method that can compensate for or ignore irregularities and variations is not to be spurned.

There is a danger that we can get carried away "perfecting" individual techniques. SEER rather emphasizes ways to utilize imperfect techniques, especially where the nature of the data may make a single ideal technique impractical or impossible. Imperfect techniques may receive higher level guidance or different methods can be advised appropriately by context. On the one hand, this leads in the direction exemplified by line proposing, where a general imperfect method is improved by a critic. We also can consider limited methods that only deal with aspects of the problem.

AN ALTERNATIVE APPROACH ORGANIZES A TASK INTO SUBCLASSES AND SPECIAL CASES WHICH CAN BE HANDLED SEPARATELY

After identifying crude properties of a hammer, we could require that it be identified specifically as ball peen or sledge hammer before we were certain it was a hammer. Or if an object were either rectangular or round, but never any other shape, we would be hard put to characterize its shape property in any general way, but would merely specify: round or rectangular.

Special case handling becomes most useful, though, when different views of an item or extrinsic factors like lighting are involved. A hammer viewed from different orientations, changes appearance. We may have orientation techniques that can operate on, or be incorporated in, our definition of a hammer. But a hammer definition may need distinct subnodes in its knowledge structure analysis that deal individually with different views. These alternatives call for the sort of guidance and cooperation SEER interactions can provide.

THERE ARE HYBRID APPROACHES

The "critic" strategy might be viewed as one. Heterarchical interruption and interaction of one program level with another can combine general methods with specific advice. There have been attempts to provide general descriptions or methods with specific advice. There have been attempts to provide general descriptions or methods from which specific cases can be computed, e.g. matrix descriptions of three dimensional objects produce two dimensional projections [Roberts 1963].

There is also a possibility of heuristic projection or interpolation. A number of specific models of a phenomena may "cover" the range of variation sufficiently under some general method of working from them to accept "distorted" images. General observations, about shadows, for example, may require computation upon specific phenomena to produce results for individual cases. We may wish to store some results in "compiled" or fully computed form.

I will review the various factors that cause visual variety, indicating briefly how the general approaches and seer mechanisms we have discussed can be brought to bear. Some of this discussion is tutorial and some speculative.

INTRINSIC FACTORS INCLUDE THE VARIETY OF DIFFERENT EXAMPLES AND ORIENTATIONS OF AN OBJECT OR FEATURE

Different examples may have different texture, shape, etc. Different orientations cause distortions, present different pieces of an object to view. These problems arise below the object level as well. There are many examples of "fuzzy texture", and a single example may vary in different orientations. My emphasis for different examples has been on general definitions and techniques that are crude enough or flexible enough to deal with this variety. For different orientation, I exploit SEER's aptitude for modular special case programming and for partial results advising other investigations.

DIFFERENT EXAMPLES

The amount of detail we require to recognize objects (or features) is remarkably low. There have been experiments in this direction. It is rather obvious that many objects are overdefined, i.e. many subsets of their properties suffice to define them uniquely. On the other hand, completeness and

detail of description may define a particular example of an object to the exclusion of other examples that should be similarly identified. We have every reason then to be frugal in making our definitions, adding detail only as we find necessary to distinguish an object from others. Redundancy should still be used to provide alternative descriptions to make recognition easier when one set of features are obscured in some fashion.

How fine or crude a recognition is sufficient often depends upon context. A hammer head alone requires more detail to verify than a head in its proper place on the end of a handle. It therefore is expedient to work "in context" as much as possible.

A fine recognition may be necessary to suggest an appropriate context: The head seen in detail, the handle then viewed cursorily to verify the hammer. However, even a crude recognition may trigger a conjecture that pursues other crude objectives to satisfy a larger recognition. A handle suggests a hammer without verifying the subtle shape that confirms a hammer hanldle. The hammer finds a crude head in the proper relationship to the handle. The crude pieces together with the proper relationship would be enough to satisfactorily perceive a hammer.

A crude definition of an object may consist of a larger class to which the object belongs: handles comprise hammer handles, screwdriver handles, etc. Or it may be a definition of a specific object only, merely less detailed in some fashion. A recognition of an object as a handle, can trigger several suggestions to refine the definition as a hammer handle, screwdriver handle, or whatever. Having first found the general class may prove helpful in advising how to distinguish the specific member.

As we noted however, handle suggests a search for hammer directly. This may lead to a resolution more easily than proceeding to identify the hammer handle, as such, directly. Fulfilling a crude definition of hammer may be easily accomplished, and then the handle must be a hammer handle. Finding (or defining) the subtle shape that evokes hammer handle in a given scene may be more difficult and unnecessary.

I tend to believe people recognize most objects by quickly conjecturing their identity from a few features, and then easily verifying a few more, often without needing detail. A car goes by; afterwards we know we have seen a car, but what shape exactly were its windows? Of course, if they

were unusual, our complacent verification of windows is upset and we stop and notice more detail.

SEER uses relative and global means to increase the ability of its mechanisms to cope with reality on their own, to help determine significance and deal with noise and variety. SEER is also "conservative" in its decision making, avoiding decisions until further context and higher level guidance can be brought to bear. A prime example of all this, of course, is the affinity mechanism. The use of this mechanism for region finding and surface shape determination has been analyzed in chapter five. SEER's calculation of boundary shape also illustrates use of relative and global principles and postpones final decisions at a low level, leaving organized data available to higher levels.

DIFFERENT ORIENTATIONS

Significantly different views result from altering the orientation of an object with respect to the viewer. Entire regions may appear in one view and not another. More subtly, features of an object may change character or be distorted. A round shape can appear elliptical, and the length of the minor axis of this ellipse can be arbitrarily small. Textural appearance can be altered.

SEER is well situated to break multiple views into special case approaches that can be activated independently, while sharing features whenever possible. Changes in the appearance of features can be handled in part by relying on invariants that orientations do not affect. Changes in orientation cause changes in our view of different parts or features of an object, but as these aspects of the object are related so will be the view changes. The relationships may enable us to work from a recognition of one feature to recognition of another. This type of interaction is SEER's forte. In other words we can recognize one feature, and by judging how it has changed from some canonical norm, judge how other features should appear. Then once again we are in verify mode.

This approach is particularly useful when the orientation makes one feature easily recognized and the other difficult to perceive, easily confused, or undistinctive. In fact features may well be related so that the more one is distorted, the clearer is the other, and vice versa. The obvious example is two flat planes set at right angles. We may be able to arrive at a set of "canonical positions" for two features, such that at least one feature is in a canonical position in any orientation, and from it we can determine what the other should look like. That way we only need to learn to look for the limited set of canonical positions. The others can be computed and verified. One feature may

have a range of appearances difficult to check for anyway. It may be easier to determine what it should be and then verify it.

These relationships in appearance of features under orientation changes may be expressed by a set of corresponding pairs or a function to operate on one feature and produce the other. The relationship between the two may be expressible as an invariant under orientation. Thus a discrete set of possible views (within some tolerance) may cover the range of possibilities for a feature. Or a function may continuously compute the possibilities, especially when the function is easy or the range hard to describe discretely.

Of course we can go up a level. Recognition of one or more features can conjecture the orientations of the object. This orientation information can then work directly to compute appearances under distortion or specify among sets of possible views. This high level approach can also be used to compute or pick views when we are actually seeing different features, i.e. "sides" of the object, as opposed to having features altered by the view. This approach goes back to Roberts, of course [Roberts 1963]. While mathematical transformations of Robert's sort are elegant, I suspect that the imprecise nature of most definitions of objects and features does not lend itself to such a general approach. On the other hand much of this discussion of alternatives is very speculative at this point.

EXTRINSIC FACTORS INCLUDE MATTERS OF CONTEXT

I use the term "extrinsic" because "context" has become rather overworked and thus diffused in meaning. SEER provides mechanisms that encourage its programmer to say:

This recognition is difficult or impossible to do in a general way that can deal with the variation that these factors can cause. Can I program special cases that SEER can integrate into a system? Is there anything else the system might know about that it could use to advise or assist this recognition?

We should direct some attentions to specific extrinsic factors that affect the recognition of an object. The most prominent of these are the effects of other objects in the scene and of lighting.

OBJECTS

Objects in a scene may obscure (occlude) or abut each other. They may reflect on one

another or shadow one another. Background may blend with some objects or highlight others. Features are always present in combination. A complex boundary shape may make computation of an axis difficult. Perception of surface shape or recognition of regions may be confused by textural patterns.

SEER has two major items in its arsenal to deal with such problems First, active knowledge. It expects that background information will want to advise region finding, or early success with unoccluded objects will be available to help floundering recognition efforts. All such efforts need not be made independently in a single pass. Neither do we need to know which objects are unoccluded, to try them first. Such a process can easily become circular; do we need recognition to help judge occlusion? SEER pursue results in parallel; as success is obtained it is available to advise other investigations.

Secondly, SEER is used to dealing with partial results. It is designed to work up from pieces. Our straw man model matching system balks at occlusions because it has a model of an entire object that it expects to match. SEER expects to find pieces and suggest the presence of wholes from them. If wholes are not there then occluded wholes may be.

LIGHTING

Lighting is a diffuse subject. There are a number of lighting factors, that cause a number of effects, that affect various features in the scene. Direction of lighting causes shading or shadows that affect the appearances of surfaces: changing textural appearance, altering the affinity organization into regions. Multiple light sources, uneven lighting and the like can affect so many aspects of a scene that the views of an object under different lighting conditions can be as distinct as views under different orientations.

While these affects of lighting must be overcome in some sense, we must also remember that light is our only real source of data. Even seemingly obstructive lighting effects may in fact aid identification if properly understood. Waltz demonstrated this with shadows in [Waltz 1972]. Utilization of features like hilights has long been advocated. Many of our perceptions of shape and texture are closely related to characteristic responses to light, shading, reflections [Horn 1970]. (Note the work of the "superrealist" school of painting.)

We must utilize active knowledge to integrate general knowledge about lighting and lighting results, such as shadows, into object recognition. And our object knowledge may include special purpose knowledge about the lighting effects on the object, likely results, even full scale special case recognition methods for common lighting conditions. This knowledge of the objects can react to information about, or advice from, lighting conditions.

We finish this chapter by reviewing the central problem of region generation.

IS NOT THE AFFINITY PROCESS MERELY A DISTINCT "PASS" IN THE BAD OLD TRADITION OF THE "HORIZONTAL" VISION SYSTEMS?

Certainly it has many of the properties of a horizontal pass. Affinity accepts a description of the scene as input and processes it all independent of any interruption or advice from other levels of the system. A new description of the scene is produced as input for the next phase of the system.

I will make two arguments. First, it is misleading to stop our analysis here, and condemn the affinity mechanism as anti-heterarchical. In fact, when the issues at this level of processing are understood, affinity will be seen to support key aspects of our heterarchical methods. Second, we should be aware of some legitimate purposes which a pass structured process may serve at this level.

The other levels of a horizontal system can only interact with the output of a line finding pass. In SEER subsequent analysis can still interact with the affinity process. A complete trace of this process remains in the form of the affinity tree. It is this trace on which higher levels operate. In this respect heterarchical interaction with the affinity mechanism is still possible.

It is worth noting also that subsequent processing can even go back beyond the output of the affinity pass to the original data input. We are not at all locked into the head to tail elephant walk of absolute horizontal processing. Nevertheless, no other levels interrupt the affinity mechanism to direct its results. Lack of interruption, however, is hardly significant here because the aims of the affinity process are so limited.

A line finder in a horizontal system is expected to be "complete", i.e. to produce all the real lines in the scene, and only those lines. It is difficult, if not impossible, to attempt all this without higher level guidance. Shirai's system [Shirai 1972], on the other hand, looks first at outer boundary

lines. A pass is made to find these lines. However, they are the easiest lines in the scene to find. Shirai can move from them to the interior lines with heterarchical assistance.

Similarly the regions produced by the affinity tree are not intended to be a complete set of (semantically) significant regions for the scene. Furthermore, no claim is made that all of the regions that are found are significant. The affinity results also do not need to present a "final" description of a region. Details and precise locations can come later. The more incomplete and tentative the initial phase is, the more opportunity for subsequent heterarchical interaction, informed by developing results. However, we still must provide a sufficient base from which to start.

It is also important that the higher levels of the system be flexible in the initial regions which they can accept. A system where the first items found are intended to be a complete and final representation of the regions of the scene is easily confused by the variety of reality. In [Barrow and Popplestone 1971] for example, the recognition level expected a hammer to have two regions, handle and head. When head blended with handle or with background, the results could not be dealt with properly.

SEER can take the handle region alone and work from it to propose a search for the head. (In a specified region at the appropriate end of the handle.) Or it can begin with the entire hammer, and pursue sections of it, as suggested handle and head. A crude overview of the entire scene can serve as basis for "planning" [Kelly 1971].

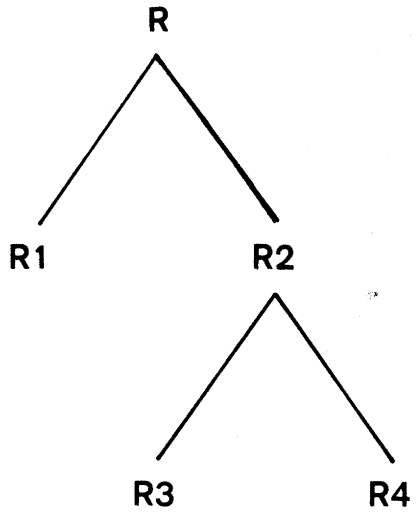
Some syntactic measures may be tried for estimating which regions are most likely to be interesting, and these can be explored first. For example, suppose 9 - 2 represents a portion of the affinity tree. If R3 and R4 were much more alike than R1 and R2, this might encourage us to look at R2. Unfortunately, my brief experimentation with such syntactic criteria, even at this level, has not been encouraging.

However, we can prune the tree. If we are willing to assume that objects of interest lie completely within the picture we can eliminate regions that continue "out of" the scene into the border. If a region consists of two subregions, one of which is quite small, we may assume that these subregions do not represent important distinct region groupings.

The affinity process might be repeated, in specified areas, with a finer initial cell size on the

FIGURE 9-2

AFFINITY TREE SEGMENT



basis of results obtained from the initial crude tree. An overview may prove a useful balance against the dangers of over presumptious heterarchy. Yakimovsky and Feldman in [Yakimovsky and Feldman 1973], for example, use higher level expectations very early to prune the tree of region growing possibilities. There is a danger of getting stuck out on the wrong limb after having sawed off the rest of the tree, to conjure up the classic cartoon analogy.

It helps to "know what you are looking for"; however, we must not be overly subject to the trap of "seeing what we expect to see". (No more than people are, at least. To an extent this "deficiency" may be symptomatic of a basically efficient visual processing model.) The affinity tree provides an independent overview. Independent starting points may contradict earlier conjectures. In any case we have a good base to fall back on if we lose our way further along.

A pass over the entire scene at first also reflects our attention to proper use of both local and global modes of operation. The levels of the tree reflect a range of scene "support". The lower levels encompass a very local view, the higher levels attempt to reflect the global appearance of the scene. In very practical terms, as discussed earlier, it is often not possible to get proper results from low level processing without introducing a global element.

SUBSEQUENT REGION GENERATION REFLECTS USE OF ACTIVE KNOWLEDGE

SEER requires regions as it develops the PK structure. Given one element of a relationship, SEER seeks the other, part/whole relationships for example. Given a part SEER proposes the whole; to establish a whole, SEER may need to recognize the part. In the initial affinity pass SEER seeks regions to ask "what is this?", specifically: what properties does it have; does it have this property? Subsequently regions are sought as arguments for suggestions. The distinction between "what" to ask and "where" to look blurs.

Rather than seeking the properties of a region, we seek a region with desired properties. "Knowing what to look for" can help us find it. We may have other information to help direct our search, as we discuss below. In other words semantic, active knowledge participation can assist our region finding. It also may be more efficient to seek regions to meet motivated needs.

Generating a region to "satisy" a suggestion, involves generating "candidate" regions. Several

suggestions may be initiated, in parallel or serially. Where do these candidates come from? They can, of course, be sought in the affinity tree. We can move down in the tree until we find a region with the right properties (including location in the scene). If we are seeking a part or whole, given a known region R, we can move up or down the tree from R.

This process can be speeded up by simplifying assumptions. Of course, we can ignore already identified regions. We assume that objects of interest are entirely within the scene, and so can ignore regions bordering the boundary of the picture. Regions composed of a very few cells may be too small to be of interest. In specific cases other simplifications may be justified. The context of the investigation will determine what we can get away with. Suppose a region R is composed of two regions, R1, R2, and R1 is very small. We may feel justified in assuming R1 is of no independent interest, while R and R2 are essentially the same and may both be represented by R.

Search in the tree can vary with the cause for search. Searches through the whole tree are "cruder" than searches within a specified region, e.g. within a head region for a face. Figure 9 - 3 shows a crude tree from the top, for the ball peen hammer scene discussed in chapter one. Figure 9 - 4 is a finer tree within region 196, the hammer head in that scene. In this particular instance we only ran the affinity program over part of the scene to save time and space. It was clear that a reasonable cell size would not find the handle properly anyway. In general we run the affinity pass over the full scene, though we could run a very rough pass and use it to guide finer passes in limited areas.

Candidates can also be culled from the data base. For example, if we are pursuing a hammer and have previously identified a handle, we might want to check if it will serve as a hammer part now (unless it has already been used for a part elsewhere).

We may need to generate an entirely new region, not already in the affinity tree or data base. SEER can generate a "dummy" region, which will be defined by its properties or parts. Figure 9 - 5 shows the generating function, "gen-dum", generate dummy region, on link L107, that generates D47 in cycle C33, shown in figure 9 - 6 . We can also generate a "real" new region. It may be distinguished as a part or superset of a previous region, or be totally fresh. The manner and extent to which it is perceived, described and named will depend on how it is found. We could set up a new region, with a full boundary description, for example, or merely verify that certain properties distinguish one end of a

FIGURE 9 - 3

CRUDE AFFINITY TREE

```
R199 120. 448.  <R199 is the whole area processed>
   R196 10. 480. <R196 is below R199 in the pruned tree.>
   R140 6. 451. <R140 is comprised of six of the initial cells.>
   R185 6. 444. <R185 has an average intensity of 444.>
   R191 14. 467.
```

FIGURE 9 - 4

FINE AFFINITY TREE BELOW R196

R196 10. 480. <R196 is the hammer head shown in chapter one.>
  R145 3. 307. <R145 is the striking face of the hammer head.>
  R169 7. 554.
    R144 3. 561.
    R142 3. 556.

FIGURE 9 - 5

```
GK SEGMENT
ABOVE Q75 IS-A-HP1
TO DEPTH OF 1.
SHOWING PROPERTY EXPLOIT

<- Q75 IS-A-HP1 (L107 (GEN-DUM) (SE SI))
 <- Q79 HAS-A-HP1 (L93 (I) (PUI SI SE A)) (L65 (I) (PUI SI SE A))
```

FIGURE 9 - 6

```
                    CYCLE C33
                    INVESTIGATING D41 (BAR-DIMENSIONS R196)
                    AT: D42 (BAR-DIMENSIONS-C R196)


                    STATE OF INVESTIGATION
                    OF D41 (BAR-DIMENSIONS R196)
                    AT START OF CYCLE C33

-> # D41 (BAR-DIMENSIONS R196) # C
 -> $$ D42 (BAR-DIMENSIONS-C R196) $$ C
 -> D39 (BAR-DIMENSIONS-N R196) S

EXPLORE D42
EXECUTED: 0.04 SECONDS 0.13 REALTIME
STATUS -> S
EXPLOIT D42
<--- D41 (BAR-DIMENSIONS R196)
     ENTERED: NIL
     FIRED: SE
     -INVES
     STATUS -> S
     EXPLOIT D41
     <--- D28 (IS-A-BAR-M1 R196)
          ENTERED: (PUI SI SE A)
          FIRED: SE
          -INVES
          STATUS -> S
          EXPLOIT D28
          <--- * D43 (IS-A-BAR R196)
               +INVES
               ENTERED: (SE SI PUI A)
               FIRED: SE
               -INVES
               STATUS -> S
               EXPLOIT D43
               <--- * D45 (IS-A-HP1-M1 R196)
                    +INVES
                    ENTERED: (PUI SI SE A)
                    FIRED: SE
                    -INVES
                    STATUS -> S
                    EXPLOIT D45
                    <--- * D46 (IS-A-HP1 R196)
                         +INVES
                         ENTERED: (SE SI PUI A)
                         FIRED: SE
                         -INVES
                         STATUS -> S
                         EXPLOIT D46
                         <--- * D47 (HAS-A-HP1 DR1)
                              +INVES
```

```
                                        ENTERED: (SE SI)
                                        FIRED: SE
                                        -INVES
                                        STATUS -> S
                                        EXPLOIT D47
                                        <--- * D49 (IS-A-H-M1 DR1)
                                             +INVES
                                             ENTERED: (PUI SI SE A)
                                             FIRED: SE
                                             DIFF -> 417.51
                                             PRIORITY -> 119.75
                                        <--- * D48 (HAS-A-HP2-M1 DR1)
                                             +INVES
                                             ENTERED: (PUI SI SE A)
                                             FIRED: SE
                                             DIFF -> 267.51
                                             PRIORITY -> 1.86E-3
                          <--- * D44 (IS-A-HP2-W R196)
                               +INVES
                               ENTERED: (PUI SI SE A)
                               FIRED: SE
                               -INVES
                               STATUS -> S
                               EXPLOIT D44
                               <--- * D50 (IS-A-HP2-M1 R196)
                                    +INVES
                                    ENTERED: (PUI SI SE A)
                                    FIRED: SE
                                    DIFF -> 6.92
                                    PRIORITY -> 72.15
NEW TOP PRIORITY INVESTIGATION: D49 (IS-A-H-M1 DR1)
```

previous region.

The basic model is "generate and test". Candidates are put forward and then tested to see if they have the desired properties. However, in SEER some of the properties may be used to help generate the candidate. Other PK results can combine with the GK model of what is expected.

Even in searching "mechanically" through the affinity tree we know to look for a part within a whole, and the properties can tell us when to stop generating candidates. Knowing what kind of region we are looking for can help find new regions. Knowing the region we see must be dark or symmetric, or a particular shape, for example, SEER may allow (advise) special purpose generators, or simplify and guide general processes.

Active knowledge permits previous results to combine with the properties of the conjectured region in guiding our search. Note when a part suggests a whole or vice versa, the properties of the recognized region may constitute some already established properties of the sought after region. Most importantly, previous results can help direct us where to look for a new region. Properties of the required region may specify its position and size in relation to established features of the scene.

We saw a good example of some of these features in the chapter one example. SEER provided advice on where to look and what to look for in directing Lozano's region profile tracker to search for a hammer handle profile perpendicular to the head we had found.

In the extreme all the required properties may have been used in generating the candidate; generation and testing merge. The acquisition process constitutes a verification of identity. In fact we may only verify various properties and never really generate a region candidate at all, in the usual sense. No boundary description need be found, for example. The description of a region is the trace of the process used to verify the required properties. There is no region R with properties P, but merely a locus of properties. The boundary of a cylinder side coincides in part with the outer boundary of the whole cylinder. The description or recognition of the side may employ a method that "refers to" or "assumes" (links to) a previous recognition of the outer boundary. It can employ the "affinity along a line" technique to take region profiles along a few lines through the cylinder, to verify a few points of division between the side and the end of the cylinder, and the surface shape of the side [Freuder 1973b]. When we get to the end of the cylinder, all we now need to establish is a flat

region running the width of the cylinder, and somewhat narrower in the other dimension. A few region profiles suffice to perceive and recognize the end.

It is in these cases that we see most most clearly the process of description and recognition merging, as I had hoped they would when I began this research. The distinction between what and where is further blurred here; there is no region R with properties P, but merely a locus of properties. The description and recognition were performed simultaneously and are both found in the trace of the process.

10

RELATED WORK AND FURTHER POSSIBILITIES

RELATED WORK

This section reviews related work and makes some of the obligatory contrasts. I discuss two streams of work: region oriented vision; heterarchy and other control models.

Let me observe first that most AI ideas can be found in some form in one of the early "classic" efforts. Environment driven concepts, for example, can be found expressed in Ernst [Ernst 1961] and in Simon's colorful ant analogy [Simon 1969]. Newell, Simon and Shaw, Gelernter, Slagle all move about in control networks (see [Feigenbaum and Feldman 1963]). Raphael, Quillian (see [Minsky 1968]) and Winston [Winston 1970] have exemplary descriptive networks.

There have also been two general inspirations for my attempt to build a complete coordinated "system". One is the series of efforts at M.I.T. to build a vision laboratory [Winston 1972a, 1973]. The other is Terry Winograd's success with his language system [Winograd 1971].

REGION BASED VISION

The focus of attention in AI vision research was for some time on line finding. However, region based analysis began some time ago, and after a period of inactivity, seems to be becoming popular again.

At M.I.T., Seymour Papert headed an early vision project that was region based [Papert 1966]. Binford worked on a region finder (difficult to reference adequately, but see [Minsky and Papert 1967]), and Guzman developed higher level region based programs [Guzman 1967].

Horn [Horn 1970] and Krakauer [Krakauer 1971] worked with regions. Though Krakauer's data structure is also a tree, it is not at all the same as the affinity tree. Krakauer's tree is based on an intensity "topographical map" of the scene. A region is formed from points below a threshhold intensity, not from points similar either within a threshhold, or in a relative sense. Krakauer is not concerned with finding objects, but with recognizing a single object.

I took up region analysis when my interests shifted from parallelipipeds to more realistic

objects. Jerry Lerman and I rediscovered a basic region finding algorithm. I developed a couple of other region growing ideas, but the key development was the affinity process. I felt this could serve as a basis for my study of realistic domains.

Azriel Rosenfeld has given me region growing references back to Muerle [Muerle 1970]. Rosenfeld himself has done some region based work [Strong and Rosenfeld 1973]. Of course textural analysis research deals with regions.

Brice and Fennema based an AI vision system on region primitives [Brice and Fennema 1970]. The basis for region formation is absolute threshholding (with sophisticated heuristics). The Edinburgh group adapted this work and further developed higher level region based analysis. The Edinburgh cup recognizer [Barrow and Popplestone 1971], though not their latest effort, is perhaps closest to the SEER effort in terms of being a complete region based recognition system. It also deals with non blocks world objects. However, they are rather idealized; SEER is concerned with confronting reality. The system is a bit early to have benefited from "heterarchical" concepts of organization. Its lack of "active knowledge" has been criticized earlier. Some other work at Edinburgh [Barrow, Ambler and Burstall 1972] [Milner 1970] shows promise in these directions, though not benefitting from contact with real data in a complete system.

Yakimovsky and Feldman [Yakimovsky and Feldman 1973] have developed an interesting region based system that uses Bayesian analysis to direct decisions. There is a "relative" step in their processing as well, but of a different sort. At each iteration toward a scene analysis they choose to erase a boundary line. The line is erased which brings the analysis closest to an acceptable model. This relative step involves the scene as a whole and semantic model criteria. The affinity process, of course, is local and syntactic. Many merges are made at each stage on the basis of relative region similarity.

In the affinity process regions are compared to each other; in the Yakimovsky and Feldman process the scene is compared to a model. Their process attempts to satisfy the entire scene simultaneously with a uniform algorithm. On the one hand the early semantic pruning of the partition choices threatens to get stuck with a faulty analysis. On the other hand the uniform, global nature of the algorithm requires a great deal of computation. There is still a good deal of the "apply each

predicate to each region" procedure found in less sophisticated model matching schemes.

The SEER affinity process is a "non purposive" pass that can efficiently provide us with some potential simplification without committing us to any final decisions. Then the active knowledge approach combines semantic information with local and incremental results to focus attention efficiently.

CONTROL

The major influences on SEER's control structure are probably Minsky and Papert's "heterarchy" [Minsky and Papert 1967, 1970a, 1970b, 1972] and Hewitt's PLANNER [Hewitt 1972]. [Winston 1973] describes some of the stages in the development of the heterarchy concept, by Winston, Shirai, et. al. SEER was conceived as going further in the direction of an integrated whole, rather than communicating parts. SEER emphasizes the domain directed possibilities. Hewitt emphasized the important distinction between antecedent and consequent deductive processing. Antecedent possibilities were developed in Charniak's thesis as "demons" [Charniak 1972].

Antecedent processing originally emphasized the use of results to cause immediate actions, add assertions, initiate computations. Demons were less immediate. They involved adding something to wait for something else. This is fine for a dynamic data base. However, a sequence of demons is an awkward way to handle a simple case where multiple arguments are required to establish a result. SEER investigates the problem of procedure invocation based on more than single assertions.

SEER's suggestions are on the one hand less concerned with immediate action than many antecedent concepts. They form proposals that go into a pool, to await further support or action. Other shades of potential antecedent interaction are explored, like priority effects. On the other hand suggestions are less passive, than say demons. They do not have to wait for results to come to them. They can direct efforts to establish them.

SEER combines antecedent and consequent functions into a unified node structure. Antecedent processing alone is bottom up, consequent processing top down. SEER, as we have noted, tends to "zig zag" between the two modes in a more "middle out" manner. SEER invests descriptive elements with a dual role as loci of control. Description and control information is imbedded in the same structure. PLANNER emphasized deductive knowledge as procedures. SEER permits network descriptions perhaps more natural to its domain than PLANNER-like theorems, to be used in a manner

which allows them to assist in their own acquisition in specific instantiations. It permits the programmer to write description and program control together in a natural way that permits effective processing.

The basic antecedent idea takes a -> b and uses it to say: given a, execute (or assert) b. Charniak's demon notion takes a + b -> c, notes that this implies a -> (b -> c) and uses that to say that we can iterate the antecedent process: given a, we assert b ->c, then wait for b in the usual antecedent fashion. Thus we add additional antecedents and defer action to a degree. SEER takes a + b -> c, infers that a -> maybe c and b -> maybe c, and says: given a or given b conjecture c. Set up c as a consequent goal; however, execution of that goal can be deferred subject to global comparison with other goals and possible further encouragement.

The subsequent developments of CONNIVER [Sussman and McDermott 1972; McDermott and Sussman 1974] may be related to SEER's ability to skip around and return to various investigative paths. However, SEER does this "out in the open" using intermediate steps as data, rather than in system stored environments.

Several other models came to my attention a bit late to influence the original development of SEER, but have helped me analyze and refine it. Seymour Papert's "little man" concept [Papert 1972], seen in Hewitt's actors [Hewitt and Smith 1975] and Kay's Smalltalk system, provides a good way of looking at the GK and PK nodes. Marvin Minsky's frame systems [Minsky 1974] obviously provide a good model for the "investigations", and potentially for extended use of context, and larger collections of GK.

DIRECTIONS FOR FURTHER RESEARCH

Vision could benefit from more system oriented research. I have in mind projects along the line of Moses' Mathlab, Martin's automatic programming project, or a language effort like Shank's. An ongoing system will permit researchers to deal with real data, find motivation in real vision problems. The central vision problem is extracting perception from sensory input. Operations on symbolic models of the visual domain do not address this problem. Research in individual aspects of vision needs to be attacked with the interactive spirit that a system like SEER encourages. SEER provides a formalism for

interaction. Not to be overlooked is the benefit to researchers of an interactive research environment where problems and successes can be shared. Even novices can work on a motivated problem and see their results operate in a real environment.

The effort needed to address the vision problem from sensory input, and the dissimilar interests and expertise required at different levels, discourages integrated efforts, and may discourage entry into the field. An effort like SEER, as a single thesis project, of necessity sacrifices depth for breadth. The SEER project emphasized interaction over expertise. Once a framework like SEER has been established, however, it can be used by many students who wish to integrate their efforts in individual aspects of vision, or extend the services the SEER system can provide.

Research on problems like texture can be done within the SEER system. Lower level components, like the affinity process can be extended as required, based on the motivation of experience with real scenes. An effort should be made to build up a "vocabulary" of basic visual predicates, and of higher level concepts. We would like to determine the size and structure of such a vocabulary for a given set of phenomena. We want to develop a vocabulary that will exhibit the "asymptotic saturation" phenomena, eventually requiring fewer and fewer additions to encompass additions to the recognition domain of the system.

I would like to develop the concept of really "visual" predicates, like "in there", that are not easily encompassed in the common verbal analyses of phenomena.

SEER needs more case studies. Its area of competence should be expanded. More attention should be given to the problems of handling large amounts of data, perhaps through frame constructs or layered memories of some sort. The basic interactive process and individual problem areas, like lighting can be explored through case studies.

The system "user" performing these case studies will be programming in GK. Another task would be to improve GK as a programming language and efficient processer.

Research in the areas of computer program writing and learning could be carried on in the SEER context. We would like SEER to be able to program GK itself. In a sense SEER already chooses a PK program for the scene from the GK. GK is written so that SEER can understand and use it. We should have a good domain for "automatic programming".

Learning techniques can also be applied to the dynamic improvement of the priority system with experience, and its sensitivity to context. Other SEER "system" elements could be extended as required by results. Theoretical understanding of relevance analysis and interactive possibilities might be extended, motivated by experience, and applied for evaluation. Particular elements of the system, monitor, e.g. exploitation options, could become more sophisticated to meet new problems.

Throughout this process the system concept allows for motivation, experimentation, and evaluation in the real visual domain. Theory and practice can provide mutual feedback. Ultimately we would hope to build and implement extended models of the visual process. Experimentation can motivate our models. We can evaluate whether the implementations meet our models and satisfy our needs. The models can be related to visual system studies in psychology and neurophysiology. Formal models can be made of the control structure and processing flow in general computation theory terms as well. What we learn about interactive control in large systems can be applied to other areas of AI research on sensory systems, problem solving, knowledge structures and intelligent processing.

Ideally the vision project could be part of a larger effort, e.g. to develop a "craftsman". (The potential for computer "generalists" is an advantage that AI has over convential computer science and engineering efforts.) A "handy man" project would explore and exploit this potential. Beginning with a system that recognized a few tools, we could develop a program that could use both visual and manipulative skills in an extensible fashion. This project could provide a domain for the new efforts being made in automatic programming and procedural learning. Our handyman would acquire a wide variety of knowledge which could be applied to develop new skills and adapt to new environments.

BIBLIOGRAPHY

AIMIT = Artificial Intelligence Laboratory, M.I.T., Cambridge, Massachusetts

[Barrow and Popplestone 1971]
H.G. Barrow and R.J. Popplestone. Relational Descriptions in Picture Processing. in Machine Intelligence 6. B. Meltzer and D. Michie, Eds.. Halsted Press. New York. 1971.

[Barrow, Ambler and Burstall 1972]
H.G. Barrow, A.P. Ambler and R.M. Burstall. Some Techniques for Recognising Structures in Pictures. in Frontiers of Pattern Recognition. Academic Press. New York. 1972.

[Brice and Fennema 1970]
C.R. Brice and C.L. Fennema. Scene Analysis Using Regions. Artificial Intelligence. vol 1, no. 3. Fall 1970.

[Charniak 1972]
E. Charniak. Toward a Model of Children's Story Comprehension. AI TR-266. AIMIT. December 1972.

[Ernst 1961]
H.A. Ernst. MH-1, A Computer-Operated Mechanical Hand. D.Sc. Dissertation. M.I.T. December 1961.

[Feigenbaum and Feldman 1963]
E.A. Feigenbaum and J. Feldman, Eds. Computers and Thought. McGraw-Hill. New York. 1963.

[Freuder 1971]
E.C. Freuder. Views on Vision. Working Paper 5. AIMIT. February 1971.

[Freuder 1972]
E.C. Freuder. Recognition of Real Objects. Working Paper 33. AIMIT. October 1972.

[Freuder 1973a]
E.C. Freuder. Suggestion and Advice. Working Paper 43. AIMIT. March 1973.

[Freuder 1973b]
E.C. Freuder. Active Knowledge. Working Paper 53. AIMIT. October 1973.

[Goldstein 1974]
I.P. Goldstein. Understanding Simple Picture Programs. AI TR-294. AIMIT. April 1974.

[Guzman 1967]
A. Guzman. Some Aspects of Pattern Recognition by Computer. AI TR-224. AIMIT. February 1967.

[Guzman 1968]
A. Guzman. Computer Recognition of Three-dimensional Objects in a Visual Scene. MAC TR-59. Project MAC, M.I.T. 1968.

[Hewitt 1972]
C. Hewitt. Description and Theoretical Analysis (Using Schemata) of Planner: A Language for Proving Theorems and Manipulating Models in a Robot. AI TR-258. AIMIT. April 1972.

[Hewitt, Bishop and Steiger 1973]
C. Hewitt, P. Bishop and R. Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. Third International Joint Conference on Artificial Intelligence. August 1973.

[Hewitt and Smith 1975]
C. Hewitt and B. Smith. Towards a Programming Apprentice. IEEE Transactions on Software Engineering. volume SE-1, number 1. March 1975.

[Hollerbach 1975]
J. Hollerbach. Hierarchical Shape Description of Objects by Selection and Modification of Prototypes. Master's Dissertation. M.I.T. January 1975.

[Horn 1970]
B.K.P. Horn. Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View. AI TR-232. AIMIT. November 1970.

[Horn 1974]
B.K.P. Horn. Determining Lightness from an Image. Computer Graphics and Image Processing. volume 3, number 4. December 1974.

[Huffman 1971]
D. Huffman. Impossible Objects as Nonsense Sentences. in Machine Intelligence 6. B. Meltzer and D. Michie, Eds. American Elsevier. New York. 1971.

[Kelly 1971]
M.D. Kelly. Edge Detection in Pictures by Computer Using Planning. in Machine Intelligence 6. B. Meltzer and D. Michie, Eds. American Elsevier. New York. 1971.

[Krakauer 1971]
L.J. Krakauer. Computer Analysis of Visual Properties of Curved Objects. MAC TR-82. Project MAC, M.I.T. 1971.

[Lozano-Perez 1975]
T. Lozano-Perez. Parsing Intensity Profiles. AI Memo 329. AIMIT. 1975.

[Marr 1974]
D. Marr. The Computation of Lightness by the Primate Retina. Vision Research. volume 14. 1974.

[Marr 1975]
D. Marr. Early Processing of Visual Information. AI Memo 340. AIMIT. 1975.

[McDermott and Sussman 1974]
D. McDermott and G.J. Sussman. The Conniver Reference Manual. AI Memo 259A. AIMIT. 1974.

[Milner 1970]
D. Milner. A Hierarchical Picture Interpretation System Utilising Contextual Information. Department of Machine Intelligence and Perception, University of Edinburgh. 1970.

[Minsky 1968]
M. Minsky, Ed. Semantic Information Processing. The M.I.T. Press. Cambridge, Massachusetts. 1968.

[Minsky 1974]
M. Minsky. A Framework for Representing Knowledge. AI Memo 306. AIMIT. June 1974.

[Minsky and Papert 1967]
M. Minsky and S. Papert. Research on Intelligent Automata. in Project MAC Progress Report IV. M.I.T. Press. Cambridge, Massachusetts. 1967.

[Minsky and Papert 1970a]
M. Minsky and S. Papert. 1968-1969 Progress Report. AI Memo 200. AIMIT. 1970.

[Minsky and Papert 1970b]
M. Minsky and S. Papert. Proposal to ARPA for Research on Artificial Intelligence at MIT 1970-1971. AI Memo 185. AIMIT. December 1970.

[Minsky and Papert 1972]
M. Minsky and S. Papert. Progress Report. AI Memo 252. AIMIT. January 1972.

[Muerle 1970]
J.L. Muerle. Some Thoughts on Texture Discrimination by Computer. in Picture Processing and Psychopictorics. B. Lipkin and A. Rosenfeld, Eds. Academic Press. New York. 1970.

[Papert 1966]
S. Papert. The Summer Vision Project. AI Memo 100. AIMIT. 1966.

[Papert 1972]
S. Papert. Teaching Children to be Mathematicians versus Teaching about Mathematics. Int. J. Math. Educ. Sci. Technol. vol. 3. 249-262. 1972.

[Popplestone, Brown, Ambler, Crawford 1975]
R.J. Popplestone, C.M. Brown, A.P. Ambler, G.F. Crawford. Forming Models of Plane-and-Cylinder Faceted Bodies from Light Stripes. Advance Papers of the Fourth Int. Joint Conf. on Artificial Intelligence. 1975.

[Rosenfeld 1969a]
A. Rosenfeld. Figure Extraction. in Automatic Interpretation and Classification of Images. A. Grasselli, Ed. Academic Press. New York. 1969.

[Rosenfeld 1969b]
A. Rosenfeld. Picture Processing by Computer. Academic Press. New York. 1969.

[Roberts 1963]
L. Roberts. Machine Perception of Three-Dimensional Solids. Technical Report 315. Lincoln Laboratory, M.I.T. May 1963.

[Shirai 1972]
Y. Shirai. A Heterarchical Program for Recognition of Polyhedra. AI Memo 263. AIMIT. June 1972.

[Simon 1969]
H.A. Simon. The Sciences of the Artificial. M.I.T. Press. M.I.T., Cambridge, Mass. 1969.

[Strong and Rosenfeld 1973]
J.P. Strong III and A. Rosenfeld. A Region Coloring Technique for Scene Analysis. Communications of the ACM. volume 16, number 4. April 1973.

[Sussman 1973]
G.J. Sussman. A Computational Model of Skill Acquisition. AI TR-297. AIMIT. August 1973.

[Sussman and McDermott 1972]
G.J. Sussman and D. McDermott. Why Conniving is Better than Planning. AI Memo 255A. AIMIT. April 1972.

[Sussman, Winograd and Charniak 1971]
G.J. Sussman, T. Winograd and E. Charniak. Micro-Planner Reference Manual. AI Memo 203A. AIMIT. December 1971.

[Waltz 1972]

D. Waltz. Generating Semantic Descriptions from Drawings of Scenes with Shadows. AI TR-271. AIMIT. November 1972.

[Winograd 1971]
T. Winograd. Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. AI TR-235. AIMIT. February 1971.

[Winston 1970]
P.H. Winston. Learning Structural Descriptions from Examples. AI TR-231. AIMIT. September 1970.

[Winston 1972a]
P.H. Winston. The M.I.T. Robot. in Machine Intelligence 7. B. Meltzer and D. Michie, Eds. John Wiley and Sons. New York. 1972.

[Winston 1972b]
P.H. Winston. Wizard. Working Paper 24. AIMIT. 1972.

[Winston 1973]
P.H. Winston, Ed. Progress in Vision and Robotics. AI TR-281. AIMIT. May 1973.

[Yakimovsky and Feldman 1973]
Y. Yakimovsky and J.A. Feldman. A Semantics-Based Decision Theory Region Analyzer. Third International Joint Conference on Artificial Intelligence. August 1973.

[Zucker, Hummel and Rosenfeld 1975]
S.W. Zucker, R. Hummel and A. Rosenfeld. Applications of Relaxation Labelling, 1: Line and Curve Enhancement. TR-419. Computer Science Center, University of Maryland. 1975