A PROCEDURAL MODEL OF RECOGNITION

William S. Havens
Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada

1. Introduction

This work is based on the belief that a computational theory of perception must encompass dual theories of representation and recognition. We propose a procedural model of recognition which 1) provides a mechanism for flexibly integrating top-down and bottom-up control structures, 2) supports the use of a hierarchy of model driven methods, and 3) defines a deductive process scheduling mechanism.

2. The Chicken and Egg Problem

The top-down recognition model proposed by Minsky(1974), championed by Kuipers(1975), and used by many others exhibits a number of short-comings:
1. A frame must explicitly be made the current hypothesis before its expertise is available to the recognition process.
2. An ordering must be placed on alternative subgoals. The top-down paradigm forces the choice of a single, hypothesis at a time. The mechanism for choosing a next hypothesis is completely failure driven.
3. Identical subgoals must be carried out independently. The same subgoal may be attempted and achieved numerous times by the system.

To avoid this thrashing behaviour, Kuipers(1975) has advocated the use of a similarity network to recommend a replacement candidate frame on failure. This scheme assumes first that a mapping exists between the failing frame and each next candidate frame and secondly that the network is in some sense "complete". Any "surprises" cause the system to revert to blind automatic backtracking.

These difficulties are three manifestations of what Mackworth(1975) has called "the chicken and egg problem". It seems clear that to avoid these problems, an integration of both top-down and bottom-up techniques are needed. Kaplan(1973) and more recently Bobrow & Winograd(1977) have advocated the use of multiprocessing with priority queue scheduling. We argue that in order to effectively integrate top-down and bottom-up recognition methods more sophisticated mechanisms than priority queues are needed.

3. Implementation

We define a procedural recognition model which utilizes multiprocessing and a deductive scheduling mechansim to realize an integration of top-down and bottom-up search. The model operates in two mutually recursive modes. The first mode is called matching and incorporates essentially the top-down recognition paradigm. The expectations of a frame are matched by direct observation by recursive use of matching I.e. subgoaling, and by recursive calls to the second mode described below.

The second mode implements bottom-up search and consists of two cyclical phases called expectation and completion.

Each frame may have associated with a number of processes called supergoals. Instead of relying entirely on matching to recognize an instance, a frame may suspend its recognition by computing a set of current expectations and suspending supergoal processes to patterns representing those expectations. Each such process remains suspended until evidence is discovered which satisfies its expectations. The process is then resumed and the recognition of its associated frame continues.

Whereas a subgoal has an explicit caller and must eventually return a success or failure to that caller, a supergoal has no explicit caller. Instead, when a supergoal succeeds it begins the completion phase by resuming in turn one or more suspended supergoals of those higher frames of which its frame may be part. This provides a mechanism for handling non-determinism without resorting to failure driven methods. A particular frame instance need be recognized only once.

Completion also provides a deductive process scheduling mechanism. Processes are activated when the expectations that they are bound to have been satisfied. The model, as well, supports the use of a hierarchy of model driven methods to guide the recognition process. A frame's method computes a set of expectations, binds supergoals to those expectations, and may relinquish control to a higher method. Alternatively, the method may retain control. This permits a frame to pre-empt a higher but less specific method and Impose its own more powerful method to encourage the discovery of evidence satisfying its expectations and propagating its own recognition.

This recognition model has been implemented as an experimental programming language, a dialect of LISP called MAYA (Havens,1976). The language provides primitives for constructing and accessing frames and semantic networks. MAYA provides control structure mechanisms for implementing the recognition model described. Multiple processes associated with particular frames may be created, suspended to patterns within frames, and resumed again by matching those patterns. Specific control primitives are defined to permit subgoal and supergoal interaction and to support the realization of method hierarchies. We are presently experimenting with the recognition model by applying MAYA to a computer vision task (Havens,1977).

Bibliography

Bobrow, D.G. & Winograd, T.(1977) An Overview of KRL: A Knowledge Representation Language, Cognitive Science, vol 1, //I, Jan'77.

Havens, W.S.(1976) MAYA Language Reference Manual, TM-13, Dept.of Comp.Sci., UBC,Vancouver, Canada.

Havens, W.S.(1977) A Computational Model of Recognition for Perception, Ph.D. thesis, Dept. of Comp. Science, Univ.of British Columbia, Vancouver Canada, in preparation.

Kuipers, B.J.(1975) A Frame for Frames: Representing Knowledge for Recognition, in Representation and Understanding, D.G.Bobrow & A.Collins (eds.) Academic Press, New York.

Minsky, M.(1974) A Framework for Representing Knowledge, AI Memo #306, MIT AI Lab, Cambridge, Mass.

Mackworth, A.K.(1975) How to See a Simple World, in Machine Intelligence 8, E.W.Elcock & D.Michie (eds.) in press.

Kaplan, R.M.(1973) A Multiprocessing Approach to Natural Language, Proc. 1973 National Computer Conference, AFIPS Press.

Author's new address: Department of Computer Science University of Tennessee, Knoxville, TENN. 37916

REPRESENTATION OF ACTIONS THAT HAVE SIDE-EFFECTS

N.S. Sridharan and F. Hawrusik
Rutgers University
New Brunswick, NJ 08903

Systems that reason about actions, whether they do plan generation [1] or plan recognition [2] typically model the effects of actions that occur in a plan. Simple declarative schemata allow the specification of assertions to be added/deleted to model the primary effects of actions. Side-effects of actions are those that are conditional on properties of the state in which the action is taken. When representing actions which have side-effects, conventional wisdom suggests adopting a procedural representation for they allow detailed specification of side-effects. However the procedural representation hides this knowledge from other parts of the system, thereby hindering the system in reasoning about side-effects. We use a STRIPS like declarative schema for actions that has parameters, preconditions, assertional forms for goal and outcomes and investigate three methods of representing knowledge about side-effects and discuss how the system computes side-effects without running into severe combinatorics. The following discussion and examples deal with knowledge representation as implemented in the AIMDS system which forms the AI framework for the BELIEVER project.

Examples

Consider a normal input of the form "John walked from the office to the bus station" interpreted in a world model where "John is at the office" is true. The conclusions drawn include "John is at the bus station". This can be handled using an act schema with three variables P, FL and TL as shown below.

```
(Each WALK act has
    (agent        [a PERSON [refer: P]])
    (fromloc      [a LOCATION (refer: FL]])
    (toloc        [a LOCATION [refer: TL]])
    (goal         (PROPOSITION [P loc TL]))
    (precond      (PROPOSITION [P loc FL))))
```

We have extended the interpreter to deal with some simple cases of partial act instance descriptions and incomplete world models. The incomplete description "John walked to the bus station" can be filled in using the world model, so that the system now can conclude "John walked from the office". Similarly, if in the world model "John's location is unknown" then from the normal input "John walked from the office to the bus station" the system concludes "John was at the office before walking to the bus station". Dealing with side-effects requires additional knowledge. For example, consider the world model "John is at the office; John is holding a package; The package is at the office". To update the world model properly, the location of the package must be changed to the same location as John.

We explain three methods in which this knowledge about side-effects is represented.

(a) <u>Conditional Outcomes</u>: We associate with the WALK schema a set of conditional outcomes which include

[P inhand 0] => [0 loc TL]

with the interpretation "when the person P walks to TL and has in hand an object 0, the object O shifts location to TL". The interpreter that updates the world model then accounts for these conditional outcomes by testing each left side and effecting the changes prescribed in the right side. If the user attempts to be as complete as possible in writing down conditional sentences, he risks the possibility that most of these would be irrelevant to any particular instantiation of an act and the possibility that he might have missed some situations, Furthermore, the system is liable to waste effort in testing a combinatorially prohibitive number of conditional sentences. If a relevant conditional sentence is missing the system has no way of recognizing this to prompt the user.

(b) <u>Consistency conditions on the wor1d model</u>! The discipline of stating consistency conditions on the world model introduced in [3] provides a second representation. A causal dependency of the physical world may be captured by writing a consistency condition such as,

For Person P and Object 0,
[P inhand 0] »> [(loeof P) = (loeof O)]

which is associated with the relation "inhand". This is deceptively similar to the conditional sentence shown in (a) but an important difference is that here it is NOT knowledge associated with WALK. When "John has in hand the package" is initially put into the world model, the system checks this consistency condition and makes note of the depdendency of this assertion on the two supporting assertions "John is at the office" and "The package is at the office". When changing John's location as a result of the Walk act, the system recognizes that the support for "John has in hand the package" is being changed. At this point the system retrieves side-effect rules for updating "inhand" when "loc" is changed. This side effect rule specifies that the location of the package changes and that the person still has the object in hand.

It is worth noting that with a clean factoring out of causal dependency knowledge, (i) the system performs reduced search as a result of having precomputed the dependency between assertions in the world model; and (ii) if a side-effect rule is missing the system prompts the user for an appropriate rule, permitting knowledge acquisition in context; and (iii) the ability to follow causal dependencies is useful [SOPHIE] in answering "What if.." questions where a counterfactual denies a fact in the world model and is not a result of any specific action that the system knows about.

(c) Frames of Reference: Although the method TH) reduces the combinatorics of simulating a single action and makes the system more robust, it requires a painstaking updating of the world model each time a similar action occurs. For example, if John walks to several places in succession holding the package we want the system to save effort in updating the location of the package for each act. We introduce the notion of a frame of reference whereby at the time John is said to have the package in hand, the location of the package is changed from (Package loc Office) to (Package loc [locof John]) where [locof John] refers to John's location in the current world model. The actual location of the package will be computed when needed from its frame of reference - the current location of John. The WALK schema given before can be used in the usual manner to update only John's location and computation of the side-effect is carried out implicitly, and no effort is spent on this while updating the world model. The procedural attachments available in the AIMDS system provide a convenient mechanism for effecting such implicit side-effects. We use one procedure attached to "inhand" to shift the Object's location to the frame of reference of the person when "A inhand 0" is asserted; and another procedure attached to "inhand" gets activated when "A inhand 0" is removed and the activated procedure removes the object from its frame of reference and asserts an explicit location for it.

Thus while updating the world model for any one of a series of Walk acts there is no explicit computation of side-effects. Yet after these location transfers, the system determines the location of the object with ease. When John drops the package into a mailbox, the second procedure asserts that the package is now in the mailbox and John can walk away freely. This method is similar to the familiar technique of retaining only "primitive" assertions in the world model and computing all "derived" assertions whenever needed. However, the method presented is selective so that derived assertions not changed by an action are retained and need not be rederived.

Significance

We have presented three methods which complement each other, each suited well to a particular task. Method (a) makes explicit the side-effects, permitting reasoninq about side-effects; method (b) allows extending the action set and facilitates the acquisition of side-effect rules; and method (c) seems well suited for actually simulating the effects of an action. A topic of further research is into means for automatic re-representation of side-effect knowledge from one form into another, especially from method (b) into method (a) and method (c).

REFERENCES

Fikes, R.E. and Nilsson, N. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Journal of Artificial Intelligence, 3(1), 27-68, 1972.

Schmidt, C.F. and Sridharan, N.S. "Plan Recognition: A Hypothesize and Revise Paradiqm", Fifth "international Joint Conference on Artificial Intelligence, MIT, August 1977.

3. Srinivasan, C.V. "The Architecture of Coherent Information System: A General Problem Solving System", IEEE Transactions on Computers, April 1976.