BELIEF FUNCTIONS FOR REAL-TIME SCRIPT PROCESSING

Philip Schaefer

Martin Marietta - Advanced Automation Technology
P.O. Box 179, M.S. 0427
Denver, CO 80201

ABSTRACT

Real-time situation understanding is a difficult problem, in that decisions must be made on a continuous basis with continuously changing data. An attractive approach to the problem is to match the current data set to a memory of script-like structures. Partial matching to scripts enables powerful understanding and prediction, but requires robust uncertainty mechanisms to overcome inherent ambiguity and error.

Although powerful uncertainty techniques for dealing with time-relevant data have been developed, little with regard to the implications of the real-time problem has been presented.

This paper discusses some of the special requirements for a real-time script belief function, and presents a derivation of such a function. Finally, examples demonstrating the characteristics of the Script Belief function are provided.

INTRODUCTION

Real-time situation understanding poses some unique problems for Artificial Intelligence systems. When presented with a continuous stream of input data, there is never a "complete data set" which can be exhaustively examined. Rather, the best decisions which can be made at each point in time must be determined, making the best use of the data at hand. An example of this type of system might be a weather forecasting Expert System, if it were required to continuously provide meteorological forecasts. Another example could be a process-monitoring system, called upon to continuously provide information on what was happening within the process and what predictions could be made.

One useful framework for understanding such a stream of observations is a Script-like (Schank 1978) paradigm. For lack of a better term, rather than the fairly specific definitions of Scripts put forth by (Schank 1978), in this discussion, a script will be considered as any tirae-sequenced memory structure which describes a common sequence of events. For example, the script shown in Figure 1 describes the scenario of a person going to work,

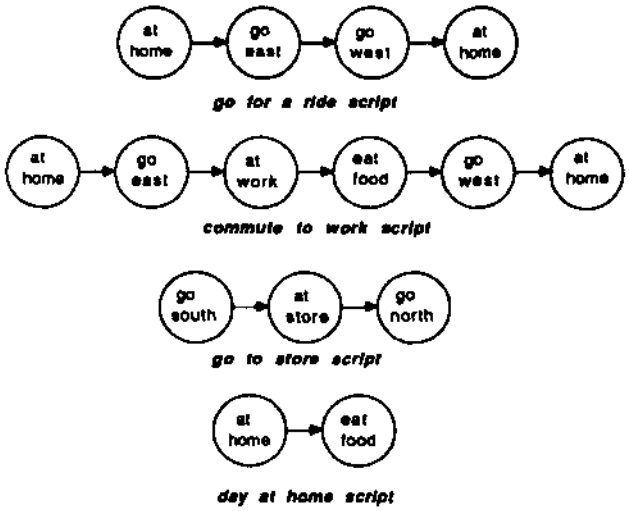

Figure 1. Example scripts.

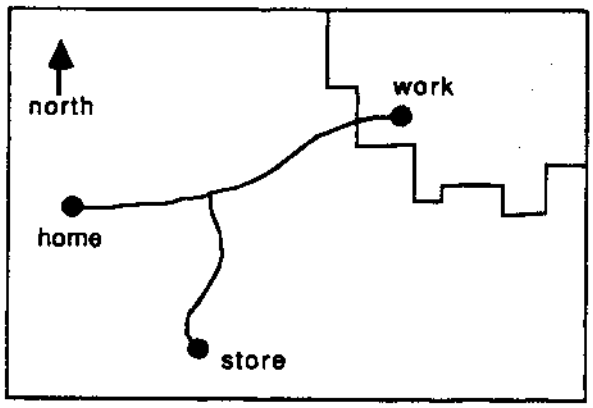

Figure 2. Area where example scripts take place

in terms of a sequence of simple observations, referred to as "script elements." The script paradigm is attractive for the real-time situation understanding application because sequences of events, which alone provide little useful information, can be matched to scripts in memory, to provide a synopsis of what seems to be taking place, and to provide predictive capability. Because the matches are likely to be partial, and because error and ambiguity are ever present, it is important to utilize a robust uncertainty management scheme, to ensure that the best among viable matches is believed.

A number of interesting issues are apparent when dealing with a continuous stream of input data, which are not important when analyzing a static data set, or data which is not time-ordered. One important issue is a sensitivity to where in a script matching occurs. For example, consider a script (such as Figure 1) in which matches occur only in elements temporally late in the sequence. Belief in this match would imply that the earlier element events did occur, but, due to uncertainty, were missed. Now, consider a script with matches only early in the element sequence. Belief in this match implies that the following elements probably will occur in the future. Because of the constraints imposed by the real-time data connection, the later events indeed can occur, so clearly, the belief in the latter type of match should be higher. Below, several possible approaches to belief management are reviewed with respect to special requirements of real-time script processing, such as the above.

One technique for choosing among script matches is to use an heuristic threshold value for the minimum acceptable number of matched elements (Lebowitz 1980), or to generate a weighting function for considering the numbers of observed and unobserved elements (Bozma 1985). Similarly, other systems rely on the Knowledge Engineering process to provide rules for how much to believe in a match, as a function of which elements are matched in the particular script (Azarewicz 1986). Such techniques have been successfully implemented in the referenced systems, but must be developed for each application domain. If the requisite information is available, these heuristic approaches may be effective. In many cases, however, a more general solution is preferable.

In the arena of Plan Recognition, issues similar to the script-matching problem have been addressed. An interesting approach to uncertainty in this area is Evidential Reasoning (Fall 1986, Lowrance 1986). One technique for dealing with the temporal aspects of the problem is to provide "frames of discernment" in which mappings between compatible events can be made (Lowrance 1986). One possible compatibility mapping is between adjacent elements of script-like structures. Another approach is to consider observations as evidence with temporal extent (Fall 1986), manipulating the constraints between those extents to determine belief. Unfortunately, because they were not explicitly concerned with the real-time issues of interest here, these approaches do not make use of the observation constraints available. For example, they are insensitive to temporal order of observations, overlooking the constraint mentioned above.

Other research in Plan Recognition uses a Logic-based approach (Kautz and Allen 1986), in which a taxonomy of possible event sequences is made, used by a deductive process which selects likely conclusions. Another approach uses the theory of Endorsements (Sullivan and Cohen 1985), wherein rules about what makes good or bad plan-matches are needed. Currently, these methods also deal primarily with the compatibility aspects of matching, rather than with real-time issues. While they appear extensible for incorporating the necessary constraints, once those constraints are properly formalized, it is currently unclear how to proceed in this direction.

The approach presented in this paper, therefore, is to derive Script Belief functions which incorporate the relevant aspects of the real-time understanding problem. A probabilistic approach was selected, because Conditional Probability theory allows the desired constraints to be explicitly addressed, while maintaining a degree of rigor. When the resulting equations have been studied and understood, it is likely that the essense of the probabilistic approach will also be applicable to many of the above techniques.

II. THE PROBABILISTIC SCRIPT BELIEF FUNCTION

Keeping the general characteristics described in the Introduction in mind, a robust belief function can be derived, as shown below. In this section, an overview of the approach is presented.A complete derivation of the belief equations appears in the Appendix.

The general approach used in the derivation is to consider the probability of each element in a script in terms of the confidence that it was observed, as well as in terms of the probability with which it appears in the Knowledge Base. The resulting belief function is

$$P(\text{script}) =$$

$$P(\text{script} \mid \text{all elements occur})$$

$$* \left[\prod_{k=1}^{n} (\text{confidence}(k\text{th element}) \right.$$

$$+ P(k\text{th} \mid k\text{-}1, \overline{k\text{th by sensors}}) *$$

$$\left. * (1 - \text{confidence}(k\text{th element})) \right]$$

where $P(kth \mid k\text{-}1, \overline{kth\ by\ sensors})$ is the a posteriori probability that, given the occurrence of the 1st through k-1st elements and no sensory evidence of the kth element, the kth element will occur.

$P(script \mid all\ elements\ occur)$ is the probability that if all the elements of the script occur, the event described by the script is occurring. This is not, in general, 1.0, because we assume that if another script which subsumes the given script is occurring, the given script is not really occurring.

confidence(kth) is the confidence of an external observer that the kth element has occurred, based on sensor indications.

The probabilities $P(script \mid all\ elements\ occur)$ and $P(kth \mid k\text{-}1, \overline{kth\ by\ sensors})$ can be obtained by one of two methods. First, the Knowledge Base can be inspected to determine how likely it is that the elements in the script can occur in another script, i.e.,

$$P(script \mid all\ elements) = \frac{1}{\text{number of scripts with elements 1 through n in proper order}}$$

By this method, $P(kth \mid k\text{-}1, \overline{kth\ by\ sensors})$ is similarly computed as

$$P(kth \mid k\text{-}1, \overline{kth\ by\ sensors}) = \frac{\text{number of scripts with elements 1 through k in correct order}}{\text{number of scripts with elements 1 through k-1 in correct order}}$$

With a "Static Knowledge Base" assumption, these probabilities, though computationally expensive, can be computed off-line and stored with the respective scripts. On the other hand, if the Knowledge Base is not static, such as the case with a script-learning system, a technique for computing the above two probabilities requiring less coupling between individual scripts is needed. An approximation, compatible with a "Dynamic Knowledge Base" assumption, requiring global parameters of the Knowledge Base, rather than details of all the scripts, has also been developed. This approximation, rather than explicitly examining the Knowledge Base, computes the probability that other scripts contain the relevant script elements. The resulting equations for the probabilities are

$$P(script \mid all\ elements) = \frac{1}{1 + \sum_{L=1}^{max\text{-}L} \left[1 - \left(1 - \prod_{i=1}^{n} p(ith)\right)^{\binom{L}{n}}\right] scripts(L)}$$

where n is the length of the script being scored, scripts(L) is the number of scripts of length L, max-L is the maximum script length, and p(ith) is the a priori probability of finding the ith element in the Knowledge Base.

$$P(kth \mid k\text{-}1, \overline{kth\ by\ sensors}) = \frac{1 + \sum_{L=1}^{max\text{-}L} \left[1 - \left(1 - \prod_{i=1}^{k} p(ith)\right)^{\binom{L}{k}}\right] scripts(L)}{1 + \sum_{L=1}^{max\text{-}L} \left[1 - \left(1 - \prod_{i=1}^{k-1} p(ith)\right)^{\binom{L}{k-1}}\right] scripts(L)}$$

These more approximate probabilites may also be computed and stored with the scripts, and updated only as often as the parameters scripts(L) and p(kth) change significantly. Upon update, additionally, the re-evaluation of the equations is computationally much simpler than that of the first, more exact, method.

The probabilistic belief equations make the assumption that there is no a priori reason to expect the occurrence of certain scripts more than others. While this may, in general, not be true for a given Knowledge Base, as the number of scripts becomes reasonably large, the equations tend to become more insensitive to the assumption inaccuracy.

III. AN EXAMPLE

A simple example of processing in a script-based system will help to show how the Probabilistic Belief Function works and demonstrate some of its unique properties.

Assume that the Knowledge Base contans scripts about a person who lives in the town shown in Figure 2. The scripts describing the person's common activities are as shown in Figure 1. The beliefs assigned by the Probabilistic Belief function, as the input observations of Table 1 are observed, are indicated in Figures 3 and 4. For purposes of the discussion, each observation will be assumed to have a confidence of 0.8.

Initially, the person is seen to be at home. This activates all scripts except the "go to store" with a small value. The "go for ride" and "commute to work" scripts are each activated in two places, but the earliest matches yield the higher, shown, score. This characteristic is preferable to a simple "compatibility check," such as afforded by a standard "evidential reasoning" system (Lowrance 1986), which would give much more uniform scores to

table 1. EVENTS DETECTED IN THE WORLD

TIME	EVENT DETECTED
1	at home
2	going east
3	at work
4	eating food
5	going west
6	going south
7	at store
8	going north

the matches. It is also an attractive alternative to requiring "script entry conditions" (Bozma 1985) to be met, because, though unlikely, a late match may turn out to be the correct match, if noise or uncertainty is high.

At this point, the "day at home" script has 50% of its elements matched, but the "dynamic assumption" assigns it only a 0.2 score. This is because the probability of subsuming scripts, such as "commute to work," is quite high. The "static assumption" gives "day at home" a slightly higher rating because it knows that there is only one such subsuming script in the particular Knowledge Base.

Later, the person is seen to be going east and arriving at work. At this point, the "commute to work" script is highly preferred, as it is a detailed script with significant matches. If the person would have gone west before arriving at work, however, the commute script would not have been preferred.

At time 4, when the person is seen to be eating, the "day at home" script has all elements matched, with a probability of 0.9 (with either assumption) that all of its elements occur. However, the belief is still much smaller than for the "commute" script, because of the high probability of being subsumed.

After work, the person is seen to travel south toward the store. The static function immediately assigns a very high score to the "go to store" script, as no other contains similar elements. The probabilistic increase from the dynamic function is less dramatic, but in both cases, the "store" script is the highest believed as more time passes.

The above example demonstrates a case of several significant divergences between the "Static Knowledge Base" and the "Dynamic Knowledge Base" equations. This results because the scripts in the example Knowledge Base are very similar to each other, violating the "equal a priori probability" assumption in the probabilistic "Dynamic" Belief derivation. Because of this similarity, the more exact Belief Function imparts less significance toa match if that match is likely to occur in numerous other scripts. Although it did not adversely affect the predictions in the example, this is a relevant point to consider when contemplating this

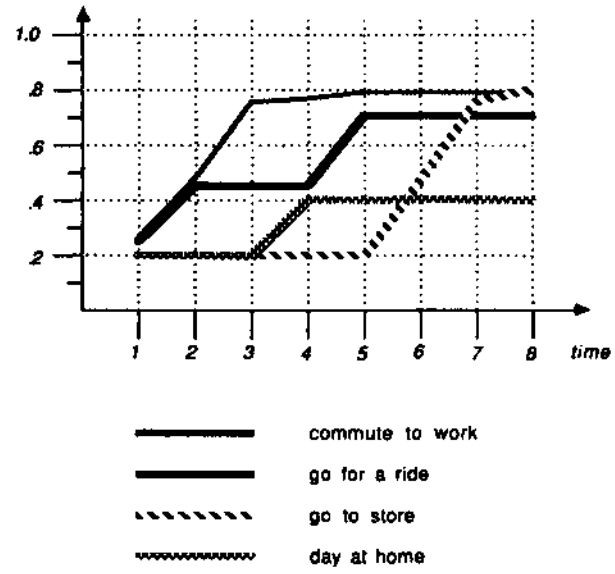

Figure 3. Beliefs vs. time for dynamic assumption

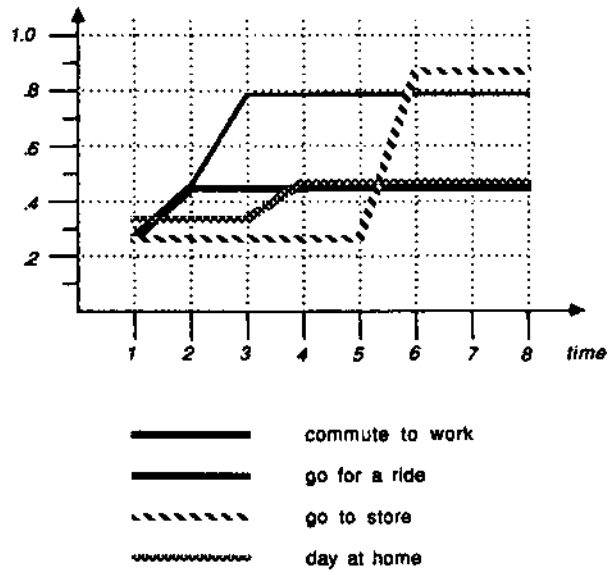

Figure 4. Beliefs for static, more exact, assumption

approach. Fortunately, however, as the Knowledge Base increases in size, encompassing a wider domain of description, the similarity among scripts tends to decrease, yielding closer agreement between the "static" and "dynamic" Belief Functions.

IV. CONCLUSIONS

Script-based reasoning is a powerful paradigm for understanding sequences of events, but to effectively deal with uncertainty and incomplete knowledge, effective belief-management techniques are necessary.

A temporally-ordered input stream, such as a real-time data source, offers constraints on script confidence which, for accuracy and robustness, should be exploited by the belief-management system. Unlike other approaches which do not explicitly address this issue, the probabilistic belief function presented in this paper is one effective way to take advantage of these constraints.

To verify the applicability of the approach, an implementation of the Script Belief functions has been developed. It is part of a system called PUB, the Martin Marietta Pattern Understanding Blackboard, a script-based situation assessment system. In example runs to date, it has provided satisfying belief assignment, as judged by human observers. Additionally, statistical tests have been run to compare the approach to methods using only compatibility-checking Evidential Reasoning (similar to (Lowrance 1986)). Although there is not room in this paper to adequately describe the tests, the simulations have so far indicated improvements in convergence time, especially as the typical script length increases.

Several areas for further development of the Script Belief function are apparent. First the function ideally should take into account the focusing, if any, provided by the data-gathering portion of the system. For example, if most of the attention has, for a time, been allocated to a small part of the domain, matching may take place to scripts relevant to another part of the domain. Such matches should be afforded more belief, as their probability of being observed was lower due to focus elsewhere.

Another useful area for future work is an extension of the Conditional Probability techniques presented here into other uncertainty paradigms. For example, the Dempster-Shafer (Shafer 1976) approach used in some Evidential Reasoning systems (Lowrance 1986) offers a range of belief values, as well as the capability to assign belief to groups of hypotheses. An integration of the approaches, such as by modifying the "rule of combination" to include the new temporal constraints, could provide useful improvements to the script-belief methodology. Similar real-time-conscious extensions to other paradigms could also benefit applications in which those paradigms are well-suited. Progress in such areas promises to provide more robust and general tools for constructing script-based processing systems.

APPENDIX I- DERIVATION OF THE BELIEF FUNCTION

To reduce the complexity of the presentation, some abbreviations are in order.

Let p(kth) be the a priori probability of the occurrence of the kth element in the Knowledge Base.

scripts(k) be the number of scripts in the Knowledge Base with length k,

scripts-in-order(k) be the number of scripts with elements 1 through k in the same order as the script being scored.

The following a posteriori probabilities are assumed given the sensor inputs:

Let P(k) be the a posteriori probability that elements 1 through k of the script will at some time occur in the proper order.

P(kth) be the probability that the kth element will occur.

P(kth by sensors) be the probability that sensor inputs indicate that the kth element has occurred.

conf(kth) be the confidence, supplied by an external observer, that the kth element has occurred (an estimate of P(kth by sensors)).

P(script) be the probability that the abstract event described by the script is actually occurring.

Suppose that the length of the script being scored is n, and that there are #SCRIPTS in the Knowledge Base.

To begin, the a posteriori probability that the script is occurring, given the observed data, depends on the probability of all its elements occurring and on the probability that, if all the events occur, it is really the event described by the script which is occurring:

$$P(\text{script}) = P(\text{script}|n)\,P(n) + P(\text{script}|\overline{n})\,P(\overline{n})$$

However, the second term is zero, because if not all the events occur, by definition the occurrence specified by the script is not occurring.

$$P(\text{script}) = P(\text{script}|n)\,P(n).$$

Now, consider P(n). This can be restated in terms of the probability that all but the last event occurred, modified by the probability that, given this, the last will also occur:

$$P(n) = P(n|n\text{-}1)\,P(n\text{-}1) + P(n|\overline{n\text{-}1})\,P(\overline{n\text{-}1})$$

$$= P(n|n\text{-}1)\,P(n\text{-}1)$$

or, equivalently,

$$P(n) = P(\text{nth} \mid n-1)\ P(n-1).$$

In the same way, P(n-1) can be expressed in terms of the previous script elements, until reaching the first element of the script sequence:

$$P(n) = P(\text{nth} \mid n-1)\ P(\text{n-1st} \mid n-2)\ P(\text{n-2nd} \mid n-3) \ldots\ P(\text{1st})$$

$$= \left[\prod_{k=1}^{n} P(\text{kth} \mid k-1) \right]$$

Now, consider the P(kth|k-1) probability. This can be expressed in terms of the kth element being observed as

$$P(\text{kth} \mid k-1) =$$

$$P(\text{kth} \mid k-1,\ \underline{\text{kth by sensors}})\ P(\underline{\text{kth by sensors}} \mid k-1) + P(\text{kth} \mid k-1,\ \overline{\text{kth by sensors}})\ P(\overline{\text{kth by sensors}} \mid k-1).$$

Because these probabilities are assumed conditional on the sensor inputs,

$$P(\text{kth by sensors} \mid k-1, \text{sensors}) \approx \text{conf}(\text{kth}), \text{ so}$$

$$P(\text{kth} \mid k-1) = \text{conf}(\text{kth}) + P(\text{kth} \mid k-1,\ \overline{\text{kth by sensors}})(1 - \text{conf}(\text{kth}))$$

Static Knowledge Base assumption: Examine the scripts in the Knowledge Base to determine the probability (assuming equal likelihood, unless other statistics available) i.e.,

$$P(\text{kth} \mid k-1,\ \overline{\text{kth by sensors}}) = \frac{\text{scripts-in-order}(k)}{\text{scripts-in-order}(k-1)}$$

Dynamic Knowledge Base assumption: Given the number of scripts of each length, probabilities about scripts meeting the above conditions are found:

P(a script has 1 though k-1 in order) =

$$1 - \prod_{i}^{\substack{\text{scripts with} \\ \text{elements in} \\ \text{order}}} (1 - p(\text{script } i))$$

where p(script i) is the a priori probability that an arbitrary script is script i.

The number of scripts of length L which could have elements 1 through k-1 in correct order is

$$\binom{L}{k-1} \triangleq \frac{L!}{(k-1)!\ (L-k+1)!}$$

If we assume independence in the choice of script elements within each script, the probability of each such script is

$$\prod_{i=1}^{k-1} p(\text{ith})$$

so P(script has elements 1 through k-1 in order) =

$$\sum_{L=1}^{\text{max-L}} \left[1 - \left(1 - \prod_{i=1}^{k-1} p(\text{ith})\right)^{\binom{L}{k-1}}\right] p(\text{length } L)$$

P(script has elements 1 through k in order) is similarly found, using k rather than k-1 in the above equation. It is desirable to include the known script with elements 1 through k in order, namely, the script being scored. Therefore, considering the above probabilities to reflect the occurrence of elements 1 through k elsewhere among possible events,

P(elements 1 through k in order) =

$$\sum_{L=1}^{\text{max-L}} \left[1 - \left(1 - \prod_{i=1}^{k} p(\text{ith})\right)^{\binom{L}{k}}\right] p(\text{length } L) + 1/\ P(\text{elements 1 through k-1 in order})\ \#\text{SCRIPTS}$$

Therefore, the overall probability

$$P(\text{kth} \mid k-1, \overline{\text{kth by sensors}}) = \frac{1 + \sum_{L=1}^{\text{max-L}} \left[1 - \left(1 - \prod_{i=1}^{k} p(\text{ith})\right)^{\binom{L}{k}}\right] \text{scripts}(L)}{1 + \sum_{L=1}^{\text{max-L}} \left[1 - \left(1 - \prod_{i=1}^{k-1} p(\text{ith})\right)^{\binom{L}{k-1}}\right] \text{scripts}(L)}$$

Finally, to obtain P(script),

$$P(\text{script}) = P(\text{script} \mid n)\ P(n)$$

P(script|n) is not, in general, equal to P(n) because if a script which subsumes another occurs, the other script is assumed not to have occurred. Such is the case with the "day at home" script in the example, Fig. 1. Proceeding as before,

Static Knowledge Base assumption:

$$P(\text{script}\,|\,n) = 1 / \text{scripts-in-order}(n)$$

Dynamic Knowledge Base assumption:

$$P(\text{script}\,|\,n) = \frac{1}{1 + \sum_{L=1}^{\text{max-L}} \left[1 - \left(1 - \prod_{i=1}^{n} p(\text{ith})\right)^{\binom{L}{n}}\right] \text{scripts}(L)}$$

In summary,

$$P(\text{script}) = P(\text{script}\,|\,n) \left[\prod_{k=1}^{n} \text{conf}(\text{kth}) + P(\text{kth}\,|\,k\text{-}1, \overline{\text{kth by sensors}}) (1 - \text{conf}(\text{kth}))\right]$$

where $P(\text{script}\,|\,n)$ and $P(\text{kth}\,|\,k\text{-}1, \overline{\text{kth by sensors}})$ are calculated with the static- or dynamic-assumption equations, as appropriate to the particular system.

REFERENCES

1 Azarewicz, J., Fola, G., Fink, R., and Heithecker, C., "Plan Recognition for Airborne Tactical Decision Making," AAAI-86, Vol.2, pp. 805-811.

2 Bozma, I.H., "A Model of Machine Learning Based on Experience, Inference, and Adaptation," M.S. Thesis, Case Western Reserve University, 1985.

3 Fall, T.C., "Evidential Reasoning with Temporal Aspects," AAAI-86, Vol.2, pp. 891-895.

4 Kautz, H.A., and Allen, J.F., "Generalized Plan Recognition," AAAI-86, Vol.1, pp. 32-37.

5 Lebowitz, M., "Generalization and Memory in an Integrated Understanding System," Research Report No. 186, Yale University Department of Computer Science, 1980.

6 Lowrance, J.D., Garvey, T.D., and Strat, T.M., "A Framework for Evidential Reasoning Systems," AAAI-86, Vol.2, pp.896-903.

7 Schank, R.C., Scripts, Plans, Goals, and Understanding, Lawrence Erlbaum Associates, 1978.

8 Shafer, G., A Mathematical Theory of Evidence, Princeton University Press, 1976.

9 Sullivan, M., and Cohen, P.R., "An Endorsement-Based Plan Recognition Program," IJCAI-85, Vol.1, pp. 475-479.