Script-Based Reasoning For Situation Monitoring

Sharon J. Laskowski and Emily J. Hofmann

The MITRE Corporation
C^3I Artificial Intelligence Center
7525 Colshire Drive
McLean, Virginia 22102-3481

Abstract

An expert system that monitors complex activity requires knowledge that is difficult to capture with standard rule-based representations. The focus of this research has been to design and implement script-based reasoning techniques integrated into a rule-based expert system for situation monitoring to address this problem. The resulting expert system, SCripted ANalyst (SCAN), for battlefield monitoring has the capability of reasoning about tactical situations as they develop and providing plausible explanations of activities as inferred from intelligence reports. Sequences of events are monitored through the use of script templates which are matched against events and the time relations between events. SCAN detects causal relations between events, generates multiple hypotheses, fills in information gaps, and sets up expectations about time-dependent events--all features a simple rule-based expert system cannot easily provide.

1. Introduction

This paper describes the research and development of artificial intelligence paradigms and structures needed to build an expert system decision aid for army tactical intelligence staff as they hypothesize about a battlefield situation. In monitoring the situation to support force command and control (C^2) decision-making, a military intelligence analyst must not only interpret the force disposition given reports from multiple sources but must also formulate a sense of how a situation is developing over a long period of time. An expert system designed to monitor sequences of events can help the analyst keep track of the many possible explanations of ongoing actions and intentions and recognize any unusual or unexpected activity.

However, such a system must have internal representations to deal with sophisticated time and order relationships so that it can generate multiple hypotheses, fill in information gaps, and set up expectations. It must notice trends and shifts in the action and activity and provide clear explanations of its inferencing. This inferencing includes the ability to expect that certain events have occurred based on information about related events without necessarily asserting these through the rule base. All this must be done in an environment where information is potentially sparse or misleading.

This research was supported by the Rome Air Development Center, under Contract No. F19628-86-C-0001.

SCripted ANalyst (SCAN) is an expert system with these capabilities made possible by integrating script-based event matching and time reasoning into a previously developed rule-based system, ANALYST, which supplies a situation map of enemy force dispositions and a critical indicator monitoring capability. The main contributions of this research are the script and time representations as coordinated with the rule-based expert system and the construction of an inference mechanism that allows an expert system to reason about causal relationships as recognized from sequences of events. This approach can also be viewed as a first step to plan recognition: detecting the adversary's goals.

The next section discusses the domain, past MITRE developments that are the foundation of our research, and other relevant work. Section 3 outlines the issues that must be addressed for situation monitoring expert systems. Definitions and details of the script representation are presented in Section 4, while the control of the inferencing process is described in Section 5. Finally, we summarize research issues for more extensive applications of script representations and plan recognition to expert systems.

2. Background

Previously, MITRE researchers developed ANALYST [3,6,9], an expert system that is able to infer real-time situation displays from multiple sensor sources and also processes mission-oriented information requests. ANALYST answers these requests with a rule base of static critical indicators that refer to the force disposition. ANALYST is also part of a project to construct a set of cooperating expert systems called ALLIES [5] designed to perform a portion of the C^2 reasoning process. ALLIES includes a military operations planning expert system (OPLANNER) and a object-oriented simulation of the war (Battlefield Environment Model). In the context of ALLIES, it became apparent that ANALYST was not capable of, but had the potential for, in-depth analysis that would give a clearer picture of the adversary's activities and intentions.

The SCAN design was inspired by scripts as applied to natural language processing [10,11]. However, there are few situation monitoring expert systems that have been developed for domains as volatile as the SCAN application. In [4] a plan recognizer with a simple goal detector

for analyzing aircraft threat is described. Blackboard architectures have been used in domains such as speech processing, but these do not fully address the problems that an expert system must handle in a rapidly changing environment. Fall's work [8] uses a representation called a "model" similar to scripts to propagate evidence through time for situation monitoring, but without a robust interface to a rule-based expert system. The Ventilator Manager (VM) program [7] is an example of a MYCIN-like system with an underlying state transition model for interpreting data in an intensive care unit, but was found to be to be inadequate for monitoring data continuously over time. The power of SCAN lies in its ability to monitor continuously changing situations with missing, inaccurate and/or deceptive sensor data and analyze multiple adversaries while at the same time preserving the desirable characteristics of a rule-based expert system.

3. Motivation

In any situation monitoring expert system applied to a rapidly changing environment, time becomes an essential element that must be integrated into the knowledge used to reason about the situation. The causal relationships between events and the expectations of events taking place in specific sequences all play a role in painting a picture of a situation given partial data about a set of activities. It is also crucial to have a well-organized history of events to support any conclusions such a system infers.

ANALYST is goal-driven by user information requests which are static critical indicators represented by propositions. Answers to the requests have likelihood values (Dempster-Shafer likelihood intervals) that indicate the configuration on the battlefield based on the current situation map (SITMAP) of military units. ANALYST can neither recognize activity as signifying an action developing over several snapshots of the SITMAP nor "bootstrap" itself into suggesting some explanation of the progress of the activity and how it relates to other past, present, or future activities.

While it is possible to place the knowledge about sequence of events into ANALYST rules, the rules would be very complex because the sequences are long and the causal dependencies are not always precisely sequential. Events have duration and may overlap in many different ways. Rules would require long chains of antecedents or long chains of rules to hook up these antecedents. Knowledge engineering and debugging these rules would be non-intuitive and quite difficult, and explanations would be confusing to a user. In other words, the major benefits of using an expert system technology over more traditional software techniques would be lost.

For the first design and implementation of SCAN, the research concentrated on the script knowledge representation and control to illustrate that, indeed, the type of knowledge described above could be represented simply. Two major assumptions were made. First, SCAN does no complicated spatial reasoning; interesting areas on the situation map are pre-defined based on the current scenario. This focuses where the major activity is located and simplifies script searching and matching. The second assumption is to use the uncertainty representation (likelihood intervals and likelihood probabilities) currently in ANALYST with little modification. These assumptions are re-examined in Section 6.

4. The Script-Based Approach

The notion of representing sequences of events as templates or scripts is analogous to representing stereotypical information for natural language processing as explored in Schank's research [10]. SCAN is a "goal detector" that could be used to guide the search of a plan recognizer similar to the plan understanding described in [11].

From the SCAN viewpoint, scripts are sequences of events, an event being an activity occurring for a specific duration that is detectable. Each event is described by another script or a proposition and viewed as an indicator that the parent script is occurring. We use this script paradigm to create a knowledge structure that acts as an event template to be matched against a series of time slices comprised of SITMAPs and associated inferences.

This matching process is complex for several reasons. Any tactical maneuver unfolds as a sequence of (possibly) overlapping steps or events and they must be fit together like pieces of a puzzle as they are uncovered. Because events are recognized by SCAN as a discrete measurement of continuous occurrences, the start and end times of events are rough approximations. Matching must then take place based on only guesses as to the ordering and durations of the events. Some events might not have been recognized at all.

4.1 The Script Knowledge Structure

The representation for script knowledge was designed to be expressive enough not only for monitoring and plan recognition applications, but also for ALLIES planning and simulation purposes where the impreciseness of sensors is not a problem. For the purpose of this paper, we illustrate the script knowledge structure and control with a football example. Although a tactical maneuver in football might last for only a few seconds as opposed to several hours in the military domain, the matching algorithms used for monitoring and guessing the adversary's actions are similar.

A script knowledge base is stored as a set of lists in a file which are then accessed through a frame language. Each script entry has the following format:

```
(defscript script-name
        script-elements-list
        script-bindings-list
        necessary-preconditions-list
        sufficient-preconditions-list
        script-analysis)
```

The script-elements-list contains names of sub-scripts which are either pointers to other scripts (and may be used to build up a taxonomy of scripts) or names of propositions which will be monitored as as information requests.

Each element in script-elements-list has the format:

(name type bindings-list preconditions-list)

The type is used to specify whether the element is another script or a proposition. The bindings-list contains variable names and their values, if present. The bindings are used to describe the context of the instantiation of a script or proposition (for example, location, time, or direction of movement) and the default constants such as the duration and weight of an event. The preconditions-list has predicates which may refer to the script-name of other script-elements in the defscript. This preconditions-list is used to specify entry conditions and time relations between the event the element represents and the other events in the parent script. The time predicates are based on Allen's temporal language [1,2].

The script-bindings-list has the same format as the bindings-list of the script-elements-list describing the context of the instantiated script.

The two preconditions-lists are made up of predicates and allow the distinction between necessary preconditions based on predicates that remain static as opposed to dynamic preconditions. For example, a necessary precondition for a particular football play could be the team possessing a specific capability such as an extremely strong running back, whereas sufficient preconditions could be field position, yardage to go, and time remaining in the game in the current context. This allows greater efficiency in searching scripts--there is no need to monitor a script in a context that does not meet the necessary preconditions.

The script-analysis describes the evaluation function used to determine how well a script matches a current situation, that is, to calculate its likelihood.

Typical examples of SCAN football script knowledge are a counter-tray-play and a running-back-fake shown in Figure 1. All time units are in seconds. The counter-tray-play contains six events, two of which are sub-scripts--a running-back-fake and a quarter-back-fake. An instantiation of the counter-tray-play script has several contextual bindings: the field location of the play, the direction of movement, and the time. The likelihood, actual duration, and currently occurring event are all stored under this context in a script instance frame.

```
(defscript counter-tray-play
  (script-name counter-tray-play)
  (script-elements ((end-in-motion proposition
    (>location >direction >time (>duration 3)))
    (snap-ball proposition
    (>location >direction >time (>duration 0.5))
    ((meets end-in-motion)))
    (running-back-fake script
    (>location >direction >time)
    ((after snap-ball 1)))
    (quarter-back-fake script
    (>location >direction >time)
    ((equals running-back-fake)))
    (tackle-pull proposition
    (>location >direction >time (>duration 2))
    ((overlaps quarter-back-fake 0.5)))
    (guard-pull proposition
    (>location >direction >time (>duration 2))
    ((equals tackle-pull))))
  (bindings (>location >direction >time))
  (necessary-preconditions t)
  (sufficient-preconditions (offensive-posture
    short-yardage))
  (script-analysis time-averaged-script-likelihood))

(defscript running-back-fake
  (script-name running-back-fake)
  (script-elements ((running-back-turns proposition
    (>location >direction >time (>duration 0.5)))
    ((running-back-reverses proposition
    (>location >direction >time (>duration 0.5))
    ((meets running-back-turns ))))
  (bindings (>location >direction >time))
  (necessary-preconditions t)
  (sufficient-preconditions (offensive-posture)))
  (script-analysis time-averaged-script-likelihood)))
```

Figure 1
Counter-Tray-Play and Running-Back-Feint Scripts

4.2 Time Representation

There are several issues in time representation that must be dealt with in developing a script knowledge representation and script matching heuristics. Time must be portrayed in a way that captures the "fuzziness" of the domain. An event must not only be recognized as happening with a certain degree of likelihood, but its start and end times must be approximated as well. Each event has a specific duration--it is not enough to postulate a point in time.

The time formalism developed by Allen [1,2] of time intervals and relations between intervals provide a language well-suited to SCAN's domain. There are two items not in this formalism but required by SCAN: the duration of an event--how long (doctrinally) an event is supposed to occur and relative start times between events.

Allen's theory of time is supported by an interval-based temporal logic and a set of properties that can hold over the intervals. In SCAN, the notion of the time of an event is described by a temporal interval, (t_1, t_2) where t_1 is the start time and t_2 is the end time of the occurrence. There is a basic set of relations that can hold between temporal intervals. "Meets" is a primitive relation such that if interval i meets interval j, i's end time is equal to j's start time. Twelve other relations can be described in terms of meet, such as: after, overlaps, equals, meets, and during. For example, in Figure 1, the tackle-pull overlaps the quarter-back-fake by .5 seconds.

We have assumed that the script-elements-list in the defscript is ordered by increasing values of the start times of the script-elements and hence some time relations are implicit. As a result, it is not necessary to list all time relations in the preconditions of an element, only those that cannot be inferred from the preceding element's preconditions and its preceding elements, given the duration information. This simplifies the SCAN scripts and is a

natural way to present the script knowledge, but the relations could be made entirely explicit if this was desirable for a different application.

5. Control Architecture

The control architecture shown in Figure 2 consists of the script inference procedure which includes a script matching function and monitoring facilities. The heart of SCAN lies in the script control. At the end of an ANALYST cycle, that is, reading in a set of reports, creation of a times slice from data fusion, and calculation of the status of the information requests (propositions from script-elements) the script control evaluates the status of the scripts, noting new scripts that have started up and monitoring the likelihood of the scripts already being monitored. All rule-based inferencing is done through the request processing under the guidance of the script control.

As the SITMAP is monitored, each script must be matched against a particular context and time. These two values are used as a key for storing and retrieving the inferences about a script and its associated sub-scripts and propositions. When a script is matched for the first time, it is instantiated with a likelihood.

Script *likelihood* is used to compare how well a script matches the situation relative to other scripts. The likelihood value is between 0 and 1, and a script is typically considered to be occurring if the value is greater than .5. This cutoff value is called the *script-occurring-cutoff*. As likelihoods change from time slice to time slice, they are stored using the script instance's context and the current time so that a history is maintained. The start and end times of scripts and propositions, as well, are calculated based on when the likelihood is above the cutoff.

As SCAN operates, it maintains two lists of scripts. *Monitored-scripts* list are script-context pairs whose necessary preconditions are met. This list is used to guide the search for scripts that match the current situation. *Active-scripts* are script-context-pairs with all preconditions met and likelihoods greater than the script-occurring-cutoff, that is, the scripts that appear to best assess the situation.

Figure 3 shows the control flow of SCAN and how it interfaces with the sensor fusor and request processor. Initially, ANALYST applies its fusion rules to reports creating units on the SITMAP. Then, the monitored-scripts list is constructed by searching all scripts on all predefined areas of interest for those scripts whose necessary preconditions are met. As a script is added to the list, all script-elements that are propositions are set up as information requests whose values will be monitored across time slices. The initial processing of the requests for the first time slice involves backward chaining to calculate all their likelihoods.

After the initialization, the active-scripts list is built by searching through the monitored-scripts for scripts whose sufficient preconditions are met, calculating a likelihood for each of those scripts and placing a script on the active-scripts list if its likelihood is high enough.

When the monitoring phase is entered, a new time slice is built and all information requests are re-evaluated for changes through a forward chaining inference procedure. (Only rules that will alter the likelihoods of the current information requests are fired.) With the updated information available, each active script is re-evaluated to update its likelihood and determine if it should remain active. Because new scripts might be starting up at any point, the monitored-script-list is examined again. At any point in time, the active-scripts can be interpreted as a set of hypotheses that explain the enemy's intentions.

Script likelihood calculation is based on the likelihoods of the script-elements, the time relations between elements, the history of likelihood values, and a *current-element-pointer* denoting the script-element in progress, if

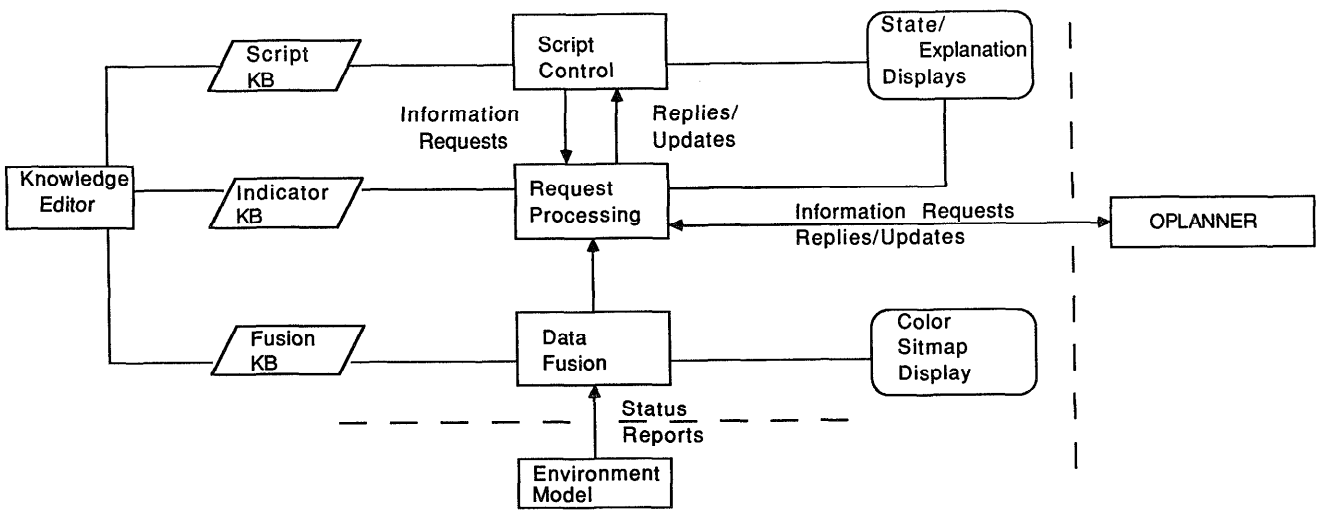

Figure 2
SCAN System Architecture

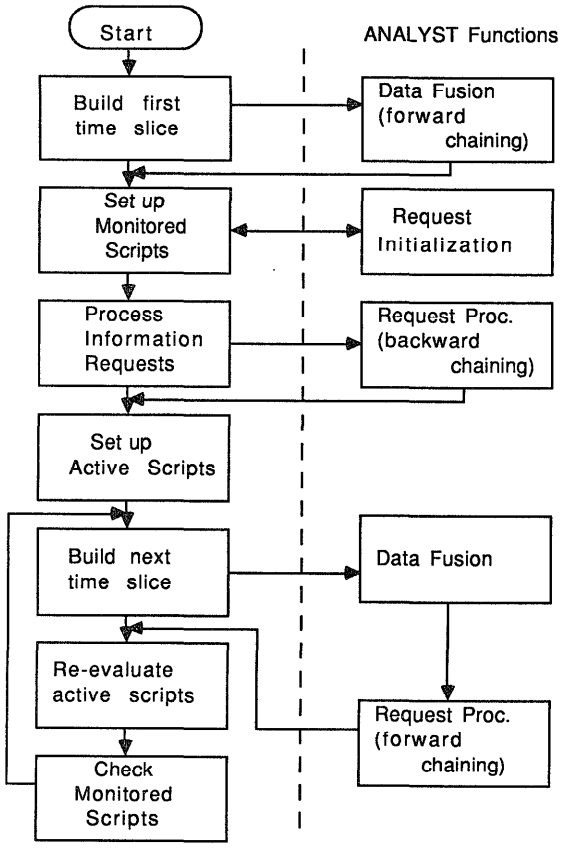

Figure 3
SCAN Script Control Flow

combined to determine a script's overall likelihood using weights specified in the bindings-list of the script and the function (for example, averaging) specified in the script-analysis.

6. Conclusions and Future Directions

Scripts do, indeed, provide a more intuitive knowledge representation for situation monitoring expert systems. SCAN as an implementation of the script mechanism is able to recognize trends on the battlefield. SCAN has a time representation and an interface to the rule-based ANALYST system that allows it to focus on the development of activities over time. However, additional experimentation is needed to explore the knowledge engineering of scripts. SCAN, at present, is very limited in its number of scripts and its matching techniques.

The assumptions that have been made for this first version of SCAN need to be re-examined if SCAN is to be more that a toy system. Spatial and terrain reasoning is an important aspect of SCAN's domain yet has been virtually ignored. The techniques for reasoning about uncertainty are quite *ad hoc* and should be changed to a representation with a firmer mathematical footing to avoid inconsistency and potential anomalies.

A goal for SCAN (and one reason for wanting intuitive knowledge representations) is to allow the expert to knowledge engineer SCAN directly. A knowledge editor with a sophisticated human-machine interface would be a step closer to achieving this.

Finally, SCAN represents only part of the software that is necessary to generate and recognize details of plausible plans. Future work includes building a plan recognizer based on the planning techniques in the OPLANNER plan generator and guided by SCAN's script hypotheses to constrain the plan search space.

References

[1] Allen, J. F., Towards a general theory of action and time, *Artificial Intelligence* 23 (1984) 123-154.

[2] Allen, J. F. and Hayes, P. J., A common sense theory of time, IJCAI (1985) 528-531.

[3] Antonisse, H. J., Bonasso, R. P., and Laskowski, S. J., ANALYST II: a knowledge-based intelligence support system, MITRE Technical Report MTR-84W00220, April 1985.

[4] Azarewicz, J., et. al., Plan recognition for airborne tactical decision-making, AAAI (1986), 805-811.

[5] Benoit, J. W. et. al., An experiment in cooperating expert systems for command and control, Expert Systems in Government Conference, October 1986.

[6] Bonasso, R. P., ANALYST: An expert system for processing sensor returns, The First Army Conference on Knowledge-Based Systems for C^3I, Army Model Management Office, Ft. Leavenworth, November 1981, 219-245.

[7] Buchanan, B. G., and Shortliffe, E. H., eds., *Rule-Based Expert Systems*, Addison-Wesley Publishing Co., 1984.

[8] Fall, T. C., Evidential reasoning with temporal aspects, AAAI (1986) 891-895.

[9] Laskowski, S. J., Antonisse, H. J., and Bonasso, R. P., ANALYST II: A knowledge-based intelligence support system, Second IEEE Conference on Artificial Intelligence Applications, December 1985, 552-563.

[10] Schank, R. and Abelson, R. *Scripts, Plans, Goals and Understanding*, Lawrence Erlbaum Associated, Inc., 1977.

[11] Wilensky, R, *Planning and Understanding: A Computational Approach to Human Reasoning*, Addison-Wesley Publishing Co., 1983.