American Journal of Computational Linguistics *Microfiche 35 : 50*

An Approach to the Organization of Mundane World Knowledge: The Generation and Management of Scripts

R. E. Cullingford

Yale University
New Haven, Connecticut 06511

ABSTRACT

In understanding stories or natural-language discourse, hearers draw upon an enormous base of shared world knowledge about common situations like going to restaurants, theaters or supermarkets to help establish the needed context. This paper presents an approach to the management of this type of knowledge based upon the concept of a situational script [Schank and Abelson, 1975]. The application of scripts in story understanding is illustrated via a computer model called SAM (Script Applier Mechanism).

In simple one-script stories, SAM constructs a trace through a preformed data structure containing the input, other events not mentioned but commonly assumed, the important

---

The research described in this paper was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract N00014-75-C-1111.

inferences associated with the events, and the interconnecting causal links. In more complicated stories, SAM handles the invocation and closing of parallel, nested and sequential scripts.

1.Ø Introduction

Natural-language processing research in recent years has increasingly focussed upon the modeling of human world knowledge and management of the resulting data base (1). This has come about largely because of the enormous problems encountered in the processing of texts, as opposed to single sentences, by traditional methods based upon syntactic analysis and low-level semantics. This state of affairs should not be surprising, since it is quite clear that people draw upon a huge store of shared, extra-linguistic world knowledge in understanding even the simplest stories or engaging in the most rudimentary conversation.

Much of the knowledge that hearers utilize to establish the background or context of a story appears to be episodic in nature, distilled from many experiences in common situations like going to restaurants, football games and supermarkets. This paper presents an approach to the representation and handling of this type of mundane world-knowledge based upon the concept of a situational script [Schank and Abelson, 1975]. The application

---

(1) See, for example, the emphasis on this area in "Theoretical Issues in Natural Language Processing", Proceedings of the Interdisciplinary Workshop in Computational Linguistics, 1Ø-13 June 1975, Cambridge, Mass.

of scripts in story understanding will be illustrated via a computer model called SAM (Script Applier Mechanism) devised for a PDP-10 computer at Yale University.

1.1 Examples of Story Understanding in SAM

Before attacking the various aspects of scripts and the particular activities carried on by SAM's script applier, it seems appropriate to give some examples of input and the various outputs that SAM produces. This will give the reader some feeling for the kinds of heavily script-based stories SAM can handle and the understanding tasks it performs.

We present three different types of story input. The first story makes references to a common script, the restaurant script, then stops the script in midstream:

Story I
John went to a restaurant. He sat down. He got mad. He left.

A 'long paraphrase' output generated by SAM for this story is as follows:

Long Paraphrase
John was hungry. He decided to go to a restaurant. He went to one. He sat down in a chair. A waiter did not go to the table. John became upset. He decided he was going to leave the restaurant. He left it.

Next is a typical boring restaurant story conforming closely to expected scriptal data:

Story II
John went to a restaurant. The hostess seated John. The hostess gave John a menu. John ordered a lobster. He was served quickly. He left a large tip. He left the restaurant.

SAM has generated summary and question-answering output for this story:

Summary

John went to a restaurant and he ate a lobster.

Question-Answering

Q: Why did the hostess give John a menu?
A: So John could ask the waiter for a meal.

Q: Why did John go to a restaurant?
A: So John could eat a meal.

Q: Did the waiter give John a menu?
A: John got the menu from the hostess.

Q: Why did John leave the waiter a large tip?
A: Probably John remembered the waiter served a lobster to him quickly.

Q: Did John pay the bill?
A: John paid the check.

The system has also provided long-paraphrase outputs for this story, not shown here, in both English and Chinese.

The final example invokes several scripts, and calls up an unusual path in one script because of an odd occurrence in an earlier one:

Story III
John went to New York by bus. On the bus he talked to an old lady. When he left the bus, he thanked the driver. He took the subway to Leone's. On the subway his pocket was picked. He got off the train and entered Leone's. He had some lasagna. When the check came, he discovered he couldn't pay. The management told him he would have to wash dishes.

Long Paraphrase

John went to a bus stop. He waited at it a few minutes. He entered a bus. The driver got the ticket from John. He went to a seat. He sat down in it. While John was on the bus an old lady and John talked. The driver took John to New York. He went to the driver. While getting off the bus John thanked the driver. John got off it.

He entered a station. He put a token in the turnstile. He went to the platform. He waited at it a few minutes. He entered a subway car. A thief went to John. The thief picked John's pocket. He went. John went to the seat. He sat down in it. The driver took John to Leone's. He left the subway car. He left the station.

He entered Leone's. He looked around inside it. He saw he could go to a table. He went to it. He sat down in the seat. He ordered some lasagna. The waiter indicated to the chef John would like him to prepare something. The chef prepared the lasagna. The waiter got it from the chef. The waiter went to the table. He served the lasagna to John. He ate it. He became full.

He asked the waiter for the check. John got it from the waiter. John read the check. John discovered he was unable to pay the check. He indicated to the waiter he was unable to pay the check. The management told John he would have to wash dishes. He entered the kitchen. He washed dishes. He left Leone's.

[paragraphing has been added to the computer output for ease of reading]

In these example stories, SAM analyzes each input sentence into a Conceptual Dependency (CD) representation. If this representation fits a script, that script is called into memory and successive inputs are matched in the script and linked up by a SAM program called the script applier. The script applier output is processed by other SAM programs depending on the type of final output desired, and English or, for Story II, Chinese is generated. The point to be stressed is that all the

'understanding' processing is done on a single data structure, the story representation constructed by the script applier. We discuss in particular the scriptal data base, the script applier and the story representation in succeeding sections. Additional details on the other parts of SAM can be found in [Schank et al, 1975].

2.Ø Situational Scripts

As implemented in SAM, a situational script is a network of CD patterns describing the major paths and turning points commonly understood by middle-class Americans to occur in stereotyped activities such as going to theaters, restaurants and supermarkets. The script idea is very similar to the independently developed 'frame system' for story understanding described in [Charniak, 1975], which is itself based loosely on the 'frame' concept [Minsky, 1974] currently used in vision research.

The patterns provided in scripts are of two general kinds: events, which we will construe broadly as including states and state-changes (2) as well as mental and physical ACTs; and causal relations among these events [Schank, 1973 and 1974].

---

(2) Certain actions like driving a car or preparing food involve complex, learned sensory-motor skills as well as scriptal knowledge. Such actions are summarized within a script as a causal relation terminating in the chief state-change effected by the action. For example, the sentence "The cook prepared the meal" is represented in LISP CD format as:

```
((CON ((ACTOR (*COOK*) <=> (*DO*)))
      LEADTO
 ((ACTOR (*MEAL*) LEAVING (*COOKSTATE* VAL (Ø))))))
```

Patterns are used in scripts not only because of the variety of possible fillers for the roles in scripts, but also to constrain the amount of information needed to identify a story input. Thus, for example, the script provides a LISP CD template like:

```
((ACTOR (X) <=> (*PTRANS*) OBJECT (X) TO (*INSIDE*
PART (RESTAURANT))))
```

to identify inputs like:

John went into Leone's.
John walked into Leone's.
John came into Leone's from the subway.

(X and RESTAURANT are dummy variables). This allows the script applier to ignore inessential features of an input (like the Instrument of the underlying ACT or the place John came from in the examples given above), and thus provides a crude beginning for a theory of forgetting.

In the present implementation, SAM possesses three 'regular' scripts, for riding a bus, for riding a subway, and for going to a restaurant (3). These scripts have been simplified in various ways. For example, all of them assume that there is only a single main actor. The bus script has been restricted to a single track' for a long-distance bus ride, and the restaurant script does not have a 'McDonald's' or a 'Le Pavillon' track. This was done primarily to have a data base capable of handling specific stories of interest available in a reasonable time, secondarily to limit the storage needed (4). Nevertheless, as

---

(3) The data base also contains script-like structures for 'weird' or 'unusual' happenings like the main actor's becoming ill, or, as in Story III, having his pocket picked. Such activities could be handled by a generalized inferencing program like the one described in [Rieger, 1975].

the examples of Section 1.1 indicate, the current scripts are a reasonable first pass at the dual problems of creating and managing this type of data structure.

2.1 Goals, Predictions and Roles in Scripts

Each situational script supplies a default goal statement which is assumed, in the absence of input from higher level cognitive processes like 'planning' [Schank and Abelson, 1975], to be what a story referring to a script is about. The restaurant script for example, defines the INGEST and the resulting state-change in hunger as the central events of a story about eating in restaurants. Closely related to the goal statement is the sequence of mutual obligations that many scripts seem to entail. Invoking the bus script, for example, implies the contract between the rider and the bus management of a PTRANS to the desired location in return for the ATRANS of the fare. Such obligations have a powerful influence on the predictions the system makes about new input. In the restaurant context, for example, an input referring to an event beyond ordering or eating is not initially expected, because these events form the initial statement of obligation. Thus the system takes longer to identify a story sequence like:

John went to a diner. He left a large tip.

Once an input about ordering has been processed, SAM is prepared

---

(4) The text for the restaurant script, presently the largest of the scripts, occupies roughly 100 blocks of PDP-10 disk storage, or about 64,000 ASCII characters.

to hear about the preparation and serving of food, actions associated with eating, or paying the bill, but not about leaving the restaurant. This is because the main actor has not fulfilled the other half of the obligation.

The binding of nominals in the story input to appropriate fillers in the script templates is accomplished in SAM by means of script variables with associated features. In the rather crude system of features presently used, each script variable is assigned a superset membership class: e. g., a hamburger is a 'food', while a waiter is a 'human'. certain variables are also given roles: e. g., a hostess or a waiter can fill the 'maitre'd' role. The former property would enable the system to distinguish between "The waiter brought Mary a hamburger" and "The waiter brought Mary the check". The latter property identifies important roles in script contexts, primarily those to which it is possible to make definite reference without previous introduction, like 'the driver', 'the cook' or 'the check'. For stories in which certain script variables are not bound, the system provides a set of default bindings for the roles not mentioned: thus, SAM fills in 'meal' for a story in which the food ordered is not explicitly named. Variables without distinguished roles default to an indefinite filler, like 'someone' for the main actor.

2.2 Script Structure

Each SAM script is organized in a top-down manner as follows: into tracks, consisting of scenes, which are in turn

composed of subscenes. Each track of a script corresponds to a manifestation of the situation differing in minor features of the script roles, or in a different ordering of the scenes. So, for example, eating in an expensive restaurant and in McDonald's share recognizable seating, ordering, paying, etc., activities, but contrast in the price of the food, type of food served, number of restaurant personnel, sequence of ordering and seating, and the like. Script scenes are organized around the main top-level acts, occurring in some definite sequence, that characterize a scriptal situation. The giving of presents, for example, would be a scene focus in a birthday party script, but putting on a party hat would not be. The latter would correspond to a subscene, perhaps within the 'preparing-to-celebrate' scene of that script. In general, subscenes are organized around acts more or less closely related to the main act of the scene, either contributing a precondition for the main act, as walking to a table precedes sitting down; or resulting from the main act, as arriving at the desired location follows from the driver's act of driving the bus. An intuitive way of identifying scene foci and scene boundaries is to visualize a script network of interwoven paths. In such a network, the scene foci would correspond to points of maximum constriction; scene boundaries to points of most constriction between foci. This essentially means that all paths through a scene go through the main act (except abort paths, discussed below), and relatively few events are at scene edges.

It is necessary, therefore, to distinguish certain events in a script: scripts, their tracks, scenes and subscenes all have 'main', 'initial' and 'final' events. For example, the main event of the 'ordering' event in a restaurant is the ordering act itself; an initial event is reading the menu; and a final event is the waiter telling the cook the order. Additionally, scripts and tracks have associated 'summaries', which refer to a script in general terms. Consider, for example, the following sentence from Story III: "John went to New York by bus". This sentence is marked in the underlying meaning representation by the SAM analyzer as a summary because of the presence of:

((ACTOR (*JOHN*) <=> (*SDO*) OBJECT ($BUS)))

in the Instrument slot (5). Such sentences have two common functions in simple stories. They may indicate that a script was invoked and completed, and no further input should be expected for this instance of the script. This function of the summary often occurs with scripts (like those associated with travelling) which tend to be used as 'instruments' of other scripts (as in getting to a restaurant or store). Alternatively, they may signal that a wider range of possible next inputs is to be expected than would be predicted if the script were entered via an initial event. For example, the story sequence initiated with a summary:

> John took a train to New York. While leaving the train, he tipped the conductor.

---

(5) The primitive ACT SDO is an extension of the primitive dummy CD ACT DO, and stands for an actor performing his script for a given situation, in this case the bus script ($BUS).

sounds more natural than a sequence beginning with an initial event:

> John got on a train. While leaving the train, he tipped the conductor.

These two functions of the summary contrast widely in the range of predictions they invoke. However, additional inputs after a summary, as in the example above, often give the psychological feeling of 'afterthoughts'.

Scenes are built up out of subscenes, which usually contain a single chunk of causal chain or 'path'. In SAM scripts, these paths are assigned a 'value' to indicate roughly their normality in the scriptal context. Several pathvalues have been found useful in setting up the story representation. At one end of the normality range is 'default', which designates the path the script applier takes through a scene when the input does not explicitly refer to it. For example, the input sequence:

> John went to Consiglio's. He ordered lasagna.

makes no mention of John's sitting down, which would commonly be assumed in this situation. The system, following the default path, would fill in that John probably looked around inside the restaurant, saw an empty table, walked over to it, etc. Next on the normality scale is 'nominal', designating paths which are usual in the script, not involving errors or obstructions in the normal flow of events. The sentences in Story II which refer to the hostess are examples of nominal inputs. Finally, there are the 'interference/resolution' paths in a script. These are followed when an event occurs which blocks the normal functioning

of the script. In a restaurant, for example, having to wait for a table is a mild interference; its resolution occurs when one becomes available. More serious because it conflicts directly with the goal/obligation structure of the script is the main actor's discovery that he has no money to pay the bill. This is resolved in Story III by his doing dishes. An extreme example of an interference is the main actor's becoming irritated when a waiter fails to take his order, as in Story I, followed by his leaving the restaurant. When this happens, the script is said to have taken an 'abort' path.

In addition to the above, certain incomplete paths, i. e., paths having no direct consequences within the script, have been included in the scriptal data base. The most important of these incomplete paths are the inferences from, and preconditions for, the events in the direct causal paths. Lumped under the pathvalue 'inference', these subsidiary events identify crucial resultative and enabling links which are useful in particular for question-answering [Lehnert, 1975]. For example, the main path event 'John entered the train' has attached the precondition that the train must have arrived at the platform, which in turn is given as a result of the driver's bringing the train to the station. Similarly, a result of the main path event 'John paid the bill' is that he has less money than previously. Both of these types of path amount to a selection among the vast number of inferences that could be made from the main path event by an inferencing mechanism like Rieger's Conceptual Memory program [Rieger, 1975].

A special class of resultative inferences are those common events which are potentialized by main path events, though they may not occur in a given story. Labelled with the pathvalue 'parallel', these events may either occur often in a specific context without having important consequences, as in "The waiter filled John's water glass"; or they may happen in almost any context without contributing much to the story, as in the sentence "On the bus, John talked to an old lady", from Story III. Since such parallel paths often lead nowhere, they are good candidates for being forgotten.

3.0 The Script Applier

Construction of a story representation from CD input supplied by the SAM analyzer is the job of the script applier (6). Under control of the SAM executive, the applier locates each new input in its collection of situational scripts, links it up with what has gone before, and makes predictions about what is likely to happen next. Since the SAM system as a whole is intended to model human understanding of simple, script-like stories, the script applier organizes its output into a form suitable for subsequent summary, paraphrase and question-answering activities.

In the course of fitting a new input into the story

---

(6) The current version of the applier is programmed in MLISP/LISP 1.6 and runs in an 85K core image on a PDP-10 computer. Processing of Story III, the longest story attempted to date, took approximately 8 minutes with SAM as the single user of the timesharing system.

representation, the applier performs several important subtasks. Identifying an input often requires an implicit job of reference specification. For example, in the sentence from Story III beginning "When the check came...", there is surface ambiguity, reflected in the parser's output, regarding donor and recipient. This ambiguity is settled in the restaurant context by the assumption that the recipient is the main actor and that the donor is a member of the restaurant staff, preferably the waiter. An allied problem arises when the applier, in placing a new conceptualization in the story representation, determines the relevant time relations. Certain types of time data are computed from the output conceptualization itself: for example, the relation between an MTRANS and its MOBJECT, which may determine whether 'remember' or 'ask for' is appropriate in the final output. Other time relations are defined by the causal structure of the script itself: thus 'eating' follows 'ordering'.

More complex time-order computations have to be made when the applier identifies two or more 'simple' conceptualizations in a compound input derived from sentences containing ambiguous words like 'during' or 'when'. Examples of this were encountered during the processing of Story III, for example, in the sentence 'When he left the bus, he thanked the driver'. The system resolves this compound input into the plausible sequence of a PTRANS to the driver, the MTRANS of the 'thanking , and the PTRANS off the bus.

3.1 Story Representation

The output of the script applier consists of linked story segments, one per script invoked, giving the particular script paths traversed by the input story. The backbone of the story representation is the eventlist of all the acts and state-changes that took place. The eventlist is doubly linked, causally and temporally, with the type of causation and time relations filled in within a story segment by the applier.

Attached to the eventlist are the appropriate, instantiated preconditions, inferences and parallel events for each main path event. As discussed above, the inferences and preconditions have been selected for their expected utility in question-answering.

Each story segment is identified by a label which gives access to important properties of the segment: what script it came from; what the particulars were of the script summary, maincon, entrycon, and exitcon this time through; and what interference/resolution cycles were encountered. Additionally, pointers are provided to extra-scriptal 'weird' events that happened in the story. At the top, the global identifier STORY gives the gross structure of the story in terms of sequential, parallel and nested scripts and the weird things. This hierarchical organization facilitates summary and short paraphrase processing, while retaining the fine structure needed for extended paraphrasing and question-answering.

Story III illustrates most of the present capabilities of the SAM script applier in story understanding. The applier accepts a CD representation of the nine sentences in turn from the analyzer and builds an eventlist consisting of 56 main path conceptualizations and 39 associated preconditions/inferences. The 'parallel' events of John talking to the old lady and the bus driver also appear in the eventlist. The eventlist is divided into four story segments, one each for the bus, subway and restaurant scripts and one for the 'weird' robbery event. The identifier for the subway segment is marked as containing the weird event, as is the global STORY. The restaurant segment contains the interference/resolution pair 'unable to pay/wash dishes'. Additionally, the lack of money encountered during the paying scene was checked with the SAM executive during the processing of Story III, since it violates one of the prime preconditions of the restaurant script. Since the executive found that the loss of money was a consequence of the stealing event that occurred earlier, this event is not marked as weird. Appropriate summaries are provided for each story segment. At the top, STORY contains the information that the four segments are organized as a sequence of bus, subway and restaurant, with the pickpocket event nested inside the subway segment.

4.0 Future Work

As the examples show, SAM is capable of handling fairly complex stories in its present state of development. However, several extensions and additions to the scriptal data base and the script applier appear to be needed before SAM can achieve its

ultimate potential.

First, a more flexible method of pattern-matching is required so that the full diversity of input role-fillers can be accommodated. A method of comparing features of nominals in the parser output to the appropriate script variables is needed so that over- or underspecified inputs can be correctly identified. For example, the applier should be able to recognize the phrase 'the restaurant' as a partially specified instance of 'Leone's', found earlier.

As an extension of this, input conceptualizations of a descriptive nature (e. g., "The restaurant was of red brick") need to be processed in a way that allows the system to update its 'image' of the role-fillers in a script. The facilities needed are similar to those provided by the 'occurrence set' in Rieger's Conceptual Memory program [Rieger, 1975].

The most important problem to be faced, however, is the generalization of the story representation to handle stories with several main actors, or with non-synchronous events. It is clear that the simple linear eventlist structure described in Section 3.1 would not be adequate for even such a simple story sequence as:

> "The cook made the lasagna. Meanwhile the wine steward poured the wine."

4.1 Acknowledgement

The programs discussed here are only a part of the SAM system, and a great deal of credit is due to my co-workers in the

Yale AI Project: to Professors Roger Schank and Bob Abelson for the theory on which SAM is based and for their overall guidance; to Dr. Chris Riesbeck for valuable discussion and criticism, as well as a substantial part of the programming effort; and to Gerry DeJong, Leila Habib, Wendy Lehnert, Jim Meehan, Dick Proudfoot, Wally Stutzman and Bob Wilensky.

References

Schank and Abelson 1975
R. C. Schank and R. P. Abelson, "Scripts, Plans and Knowledge", Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 1975.

Schank 1973
R. C. Schank, "Causality and Reasoning", Technical Report No. 1, Instituto per gli studi semantici e cognitivi, Castagnola, Switzerland, 1973.

Schank 1974
R. C. Schank, "Understanding Paragraphs", Technical Report No. 6, Instituto per gli studi semantici e cognitivi, Castagnola, Switzerland, 1974.

Schank et al 1975
R. C. Schank and the Yale AI Project, "SAM--A Story Understander", Research Report No. 43, Yale University Department of Computer Science, 1975

Lehnert 1975
W. P. Lehnert, "What makes SAM run? Script-Based Techniques for Question Answering", Proceedings of the Conference on Theoretical Issues in Natural Language Processing, edited by R. Schank and B. Nash-Webber, 1975.

Charniak 1975
E. Charniak, "Organization and Inference in a Frame-Like System of Common Sense Knowledge", Proceedings of the Conference on Theoretical Issues in Natural Language Processing, edited by R. Schank and B. Nash-Webber, 1975.

Minsky 1974
M. Minsky, "Frame-Systems", MIT AI Memorandum, 1974.

Rieger 1975
C. Rieger, "Conceptual Memory", in Conceptual Information Processing, R. Schank (ed.), North Holland, 1975.