INTERACTIVE SCRIPT INSTANTIATION

Michael J. Pazzani
The MITRE Corporation
Bedford, MA 01730

ABSTRACT

The KNOBS [ENGELMAN 80] planning system is an experimental expert system which assists a user by instantiating a stereotypical solution to his problem. SNUKA, the natural language understanding component of KNOBS, can engage in a dialog with the user to allow him to enter components of a plan or to ask questions about the contents of a database which describes the planning world. User input is processed with respect to several knowledge sources including word definitions, scripts which describe the relationships among the scenes of the problem solution, and four production system rule bases which determine the proper database access for answering questions, infer missing meaning elements, describe how to conduct a conversation, and monitor the topic of the conversation. SNUKA differs from GUS [BOBROW 77], a dialog system similar to SNUKA in its goals, in its use of a script to guide the conversation, interpret indirect answers to questions, determine the referents of nominals, perform inferences to answer the user's questions, and decide upon the order of asking questions of the user to maintain a coherent conversation. SNUKA differs from other script-based language understanders such as SAM [CULLINGFORD 78] and FRUMP [DEJONG 79] in its role as a conversational participant instead of a story understander.

I INTRODUCTION

In this paper, we wish to illustrate the knowledge sources necessary to participate in a type of natural language dialog. The type of dialog we wish to consider occurs when a person whom we will call "the planner" asks a person (or computer) whom we will call "the servant" to perform or arrange for the performance of a commonly occuring activity. Conversations of this sort are often held with travel agents, secretaries, military aids, building contractors, stockbrokers, etc. Most importantly, as the fields of knowledge based expert systems and natural language understanding emerge, these conversations will occur with a computer playing the part of the servant. The objective of this type of conversation is for the planner to describe to the servant the type of activity to be carried out, and to name the particular objects and persons participating in this activity. The servant is often capable of criticizing the planner's plan, or answering questions about objects and the roles they can play in the plan. We will call the processing the servant does in this type of conversation interactive script instantiation.

We utilize scripts as the knowledge source for participating in these stereotypical conversations to contain and control the inference processes needed during understanding. Three situations which require inference during dialog understanding have been identified: the understanding of indirect answers to questions, the memory search required to answer questions and the recognition of intentions from questions or requests. In addition, the script can serve as the model of a plan which is being generated. The model is used to help conduct a coherent conversation. We have also introduced the notion of conversation rules to help the servant decide how to generate responses and take initiative in the conversation. We have implemented our ideas in a computer program called SNUKA, the natural language dialog component of the KNOBS system.

II APPLICATION

The KNOBS system supports a planner by checking the consistency of plan components, enumerating or ranking possible choices for plan components, or automatically generating a complete plan. KNOBS embodies a model of stereotypical planning, where a generic solution to the plan is known and the planner's task is to instantiate an individual plan solution with a consistent set of plan variable bindings. (The plan variables are represented as slots of FRL frames [ROBERTS 77].) The current application domains of KNOBS are the planning of an Air Force tactical air mission, the planning of a Naval "show the flag" mission and the scheduling of crew activities of the NASA space shuttle. A sample conversation with KNOBS demonstrates both its planning and its conversational capabilities:

1. USER: I want to send 4 aircraft from the 707th Fighter Wing to strike Boston between 1140 and noon.
2. KNOBS: Which aircraft do you want to send?
3. USER: What aircraft can the wing supply?
4. KNOBS: The 707th Fighter Wing has F-4s.
5. KNOBS: Would you like to use F-4s for the aircraft?
6. USER: No, F-16s from the 606th Fighter Wing.
7. KNOBS: The 606th Fighter Wing does not contain F-16s.
8. USER: Where are there F-16s?
9. KNOBS: There are F-16s at Kennedy and LaGuardia.
10. USER: What wings at LaGuardia have them?
11. KNOBS: The 808th Fighter Wing and the 909th Fighter Wing have F-16s.
12. USER: Fly them out of the 808th Fighter Wing at LaGuardia.
13. KNOBS: What weapon system should the F-16s carry?
14. USER: Which are acceptable?
15. KNOBS: The acceptable weapon systems are D1, D2, and D4.
16. USER: Hit the target with D4 and plan the rest.
17. KNOBS: The time over target is 1150.
The time of departure is 1120.
The call sign is PEACE 7.
The transponder code is 5351.

Sentences (7), (15) and (17) respectively illustrate the constraint checking, enumeration, and automatic planning aspects of the KNOBS system. (1) is an example of entering a range as a restriction for a plan variable value which is later refined by KNOBS. (In addition to dialog-driven planning, the KNOBS system provides a menu-based interface with these same capabilities.)

SNUKA uses APE-II [PAZZANI 83], a conceptual dependency (CD) [SCHANK 72] parser, to produce a meaning representation of user input by consulting expectations from word definitions, expectations from scripts, expectations activated by system generated questions, and a database of object primitives [LEHNERT 78b]. Once the meaning of the user's utterance has been represented, an appropriate response is produced by means of a set of conversational rules. Simple rules state such things as that questions should be answered; more complex rules monitor the topics of the conversation and activate or deactivate expectations when the topic changes. A question is answered by invoking productions which are composed of three facets: a test, which is a CD pattern which typically binds variables to the major nominals in a question conceptualization; an action, which utilizes the referents of these nominals to produce a database query; and a response which is a template English sentence in which the answer and referents are inserted. (SNUKA currently does not include a natural language generation component.) This approach to question answering is discussed in further detail in [PAZZANI 83].

The remainder of this paper will first describe some prior related work, and then discuss the method of understanding the user's commands and replies as well as the method of making inferences when required to interpret a user's query.

III RELATED WORK

GUS is a frame-driven dialog system which performs the role of a travel agent. It operates by instantiating a frame and sequentially finding fillers for the slots of this frame, directed by a specification found in its prototype. When this specification instructs GUS to ask the client a question, GUS activates an expectation which is a skeletal sentence in which the client's response can be inserted to handle ellipsis by forming a syntactically complete sentence. The client's response is expected to specify a value for this slot, although the client can gain some initiative by asking a question or supplying values for other (or additional) slots.

The DM system [STEINBERG 80] is a dialog moderator which decides upon the order in which to ask questions of a user who is entering program specifications. The approach taken by DM is to produce as many potential questions as possible and select one to ask whose subject keeps the dialog "well structured" according to a set of rules specific to the domain of program specification.

ACE [CULLINGFORD 82] is a conversation program which performs the task of a faculty advisor assisting a computer science student to register for classes. ACE is similar to SNUKA in that it uses a CD parser to produce a meaning representation of the student's input, and can make inferences to answer the student's questions while the student attempts to fill out a course schedule. ACE utilizes a backward chaining deductive retrieval mechanism [CHARNIAK 80] as its means of inference.

Lehnert's QUALM system [Lehnert 78a] is a question answering program which can consult scriptal knowledge constructed during story understanding to answer questions about stories. The use of inferential analysis to "recategorize" utterances in QUALM is represented by conversational rules in SNUKA. Lehnert also introduces the notion of conversational scripts which differ from our work in that they categorize knowledge necessary to conduct a conversation while participating in a stereotypical situation as opposed to the planning of the situation.

IV SNUKA

As used by SNUKA, a script is a stereotypical means of accomplishing a goal. The script roles (i.e., the objects and actors usually implicated in a particular context) correspond to the plan variables in the KNOBS system. An important difference between the frames of GUS and scripts is that scripts are composed of causally and temporally chained scenes [SCHANK 77] which represent the relationships among the states and

actions which occur in a context. This knowledge can be used to help answer the user's questions, to extract the bindings of plan variables (script roles) from the user's utterances, to determine the questions which must be asked of the user and the order in which they are asked, and to assist the parsing process in selecting word senses and pronominal referents when analyzing the user's input.

A. Inference

The question answering component of SNUKA can use the inference patterns of the script to respond to an extended range of questions. Because the KNOBS databases are static descriptions of the attributes of objects, question answering rules typically refer to states. Many questions, however, refer to actions which are enabled by states described in the database. If a question answering question answering rule cannot be found to answer a question, and the question refers to a scene in an active script, causal inferences are used to find an answerable question which can be constructed as a state or action implied by the original question. For example, in (3), the user asks a question for which there is no question answering rule. Sentence (3) is then identified as a scene

3. USER: What aircraft can the wing supply?

in $OCA, the Air Force mission script: the transferring of control of aircraft from its wing. The action referred to by this question is enabled by a state, that the wing contains the aircraft. A new "question" is constructed by substituting the script role bindings into this state: "What aircraft does the wing contain?". The script patterns required to make this inference are illustrated in Figure 1.

```
(DEF-SCRIPT-PATTERN NAME WING-SUPPLY-AIRCRAFT
    SCRIPT $OCA
    PATTERN (*ATRANS* OBJECT &OCA:AIRCRAFT
                      FROM &OCA:WING)
    ENABLED-BY WING-HAVE-AIRCRAFT
    BEFORE AIRCRAFT-FLY-TO-TARGET)

(DEF-SCRIPT-PATTERN NAME WING-HAVE-AIRCRAFT
    SCRIPT $OCA
    PATTERN (*EQUIV* CONA &OCA:AIRCRAFT
                     CONB (PART OF &OCA:WING))
    ENABLES WING-SUPPLY-AIRCRAFT)
```

Figure 1. Definitions of Script Patterns

These script patterns are used in responding to question (3). The first step is to recognize that (3) refers to the script scene WING-SUPPLY-AIRCRAFT of the $OCA script. This is accomplished by matching script patterns from the currently active script against the question. (The $OCA script was activated by (1)). Once the scene referred to by the question is identified, the causal links of the script are traversed to find a scene which enables the scene referred to by the question. In this case, the scene WING-HAVE-AIRCRAFT is the enabling scene. This scene is instantiated with the bindings of script variables found in the question. The result of this instantiation is a question conceptualization which can be answered by a question-answering production.

In some instances, more than one scene in the script enables a scene referred to by the user's question. In this case, the answer to the question can be found by intersecting the answers to questions constructed from each of the enabling scenes. In the $OCA script, this occurs in answering a question such as "What aircraft at Logan can strike the target?". The states which enable this are that the aircraft be able to reach the target and that the aircraft be suitable for the type of target. In effect, the question that is answered is "What aircraft at Logan which can reach the target from Logan are suitable for the target?". Because scripts contain the necessary inferences, the search for all enabling conditions is very efficient. In contrast, to answer this question by using a deductive database retrieval, such as that used by ACE, may require finding a large number of enabling conditions, many of which are not germane to this problem (e.g., that the aircraft have landing gear).

Another type of inference must be made to fill in missing meaning elements when the meaning of an utterance is not complete. These "conceptual completion" inferences are expressed as rules organized by the role of the meaning element which they are intended to explicate. For example, when processing the question "What aircraft at Kennedy can reach the target?", the source of the physical transfer is not explicitly specified, but must be inferred from the initial location of the object. The initial meaning representation for this question is displayed in Figure 2a. Note that the FROM role is not filled. Conceptual completion inferences are run only when required, i.e., when needed by the conceptual pattern matcher to enable a question-answering pattern, a script pattern, or even another conceptual completion pattern to match successfully. Figure 2b and Figure 2c illustrate the question production and the conceptual completion inference pattern needed to answer the above question.

The question answering production in Figure 2b would be sufficient to answer the question "What aircraft at Kennedy can reach the target from Kennedy?". If there is no filler of the FROM role of a question, it can be inferred by the conceptual completion inference in Figure 2c. This inference binds the pattern variable &OBJECT to the filler of the OBJECT role and executes the function FIND-LOCATION which can check the database for the known location of the referent of &OBJECT or, as in this example, examine the LOCATION role of &OBJECT.

B. Understanding Requests

It is also necessary to reference the domain knowledge represented in scripts while interpreting the user's commands. For example,

```
(*PTRANS*
  OBJECT (AIRCRAFT LOCATION (KENNEDY)
                   IS-A (*?*))
  TO (TARGET REF (*DEF*))
  FROM (NIL)
  MODE (*POTENTIAL*))
```

Figure 2a. The meaning representation of "What aircraft at Kennedy can reach the target".

```
(DEF-QUESTION-PRODUCTION SCRIPT $OCA
  PATTERN (*PTRANS* OBJECT &AIRCRAFT
                   FROM &SOURCE
                   TO &DESTINATION)
  Q-FOCUS (OBJECT IS-A)
  ACTION <lisp code to compute answer>
  RESPONSE <lisp code to print answer>)
```

Figure 2b. A Question Answering Production.

```
(DEF-COMPLETION-INFERENCE SCRIPT DEFAULT
    PATTERN (*PTRANS* OBJECT &OBJECT)
    INFERENCE (FROM)
    ACTION (FIND-LOCATION &OBJECT))
```

Figure 2c. Conceptual Completion Pattern.

(1) refers to two scenes of the $OCA script, and yields bindings for several script roles:

1. USER: I want to send 4 aircraft from the 707th Fighter Wing to strike Boston between 1140 and noon.

TARGET to Boston, AIRCRAFT-NUMBER to 4, WING to 707th Fighter Wing and TIME-OVER-TARGET to the time range from 1140 to 1200. This sentence, creates an instance of the $OCA script. In SNUKA, when this occurs the script role bindings are passed to KNOBS to be checked for consistency.

The process of identifying the script roles mentioned in an utterance is a method of understanding the user's indirect answers to questions. In (13), KNOBS asks the user a question which he answers in the first part of (16). This question was generated to find a value for the weapon-system Role of the script. When this question is asked, an expectation is

13. KNOBS: What weapon system should the F-16s carry?
16. USER: Hit the target with D4 ...

activated to assist in the understanding of an elided response as well as in the selection of intended word senses. (This type of expectation is discussed in further detail in section IV D.) This expectation consists of a template meaning structure derived from the script scene referred to by the question. The indirect answer is understood by noticing that the user has accomplished the binding of the script role by means other those expected.

C. Question Ordering

An important part of participating in a coherent conversation is the selection of a reasonable order for questions to be asked. The question ordering approach used by SNUKA is similar to that of DM in that there are usually several potential questions which can be asked. SNUKA differs from DM, however, in that its rules, rather than being domain specific, are generic rules operating on a knowledge structure which represents the domain. The rules used by SNUKA encode the observations of some research [GROSZ 77] [HIRST 81] that the structure of a conversation is reflected by the structure of the plans and goals of the participants. Those questions which clarify a previous utterance or establish the goals of the user are considered most important, followed by those which continue a topic brought up by the user, followed by those questions which refer to the scene which temporally follows the scene referenced by the last utterance. A question which SNUKA asks is considered to continue a topic when the user's last utterance refers to a scene of a script, some script roles are not specified, and the question asks for a value for one of these roles. For example, if initially given a request such as "Send 4 F-4C's to the target", a question which is considered to continue the topic is "From which airbase should the F-4C's fly?".

The above discussion of question ordering focused on the topic of the question to ask. In addition, the question ordering rules of SNUKA can help to choose the form of the question. For example, (12) is a request which refers to the script scene of the departing of aircraft from a target ~~The topic of weapon systems for the next question (13) is decided upon because the next script scene is the carrying of a weapon system by the aircraft. This decision also yields the meaning representation of the question to be asked.

D. Ellipsis

To interpret ellipsis, SNUKA sets up expectations when it produces a question. These differ from the expectations of GUS in that they encode the meaning of the anticipated reply, instead of the its lexical expression. This enables SNUKA to accept a wider class of ellipses. For example, in (6) the ellipsis can not be

5. KNOBS: Would you like to use F-4s for the aircraft?
6. USER: No, F-16s from the 606th Fighter Wing.

inserted into a template English sentence because additional information must replace part of the expected response. This is an example of both replacement and expansion ellipsis [Wieschedel 82]. The expectation activated when question (5) is asked is shown in Figure 3.

This expectation sets up a context for understanding several types of ellipsis. The simplest response anticipated is "Yes." or "No." and SNUKA will expand these into complete conceptualizations which could be expressed as "I

want F-4 to be used for the aircraft of mission OCA1001." or "I do not want F-4 to be used for the the aircraft of mission OCA1001.". This expansion fills a role of the concept found in the CONTEXT facet of the expectation with the concept produced by the user's actual reply. The role to be filled (in this example, it is the MODE role) is stored in the RESPONSE-FOCUS facet of the expectation. In addition, this expectation helps in forming a complete conceptualization in the case that user replies "Use F-4s.", "I want to use F-4s.", "I want F-4s to be used." or "I'd like to employ F-4s.". A wide variety of replies can be understood by the same expectation because SNUKA's expectations are based on the anticipated meaning of the reply, as opposed to a syntactic or lexical representation of this meaning. The processing of (6) requires the same type of expansion of an elided response to a full conceptualization, with the added complexity of the substitution a subconcept of the expected response with a concept from the actual response.

```
(EXPECTATION CONVERSATION-FOCUS (AIRCRAFT)
  CONTEXT (*GOAL* ACTOR (*USER*)
                  GOAL  (*USE* OBJECT (F-4)
                               FOR (&OCA:AIRCRAFT)
                               OF (OCA1001)))
  RESPONSE-FOCUS (MODE))
```

Figure 3. A Response Expectation.

SNUKA must monitor the expectations activated when it asks a question in order to deactivate those that are no longer appropriate. In the simplest case, an expectation is deactivated when it has been satisfied. The expectation can be satisfied when it is used in the expansion of an ellipsis or when the user replies with a full conceptualization which is equivalent to that which would be produced by the expansion of an ellipsis. In the case that the CONTEXT of the expectation is a GOAL, the expectation can be deactivated when the goal is fulfilled by any means. The expectation in Figure 3 is satisfied when an aircraft is selected for the mission. This could be accomplished by a command posed in English, or by a value chosen from a menu or inserted into a form.

Expectations must also be deactivated when the topic of conversation changes, i.e., the user ignores SNUKA's question. The determining of the topic of an utterance has not been earnestly approached in this work. Instead, SNUKA uses a set of simple heuristics to determine if the user has changed the topic after it asks a question. One such rule states that the topic has not changed if a concept in the user's utterance is a member of the same class as the concept expected to be the answer to a question. This rule uses the CONVERSATION-FOCUS of the expectation.

SNUKA utilizes two kinds of expectations to interpret ellipses. The expectation in Figure 3 is called a "meaning template" expectation because it can be used as a template in which the meaning of the user's reply can be inserted, as well as a source for fillers to complete the user's reply. A weaker form of expectation is a "meaning completion" expectation. This type of expectation can only be used to complete the meaning of a user's reply. During the course of a conversation, a meaning template expectation can become a meaning completion expectation. For example, when SNUKA asks question (13) it sets up

13. KNOBS: What weapon system should the F-16s carry?
14. USER: Which are acceptable?
15. KNOBS: The acceptable weapon systems are D1, D2, and D4.
16. USER: Hit the target with D4 and plan the rest.

a meaning template expectation to handle an elided reply such as "D4." or "The lightest acceptable weapon system." instead of (14). When the user replies with a question (14), this expectation is changed to a meaning completion expectation. (This is accomplished by removing the RESPONSE-FOCUS facet of the expectation). This is done because it is not reasonable for the user to enter, for example, "D4." instead of (16). However, if the user enters "Carry D4." instead of (16) it is necessary to have the CONTEXT of the expectation available for completing the meaning of this ellipsis.

E. Conversational Rules

The top level control structure of the response module of SNUKA is implemented as a set of demons which monitor the assignment of values to state variables. Rieger calls this type of control structure spontaneous computation [RIEGER 78]. These demons are rules which describe how to conduct a conversation. The process of responding to the user's utterance is initialized by setting the variable *CONCEPT* to the conceptualization produced by the parser. This can trigger any of a number of permanent or temporary demons. In the case that the *CONCEPT* is a question with a complete conceptualization, a demon is triggered which invokes the question answering program and sets the state variables *ANSWER* to the answer conceptualization of the question, and the variable *UNDERSTOOD* to the question conceptualization. The handling of ellipsis was added to SNUKA by activating temporary rules which monitor the setting of the variable *CONCEPT*, and expand an incomplete concept (i.e., one produced when parsing ellipsis) to a complete concept before the *CONCEPT* is processed for question answering or responding to requests. These temporary rules are activated when SNUKA generates a question and deactivated by other temporary rules which monitor ellipsis processing or monitor *UNDERSTOOD* for changes in topic.

In addition to instructing SNUKA on how to respond to the user's input, there are rules which allow SNUKA to take some initiative to assist the user in accomplishing his goals. One such rule instructs SNUKA to ask the user if he would like to use a particular value for a script role, in the case SNUKA asked the user to specify a value

```
(DEF-CONVERSE-RULE NAME MONITOR-ANSWER
                   VARIABLE *UNDERSTOOD*
                   TEST (AND (IS-QUESTION? $*CONCEPT*)
                             (UNIQUE? $*ANSWER*)
                             (EXPECTED-QUESTION? $*CONCEPT* $*ANSWER*
                                                 $*EXPECTED-ANSWER*)
                             (ACCEPTABLE? $*ANSWER* $*SCRIPT-ROLE* $*SCRIPT*))
                   ACTION (ASK-USE-Y/N $*ANSWER* $*SCRIPT-ROLE* $*SCRIPT*)))
```

Figure 4. A Conversation Rule

for this role, and he replied with a question whose answer is a unique acceptable value for this role. Question (5) is generated by this rule (see Figure 4.).

The rule in Figure 4 would be activated by SNUKA after producing question (2). In addition, SNUKA sets the state variables *SCRIPT* to the

2. KNOBS: Which aircraft do you want to send?
3. USER: What aircraft can the wing supply?
4. KNOBS: The 707th Fighter Wing has F-4s.
5. KNOBS: Would you like to use F-4s for the aircraft?

current script, *SCRIPT-ROLE* to the script role mentioned in question (2), and *EXPECTED-ANSWER* to a meaning template expectation similar to the one in Figure 3. (SNUKA also sets up a demon which also uses *EXPECTED-ANSWER* to interpret a possible elided response.) The demon, MONITOR-ANSWER, is executed after answering question (3), when *UNDERSTOOD* is set to the question conceptualization. In this example, the test evaluates to TRUE, and the action procedure, ASK-USE-Y/N, produces (5) and sets up a context to understand the user's reply.

V FUTURE WORK

Future extensions to SNUKA include the incorporation of a natural language generation program and the incorporation of graphical input and output devices. Some questions might best be answered by a table or graph instead of an English reply. A generalization of the conversational rule in Figure 4. could display a menu for the user to select his choice in the case that he asks a question whose answer is a group of objects. This would be an appropriate response to (14).

The most important aspect of intelligently participating in a conversation is the recognition of goals and plans of the other participant. These goals and plans must be inferred from the participant's utterances. Once, the goals and plans are known, they must be taken into consideration when generating responses. In SNUKA, we expect to make use of the user's intentions when answering his questions and when generating questions to ask the user. The following hypothetical conversation illustrates responses which must make use of the user's intentions:

101. USER: I want to strike Boston at 10:30 with 4 aircraft carrying D2.

102. KNOBS: Which aircraft do you want to send?

103. USER: What aircraft does Kennedy have?

104. KNOBS: The aircraft at Kennedy which can carry D2 are F-16's.

105. KNOBS: Would you like to send 4 F-16's from Kennedy?

If question (103) were taken literally, an appropriate response would be a listing of all the aircraft at Kennedy including those which are not appropriate for this type of mission, those which cannot carry the weapon system D2, and those which are assigned to other missions. By recognizing the user's plan a more intelligent response such as (104) can be produced. In addition, question (105) can be generated to ask the user to confirm both an aircraft and an airbase to supply the aircraft. The conversation rule illustrated in Figure 4 would not be able to ask about the airbase because it does not utilize the user's intentions.

Allen [ALLEN 82] has shown how possible plans can be inferred from an utterance. This inference process chains forward from the logical form of an utterance and backward from a set of possible goals.

In conversation about stereotypical situations scripts can provide an efficient mechanism to infer plans and goals from utterances. To be fully understood an utterance must be related to the context in which it was produced. The identification of the script scene and the script roles referenced by an utterance can help to recognize the plans and goals it was intended to serve. The conditions which a helpful, intelligent response must meet can be discovered as part of the memory process necessary to understand an utterance. In this manner, responding to an utterance can be considered an opportunistic process [McGUIRE 82]. For example,

question (103) refers to the script scene AIRCRAFT-AT-AIRBASE with the AIRBASE role identified as Kennedy. By infering that the user intends to use Kennedy for the AIRBASE, and an aircraft at Kennedy for the AIRCRAFT in his plan, a better answer can be produced for his question. The aircraft which can be used are those which are compatible with the proposed binding of the AIRBASE role in addition to the binding of other script roles established by (101). The conditions which potential bindings for the aircraft must meet are represented by the enabling state of the script.

VI CONCLUSION

SNUKA integrates a number of knowledge sources to conduct a conversation in a stereotypical domain. The most important of these knowledge structures, the script, is also the one which most limits its applicability (to conversations about stereotypical situations). However, this script based approach is most appropriate as an interface to the KNOBS system which assists a planner by instantiating a stereotypical solution to his problem. SNUKA has demonstrated a method of conducting a conversation utilizing the domain knowledge represented in scripts, knowledge which had previously been applied to story understanding.

ACKNOWLEDGMENTS

This work was supported by USAF Electronics System Division under Air Force contract F19628-82-C-0001 and monitored by the Rome Air Development Center. Special thanks are due Sharon Walter and Dr. Northrup Fowler, III. I would also like to thank Carl Engelman, Bud Frawley, Richard Brown, Frank Jernigan, and Max Bacon for their comments on this work. Captain Louis Albino provided valuable assistance during the development of SNUKA.

REFERENCES

[ALLEN 82] Allen, J., Frisch, A., Litman, D. "ARGOT: The Rochester Dialogue System", Proceedings of the National Conference on Artificial Intelligence, Pittsburg, 1982.

[BOBROW 77] Bobrow, D.G., Kaplan, R., Kay, M., Norman, D., Thompson, H.S., and Winograd T., "GUS, a Frame-driven Dialog System", Artificial Intelligence, 8(2), April 1977.

[CHARNIAK 80] Charniak, E., Riesbeck, C. and McDermott, D., Artificial Intelligence Programming. Erlbaum Press. Hillsdale, NJ, 1980.

[CULLINGFORD 78] Cullingford, R., "Script Application: Computer Understanding of Newspaper Stories", Research Report 116, Department of Computer Science, Yale University, 1978.

[CULLINGFORD 82] Cullingford, R., "ACE: An Academic Counseling Experiment",EE&CS Department TR-82-12A, University of Connecticut, 1982.

[DEJONG 79] DeJong, G., "Skimming Stories in Real Time: An Experiment in Integrated Understanding", Research Report 158, Department of Computer Science, Yale University, 1979.

[ENGELMAN 80] Engelman, C., Scarl, E., and Berg, C., "Interactive Frame Instantiation", Proceedings of the First Annual Conference on Artificial Intelligence, Stanford, 1980.

[GROSZ 77] Grosz, B., "The Representation and Use of Focus in Dialog Understanding", PhD dissertation, University of California at Berkeley, 1977.

[HIRST 81] HIRST, G., "Discourse-oriented Anaphora Resolution in Natural Language Understanding: A Review", American Journal of Computational Linguistics, 7(2), June 1981.

[LEHNERT 78a] Lehnert, W., The Process of Question Answering, Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

[LEHNERT 78b] Lehnert, W., "Representing Physical Objects in Memory", Research Report 1978, Department of Computer Science, Yale University, 1979.

[McGUIRE 81] McGuire, R., Birnbaum, L., and Flowers, M. "Opportunisitc Processing in Arguments", Proceedings of the Seventh IJCAI, Vancouver, B.C., 1981.

[PAZZANI 83] Pazzani, M., and Engelman, C., "Knowledge Based Question Answering", Proceedings of the Conference on Applied Natural Language Processing, Santa Monica, 1983.

[RIEGER 78] Rieger, C., "Spontaneous Computation in Cognitive Models", Cognitive Science, I(3), 1978.

[ROBERTS 77] Roberts, R., and Goldstein, I., "The FRL Manual, " MIT AI Lab. Memo 409, 1977.

[SCHANK 72] Schank, R., "Conceptual Dependency: A Theory of Natural Language Understanding", Cognitive Psychology, 3(4). 1972.

[SCHANK 77] Schank, R. and Abelson, H., Scripts, Plans, Goals, and Understanding, Lawrence Erlbaum Associates, Hillsdale NJ, 1977.

[STEINBERG 80] Steinberg, L., "Question Ordering in Mixed Initiative Program Specification Dialog", Proceedings of the First Annual Conference on Artificial Intelligence, Stanford, 1980.

[WEISCHEDEL 82] Weischedel, R. and Sondheimer, N., "An Improved Heuristic for Ellipsis Processing," Proceedings of the 20th Annual Meeting of the Association for Computation Linguistics, Toronto, 1982.