American Journal of Computational Linguistics Microfiche 67

A GOAL-ORIENTED MODEL OF HUMAN DIALOGUE

James A. Moore
James A. Levin
William C. Mann

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, California 90291

The research reported herein was supported by the Personnel and Training Research Programs of the Office of Naval Research, Contract N00014-75-C-0710, under ARPA Order Number 2930 from the Cybernetics Technology Office of the Advanced Research Projects Agency.

SUMMARY

Within a view of language users as problem solvers, speakers are seen as creating utterances in pursuit of their own goals. Dialogue "works" because this activity tends to serve goals of both participants. Hence, for the model of dialogue comprehension presented here, recognition of these goals of the speaker is central to the comprehension of dialogue.

We have found that dialogues are composed of structured interactions represented by collections of knowledge which describe the interrelated goals of the participants. We call these knowledge structures "Dialogue-games" (DGs). This paper describes DGs in general, a particular one (the Helping-DG) in some detail, how DGs are used by our Dialogue-game Model (DGM), and the benefits of this model.

A DG consists of three parts: the Parameters (the two roles filled by the participants, and the topic), the Parameter Specifications (a set of predicates on the Parameters), and the Components (a sequence of goals held by the participants in the course of the dialogue).

For example, in the Helping-DG, the Parameters are HELPER, HELPEE (the roles) and TASK (the topic). The Specifications are: 1) The HELPEE wants to *perform* the TASK: 2) the HELPEE wants to be *able* to do it but 3) the HELPEE is *not* able to. 4) The HELPER wants to *enable* the HELPEE to do the TASK and 5) the HELPER is *able* to provide this help. The Components specify that 1) the HELPEE wants to establish a context by describing a collection of unexceptional events (a partial performance of the TASK): 2) he also wants to describe some sort of undesirable surprise: then 3) the HELPER wants to explain the violation of expectation so that the HELPEE can avoid it and get on with the TASK.

The DGM makes use of DGs in five stages of processing: Nomination, Recognition, Instantiation, Conduct and Termination.

The DGM models each participant's knowledge, goal and attention states. A mechanism adds to the attention state, concepts "suggested" by those already in attention. When a hearer sees himself or his partner as potentially filling a role in a DG (by fulfilling one or more demands of the DG's Specifications) then that DG is brought into attention (Nominated).

DGs can be nominated by weak evidence: Recognition is the step of verifying that these DGs are plausibly consistent with the current state of the model. Those which are not are eliminated from attention.

DGs which survive the Recognition stage are Instantiated by asserting (as assumptions) all the Specifications not yet represented as holding. For example, when a person says "Do you have a match?", Instantiation, by the hearer (of the Action-seek DG) derives assertions that the speaker does not have a match and wants the hearer to give him one.

The Conduct of the DG is modeled by tracking the pursuit and fulfillment of the participants' goals as represented in the Components.

When the DGM detects that one participant no longer regards a Specification as holding, this creates an expectation of the Termination of this phase of the dialogue--there is no longer a possibility that it will serve both participants' goals.

The appendix contains a detailed hand-simulation of the DGM assimilating a segment of a dialogue.

CONTENTS

STATEMENT OF THE PROBLEM

The broadest goal of our research has been to improve the sorry state of interactive man-machine communication, including its appearance of complexity, rigidity, lack of continuity and the difficulty it poses for many people to acquire useful levels of competence. In our pursuit of this goal, we have adopted the following two assumptions:

> Assumption 1: When people communicate with machines, they do so by using their already well-developed ability to communicate with other people.
>
> Assumption 2: The effectiveness of this communication is diminished by any adaptation required of the human.

A scientific understanding of how people communicate is thus relevant to the design of man-machine communication schemes, but such knowledge is seldom used in the design process. Since human communication skills have not been characterized at a level of detail appropriate for guiding design, interface designers have not been able to take into account some major determinants of their success.

The operative goal of our research was therefore to *create a model of human communication* at an appropriate level of detail to benefit man-machine communication design. Any form of communication must be based on knowledge shared by the individuals engaged in that communication. However, the nature of this shared knowledge and how is it used in the communicative process have not been well understood. We have developed a working hypothesis which has deeply affected the research:

> Hypothesis: People know that certain kinds of goals may be pursued by communication, and they know which kinds of communication acts correspond to which goals. The use of this knowledge is essential to comprehending dialogue.

In particular, a person generates an utterance to advance one or more of his own goals. Thus, to assimilate a particular utterance, it is necessary to identify why the person said it.

Working with this hypothesis, we have conducted three related investigations:

1. A study of naturally occurring language to discover regularities of usage and to determine what these regularities mean to the users of the language.

2. The representation of these regularities as knowledge structures and processes in a dialogue model.

3. The establishment of standards by which the model's performance can be compared with that of humans on closely related tasks.

We have adopted two additional, tactical constraints on the task:

1. We have modeled only the receptive aspects of communication.

2. We have examined only dialogue communication, interaction in real-time, by exactly two people. These dialogues were conducted over a restricted medium so that there was no visual or intonational communication not captured in the transcript.

PAST RESEARCH ON LANGUAGE COMPREHENSION

Most of the research into language comprehension has focused on the comprehension of single sentences or fragments of sentences. However some research has indicated the importance of the context created by surrounding sentences on the comprehension of an individual sentence. One specific model for the form of this multi-sentential knowledge is the "story schema", organized within a story grammar (Rumelhart, 1975). This model has been supported by the results of story recalls (Rumelhart, 1975; Thorndyke, 1977). Other similar kinds of theoretical constructs for organizing multiple sentences of stories have been proposed called: "frames" (Minsky, 1975; Charniak, 1975), "scripts" (Schank & Abelson, 1975), and "commonsense algorithms" (Rieger, 1975).

To account for the conduct and comprehension of dialogues, multi-sentential knowledge units have also been proposed by linguists and sociolinguists to explain certain kinds of regularities observed in naturally occurring dialogues. These regularities have been called "rules" by Labov & Fanshel (1974) and "sequences" by Sacks, Schegloff, & Jefferson (1974).

Once these multi-sentential knowledge units are evoked, they serve as a basis for comprehending the successive inputs. This is achieved by generating expectations and by providing a framework for integrating the comprehension of an utterance with that of its predecessors. Recently, we have proposed (Levin & Moore, 1976; 1977, Mann, Moore & Levin, 1977) multi-sentential knowledge units that are specified primarily by the speaker's and hearer's goals. These goal-oriented units, which we call Dialogue-games[1], specify the kinds of language interactions in which people engage, rather than the specific content of these interactions. People use language primarily to communicate with other people to achieve their own goals. The Dialogue-game multi-sentential structures were developed to represent this knowledge about language and how it can be used to achieve goals.

---

[1] The term "Dialogue-game" was adopted by analogy from Wittgenstein's term "language game" (Wittgenstein, 1958). However, Dialogue-games represent knowledge people have about language as used to pursue goals, rather than Wittgenstein's more general notion. Although other "games" are similar, the properties of Dialogue-games are only those described here. For example, they are not necessarily competitive, consciously pursued, or zero-sum.

An important problem for researchers of language comprehension is posed by sentences with which the speaker performs what philosophers of language have called "indirect speech acts" (Searle, 1969). The direct comprehension of these sentences fails to derive the main communicative effect. For example, declarative sentences can be used to seek information ("I need to know your Social Security number."); questions can be used to convey information ("Did you know that John and Harriet got married?") or to request an action ("Could you pass the salt?"). These kinds of utterances, which have been extensively analyzed by philosophers of language (Austin, 1962; Searle, 1969, 1975; Grice, 1975), are not handled satisfactorily by any of the current theories of the direct comprehension of language. However, these indirect language usages are widespread in naturally occurring language--even two-year-old children can comprehend indirect requests for action almost as well as direct requests (Shatz, 1975).

One theory proposed to account for these indirect uses of language is based on the concept of "conversational postulates" (Grice, 1975; Gordon & Lakoff, 1971). If the direct comprehension of an utterance is implausible, then the indirect meaning is derived using these postulates. Clark & Lucy (1975) formalized and tested this model, and found that people's response times tend to support a three-stage model (deriving the literal meaning, check its plausibility and, if implausible, deriving the "intended" meaning" from conversational rules).

In general, this approach to indirect speech acts is inference-based, depending on the application of conversational rules to infer the indirect meaning from the direct meaning and the context. A different approach has been proposed by Labov & Fanshel (1974) and by Levin & Moore (1976; 1977). Multi-sentential knowledge, organizing a segment of language interaction, can form the basis for deriving the indirect effect of utterance within the segment. For example, a multi-sentential structure for an information-seeking interaction can supply the appropriate context for interpreting the subsequent utterances to seek and then supply information. The inference-based approach requires one set of conversational rules for information requests, a different set of rules for answers to these requests, and a way to tie these two rule sets together. The Dialogue-game model postulates a single knowledge structure for this kind of interaction, with cooperating processes for: (1) recognizing when this kind of interaction is proposed, (2) using this knowledge to comprehend utterances within its scope, and (3) identifying when the interaction is to be terminated.

THE SHAPE OF THE THEORY

Our theory of human language use has been strongly influenced by work in human problem solving (Newell & Simon, 1972) in which the behavior of a human is modeled as an information.processing system, having goals to pursue and selecting actions which tend to achieve these goals. We view humans as engaging in linguistic behavior in order to advance the state of certain of their goals. They decide to use language, they select (or accept) the other participant for a dialogue, they choose the details of linguistic expression -- all with the expectation that some of their desired state specifications can thereby be realized.

In this theory of language, a participant in a linguistic exchange views the other as an independent information-processing system, with separate knowledge, goals, abilities and access to the world. A speaker has a range of potential changes he can effect in his listener, a corresponding collection of linguistic actions which may result in each such change, and some notion of the consequences of performing each of these. The speaker may view the hearer as a resource for information, a potential actor, or as an object to be molded into some desired state.

A dialogue involves two speakers, who alternate as hearers. In choosing to initiate or continue the exchange, a participant attempts to satisfy his own goals; in interpreting an utterance of his partner, each participant attempts to find the way in which that utterance serves the goals of his partner. Thus a dialogue continues because the participants continue to see it as furthering their own goals. Likewise, when the dialogue no longer serves the goals of one of the participants, it is redirected to new goals or terminated.

This mechanism of joint interaction, via exchange of utterances, in pursuit of desired tates, is useful for achieving certain related pairs of participants' goals (e.g., learning/teaching, buying/selling, getting help/giving help, ...). Many of these paired sets of goals correspond to highly structured collections of knowledge, shared by the members of the language community. These collections specify such things as: 1) what characteristics an individual must have to engage in a dialogue of this sort, 2) how this dialogue is initiated, pursued and terminated, 3) what range of information can be communicated implicitly, and 4) under what circumstances the dialogue will "succeed" (serve the function for which it was initiated) and how this will be exhibited in the participants' behavior.

We have attempted to represent these collections of knowledge and the way in which they are used to facilitate the comprehension of a dialogue, in the Dialogue game Model.

THE DIALOGUE-GAME MODEL

This section describes our Dialogue-game Model at its current state of development. It starts with a brief overview of dialogue and how it is structured, then describes the dominant knowledge structures which guide the model, and finally describes a set of processes which apply these knowledge structures to text to comprehend it.

Within the model, each participant in a dialogue is simply pursuing his own goals of the moment. The two participants interact smoothly because the conventions of communication coordinate their goals and give them continuing reasons to speak and listen. These goals have a number of attributes which are not necessarily consequences of either human activity in general, or communication in particular, but which are nonetheless characteristic of human communication in the form of dialogue:

1. *Goals are cooperatively established.* Bidding and acceptance activities serve to introduce goals.

2. *Goals are mutually known.* Each party assumes or comes to know goals of the other, and each interprets the entire dialogue relative to currently known goals.

3. *Goals are configured by convention.* Sets of goals for use in dialogue (and other language use as well) are tacitly known and employed by all competent speakers of the language.

4. *Goals are bilateral.* Each dialogue participant assumes goals complementary to those of his partner.

5. *Goals are ubiquitous.* A hearer views the speaker as always having goals he is pursuing by speaking. Furthermore, the hearer recognizes and uses these goals as part of his understanding of the utterance.

An uninterrupted dialogue goes through three phases:

establishing goals,
pursuing goals,
decommitting from goals.

Typically this sequence is repeated several times over the course of a few minutes.

We have created knowledge structures to represent these conventions, and processes to apply the conventions to actual dialogues to comprehend them. Since the knowledge structures dominate all of the activity, they are described first. The assimilation of an utterance in the dialogue is represented in this model by a sequence of modifications of a "Workspace"[2] which represents the attention or awareness of the listening party. The modifications are roughly cyclic:

1. A new item of text T is brought into attention through the "Parser."[2]

2. Interpretive consequences of T are developed in the Workspace by a variety of processes.

3. An expression E appears in the Workspace which specifies the relation between T and the imputed goals of the speaker of T.

This final expression is of course a formal expression in the knowledge representation of the model. E represents the proposition (held by the hearer) that in uttering T, the speaker was performing an act in pursuit of G, a speaker's goal known to the hearer. Successful comprehension is equated with relating text to satisfaction of speaker's goals.

To make an explicit account of dialogue in this way, we now describe the knowledge structures that represent those conventions which supply the goals for the participants to pursue. In particular, we will answer the following three questions:

1. What is the knowledge we are representing within the definition of a particular Dialogue-game?

2. How is this knowledge used to model the receptive acts of dialogue participants?

---

[2] The Parser and the Workspace are parts of the process model and are described in a later section.

3. What sort of processes does it take to support this model?

What's in a Game?

A Dialogue-game consists of three parts: a set of *Parameters*, a collection of *Specifications* that apply to these Parameters throughout the conduct of the game, and a partially ordered set of *Components* characterizing the dynamic aspects of the game. For the balance of this section, we will elaborate on these three parts and exemplify these with an example of the Helping-game.

Parameters

Dialogue-games capture a certain collection of information, common across many dialogues. However, the individual participants involved and the content subject of the dialogue may vary freely over dialogues described by the same Dialogue-game. To represent this, each Dialogue-game has a set of Parameters which assume specific values for each particular dialogue.

The dialogue types we have represented so far as Dialogue-games have each required only three Parameters: the two participants involved (called "Roles"), and the subject of the dialogue (called "Topic").

Parameter Specifications

One of the major aspects distinguishing various types of dialogues is the set of goals held by the participants. Another such aspect is the set of knowledge states of the participants. We have found that each type of dialogue has a characteristic set of goal and knowledge states of the participants, vis-a-vis each other and the subject. Within the formalism of the Dialogue-game, these are called the Parameter Specifications, and are represented by a collection of predicates on the Parameters.

These Specifications are known to the participants of the dialogue, and the requirement that they be satisfied during the conduct of a game is used by the participants to signal what Dialogue-games they wish to conduct, to recognize what game is being bid, to decide how to respond to a bid, to conduct the game once the bid is accepted, and to terminate the game when appropriate. These Specifications also provide the means with which to explain the implicit, but clearly successful, communication which accompanies

any natural dialogue. Examples and discussions of these Specifications will accompany the following description of the Helping-game.

Components

While the Parameter Specifications represent those aspects of a dialogue type that remain constant throughout the course of a dialogue of that type, we have also found that certain aspects change in systematic ways. These are represented in Dialogue-games as Components. In the Dialogue-games we have developed so far, the Components are represented as a set of participants' subgoals, partially ordered in time.

Bidding and Accepting

Bidding and Acceptance are entry operations which people use to enter Dialogue-games. Bidding

1. identifies the game,
2. indicates the bidder s interest in pursuing the game,
3. identifies the Parameter configuration intended.

Bidding is performed many different ways, often very briefly. It is typically the source of a great deal of implicit communication, since a brief bid can communicate all of the Parameters and their Specifications for the Dialogue-game being bid.

Acceptance is one of the typical responses to a Bid, and leads to pursuit of the game. Acceptance exhibits

1. acknowledgment that a bid has been made,
2. recognition of the particular Dialogue-game and Parameters bid,
3. agreement to pursue the game,
4. assumption of the Acceptor's role in the Dialogue game.

Acceptance is often implicit, especially in relatively informal dialogue. It can be indicated by statements of agreement or approval, or by beginning to pursue the game (i.e., attempts to satisfy the goals). Alternatives to acceptance include rejecting, negotiating and ignoring.

Bidding and acceptance appear to be part of game entry for all of the Dialogue-games of ordinary adult dialogue. They are also involved in game termination. In the case of termination, three alternatives are possible: interruption and spontaneous termination by either goal satisfaction or unconditional goal failure.

Once a game has been bid and accepted, the two participants each pursue the subgoals specified for their role by the Components of this game. These subgoals are mutually complementary, each set facilitating the other. Furthermore, by the time the termination stage has been reached, pursuit of the Component-specified subgoals will have assured satisfaction of the higher, initial goals of the participants, for which the game was initiated in the first place.

The Helping - game, an Example

In this section, we exhibit a specific Dialogue-game: the *Helping - game*. This game is presented in an informal representation, in order to emphasize the informational content, rather than the representational power of our formalism. Later in this report we will present the formal analogue of this same game. In what follows, the bold face indicates the information contained in the representation of this particular Dialogue-game: the text in regular type is explanatory commentary.

The (annotated) Helping-game

- -

Parameters: HELPEE, HELPER, and TASK.

The HELPEE wants help from the HELPEE. The TASK is some sort of a problem, otherwise unspecified.

Parameter Specifications:

HELPEE: wants to perform TASK.

HELPEE: wants to be able to perform TASK.

HELPEE: not able to perform TASK.

HELPEE: permitted to perform TASK.

HELPEE: a person.

These Specifications not only constrain who would qualify as filling the role of HELPEE, but also provide reliable information about the HELPEE, given that this individual is believed to be engaged in the Helping-game. This prohibits someone from asking for help on a problem he did not want solved. Similarly, if one receives what he judges to be a sincere request for help to do some task, the helper normally assumes that the requester has the necessary authority to do the task, if only he knew how.

HELPER: wants to help HELPEE perform TASK.

HELPER: able to provide help.

HELPER: a person.

So, in order to be a HELPER, an individual must be willing and able to provide the needed assistance. Since this Dialogue-game represents shared knowledge, the HELPER knows these Specifications, and therefore will not bid the Helping-game to someone who is not likely to meet them. And similarly, no one who fails to meet these Specifications (and knows he fails) will accept a bid for the Helping-game with himself as HELPER.

Components of the Helping - game:

There are three components; the first two constitute the "Diagnosis" phase to communicate what the problem is.

1. HELPEE wants HELPER to know about a set of unexceptional, actual events.

The HELPEE sets up a context by describing a situation where everything, so far, is going well. Since the HELPEE assumes that the TASK is understood by the HELPER, he also assumes that the HELPER shares his expectations for subsequent activity.

2. HELPEE wants HELPER to know about:

1) a set of exceptional events which occurred

or

2) a set of expected, unexceptional events which did not occur.

This pattern of a Helping-game is sufficiently well known to the participants, that the HELPEE almost never needs to actually ask a question at this point. By simply exhibiting a failure of expectation, the HELPEE has communicated that this acts as a block to his successfully pursuing the TASK. The HELPER is expected to explain why the failure occurred and how HELPEE can avoid it or otherwise continue in the TASK.

The third component specifies the "Treatment" phase where the HELPER communicates an explanation for the perceived failure.

3. HELPER wants HELPEE to know about an action which will avoid the undesired event or cause the desired one.

The context description enables the HELPEE to identify a collection of activities which he understands, and in which the HELPEE is attempting to participate. The violation-of-expectation description points out just where the HELPEE's image of the activities differs from the correct image. It is from this area of difference that the HELPER selects an action for the HELPEE.

- -

Dialogue - games in the Comprehension of Dialogue

In this section we describe the five stages of dialogue assimilation and detail the involvement of Dialogue-games with each stage:

1) nomination,
2) recognition,
3) instantiation,
4) conduct,
5) termination.

Processing Environment

Our description of the model should be viewed as representing the changing cognitive state of *one* of the participants, throughout the course of the dialogue. That is, *two* models are involved, one for each participant. Since the same processing occurs for both, we will describe only one.

The Dialogue-Game Model consists of a Long-Term Memory (LTM), a Workspace (WS), and a set of processes that modify the contents of WS, contingent upon the contents of LTM and WS. LTM contains a representation of the knowledge that the particular dialogue participant brings to the dialogue before it starts. This includes knowledge about the world, relevant objects, processes, concepts, the cognitive state of his partner in dialogue, rules of inference and evidence, as well as linguistic knowledge (words and their semantic representation, case frames for verbs and predicates and the multi-turn language structures, the Dialogue-games).

WS is the volatile short-term memory of the model, containing all the partial and temporary results of processing. The contents of WS at any moment represent the model's state of comprehension and focus at that point. The processes are autonomous specialists, operating independently and in parallel, to modify the entities in WS (called "activations"). These processes are also influenced by the contents of WS, as well as by the knowledge in LTM. Thus, WS is the place in which these concurrently operating processes interact with each other. This anarchistic control structure resembles that the HEARSAY system (Erman, Fennel, Lesser & Reddy, 1973)

Nomination

When dialogue participants propose a new type of interaction, they do not consistently use any single word or phrase to introduce the interaction. Thus we cannot determine which Dialogue-games represent the dialogue type through a simple invocation by name or any other pre-known collection of words or phrases. Instead the dialogue type is communicated by attempts to establish various entities as the values of the Parameters of the desired Dialogue-game. Thus, an utterance which is comprehended as associating an entity (a person or a concept) with a Parameter of a Dialogue-game suggests that Dialogue-game as a possibility for initiation.

The Dialogue-Game Model has two ways in which these nominations of new Dialogue-games occur. One of the processes of the model is a "spreading activation" process called Proteus (Levin, 1976). Proteus generates new activations in WS on the basis of relations in LTM, from concepts (nodes in the semantic network) that are already in WS. Proteus brings into focus concepts somehow related to those already there. A collection of concepts in WS leads to focusing on some aspect of a particular Dialogue-game, in this sense "nominating" it as a possible new Dialogue-game.

MATCH and DEDUCE are two of the model s processes which operate in conjunction to generate new activations from existing ones, by means of finding and applying rule-like transformations. They operate through partial match and plausible inference techniques, and if they activate Parameters, then the Dialogue-game that contains those Parameters becomes nominated as a candidate Dialogue-game. Match and Deduce operate together as a kind of production system (Newell, 1973).

For example, from the input utterance:

"I tried to send a message to <person> at <computer-site> and it didn't go."

the following two sequences of associations and inferences result:

(1a) I tried to X.
(2a) I wanted to X.
(3a) I want to X.
(4a) HELPEE wants to do TASK.

(1b) It didn't go.
(2b) What I tried to do didn't work.
(3b) X didn't work.
(4b) I can't X.
(5b) I don't know ho to X.
(6b) HELPEE doesn't know how to do TASK.

(Where: I = HELPEE and X = do TASK = send a message to <person> at <computer-site>.)

At this point, (4a) and (6b), since they are both Parameter Specifications for the Helping-game, cause the model to focus on this Dialogue-game, in effect nominating it as an organizing structure for the dialogue being initiated.

Recognition

The processes described so far are reasonably unselective and may activate a number of possible Dialogue-games, some of which may be mutually incompatible or otherwise inappropriate. The Dialogue-game Manager investigates each of the nominated Dialogue-games, verifying inferences based on the Parameter Specifications, and eliminating those Dialogue-games for which one or more Specifications are contradicted.

A second mechanism (part of Proteus) identifies those activations which are incompatible and sets about accumulating evidence in support of a decision to accept one and delete the rest from the WS.

For example, suppose the question

"How do I get RUNOFF to work?"

leads to the nomination of two games:

Info-seek-game (person asking question wants to know answer)
and
Info-probe-game (person asking question wants to know if other knows answer)

These two Dialogue-games have a lot in common but differ in one crucial aspect: In the Info-seek-game, the questioner does not know the answer to the question, while in the Info-probe-game he does. These two predicates are represented in the Parameter Specifications of the two Dialogue-games, and upon their joint nomination are discovered to be contradictory. Proteus represents this discovery with a structure which has the effect of eliminating the conflicting Dialogue-game with the least supporting evidence. Such support might be, for example, either the knowledge that the speaker is the hearer's teacher or that he is a novice programmer (which would lend support for the choice of the Info-probe-game or Info-seek-game, respectively).

Through these processes, the number of candidate Dialogue-games is reduced until those remaining are compatible with each other and with the knowledge currently in WS and in LTM.

Instantiation

Once a proposed Dialogue-game has successfully survived the filtering processes described above, it is then instantiated by the Dialogue-game Manager. Those Parameter Specifications not previously known (represented in the WS) are established as newly inferred knowledge about the Parameters. A large part of the implicit communication between dialogue participants is modeled through Instantiation.

To illustrate this, suppose that the following come to be represented in WS (i.e., known) in the course of assimilating an utterance:

SPEAKER does not know how to do a TASK.
SPEAKER wants to know how to do that TASK.
SPEAKER wants to do the TASK.

These are adequate to nominate the Helping-game. In the process of instantiating this Dialogue-game, the following predicates are added to WS:

SPEAKER believes HEARER knows how to do TASK.
SPEAKER believes HEARER is able to tell him how to do TASK.
SPEAKER believes HEARER is willing to tell him how to do TASK.
SPEAKER wants HEARER to tell him how to do TASK.
SPEAKER expects HEARER to tell him how to do TASK.

The model predicts that these predicates will be implicitly communicated by an utterance which succeeds in instantiating the Helping-game. This corresponds to a dialogue in which "I can't get this thing to work" is taken to communicate that the speaker *wants* to "get this thing to work" (even though, on the surface, it is only a simple declarative of the speaker's ability).

Conduct

Once a Dialogue-game is instantiated, the Dialogue-game Manager is guided by the Components in comprehending the rest of the dialogue. These Components are goals for

the dialogue participants. For the speaker, these goals guide what he is next to say; for the hearer, these provide expectations for the functions to be served by the speaker s subsequent utterances.

These "tactical" goals are central to our theory of language: an utterance is not deemed to be comprehended until some direct consequence of it is seen as serving a goal imputed to the speaker Furthermore, although the goals of the Components are active only within the conduct of a particular game, their pursuit leads to the satisfaction of the goals described in the Parameter Specifications, which were held by the participants prior to the evocation of the Dialogue-game.

In the case of the Helping-game, the goals in the "diagnostic" phase are that the HELPEE describe a sequence of related, unexceptional events leading up to a failure of his expectations. These goals model the state of the HELPER as he assimilates this initial part of the dialogue, both in that he knows how the HELPEE is attempting to describe his problem, and also that the HELPER knows when this phase is past, and the time has come (the "treatment" phase) for him to provide the help which has been implicitly requested.

Termination

The processes described above perform the identification and pursuit of Dialogue-games. How, then, are DGs terminated? The Parameter Specifications represent those aspects of dialogues that are constant over that particular type of dialogue. The Dialogue-Game Model pushes this a step further in specifying that the Dialogue-game continues only as long as the Parameter Specifications continue to hold. Whenever any predicate in the Specification ceases to hold, then the model predicts the impending termination of this Dialogue-game.

For example, if the HELPEE no longer wants to perform the TASK (either by accomplishing it or by abandoning that goal), he indicates this with an utterance which bids for termination. The Helping game then terminates; this corresponds to the simultaneous termination of the helping interaction. If the HELPER becomes unwilling to give help, or discovers that he is unable, then the Helping-game also terminates. Again, we have one simple rule that covers a diversity of cases--a rule for termination that captures the variety of ways that the dialogues we have studied end.

The Dialogue - game Processes

In this section we describe the major process elements of the Dialogue-Game Model. All the major parts and their connectivity are shown in Figure 1. These parts (two memories and six Processes) will each be described separately. The appendix contains an extensive, detailed trace of the model as it analyzes (via hand simulation) a naturally occurring dialogue fragment. Finally, we will summarize our experience with the model to date.

Long-term Memory (LTM)

The Long-Term Memory is the model's representation of a participant's knowledge of the external world. It contains the initial knowledge states of the participants: the grammatical case frames, the semantic structures for word-senses, the knowledge of the subject matter of the dialogue, the various ways in which dialogues are structured, etc.

LTM is a semantic network, containing a set of nodes (also called concepts) and the relations that hold between them at the lowest level. This information is stored in the form of triples:

<node-1 relation node-2>

We have this machinery encoded and working--a full complement of read and write primitives for this representation. However, it has proven awkward for us to specify knowledge at this level, so we have implemented further machinery (named SIM) to translate n-ary predicates into these triples. Thus, for a predicate, P, having arguments A1, A2, and A3, SIM can be given the input:

P1: (Alpha P Beta Gamma)

[meaning that P1 is defined to be an instance of P (the predicate always goes ~~in second~~ position) with arguments Alpha for A1, Beta for A2 and Gamma for A3.] The resulting triples are created:

<P1 PRED P>
<P1 A1 ALPHA>
<P1 A2 BETA>
<P1 A3 GAMMA>

Let's examine a more concrete example; suppose we want to include in the LTM that:

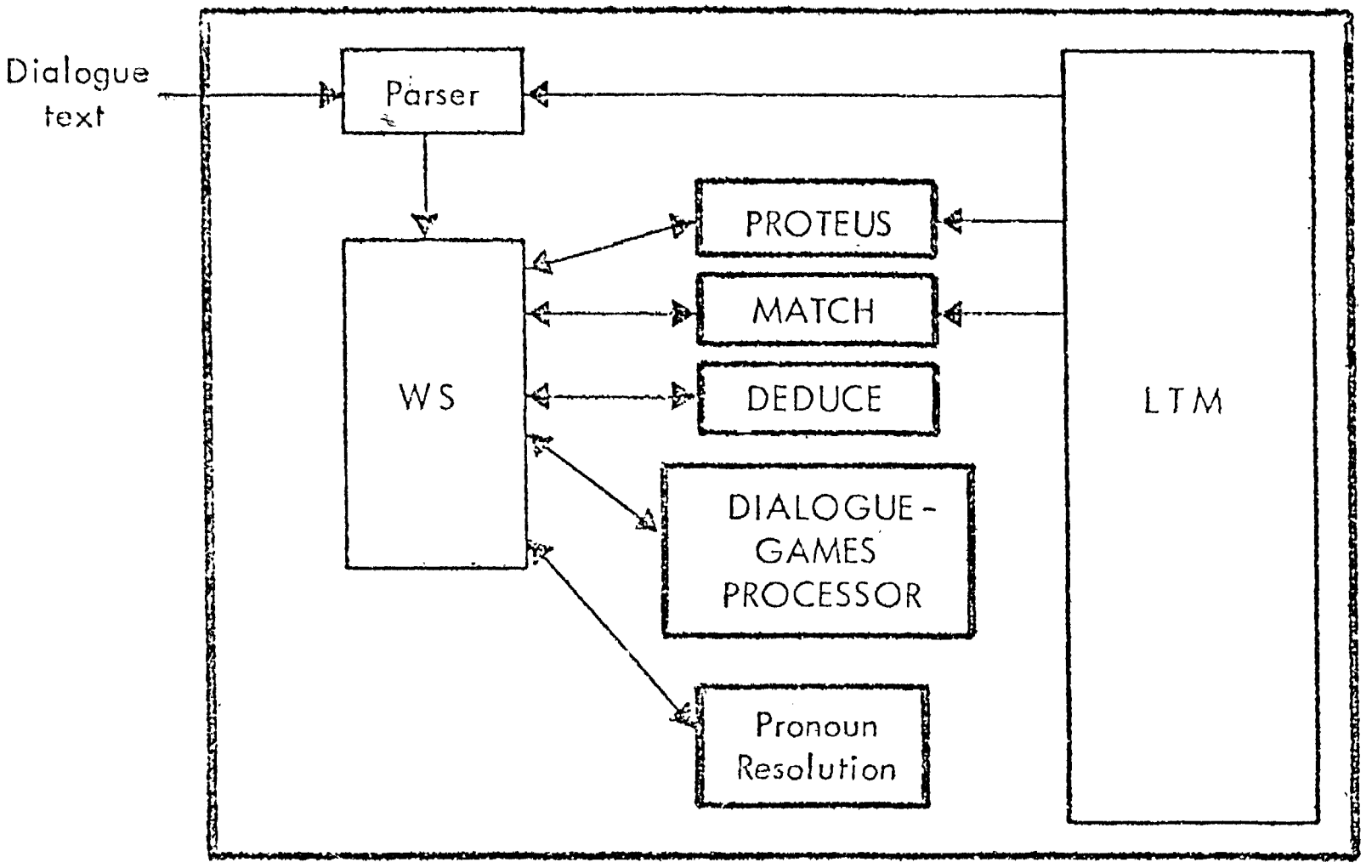

Figure 1. The Dialogue-game Model

Mary hit John with a rock.

The predicate "HIT" has two mandatory arguments (subject, object) and an optional one (instrument). The SIM representation of this assertion (which we shall name Q1) is

Q1:(MARY HIT JOHN ROCK)

which translates into the following triples:

<Q1	PRED	HIT>
<Q1	SUBJ	MARY>
<Q1	OBJ	JOHN>
<Q1	INST	ROCK>

Workspace (WS)

The Workspace is the model's representation for that information which the participant is actively using. This memory corresponds roughly to a model of the participant's focus of attention.

While the LTM is static during the operation of the model (we are not attempting to simulate learning), the WS is extremely volatile, with elements (activations) coming into and out of focus continuously. All incoming sensations (i.e., utterances) appear in the WS, as do all augmentations of the participant's knowledge and goal state. The representational format of the WS is the same as in LTM. Each node in the WS is a token (copy) of some node in LTM. Whenever some process determines that the model's attention (WS) should include a token of a specific node (C) from LTM, a new node (A) is created by copying C and this new node is added to the WS. A is referred to as an activation of C, and the relation (Is an Activation Of) is stored as

<A IAO C>

This representation provides the associative links between an object in attention, and the body of knowledge associated with it, but not yet brought into attention.

Parser

This module produces activations representing each successive utterance to be processed. These representations are generated from the surface string using a standard ATN Grammar similar to those developed by Woods (1970) and Norman, Rumelhart, & the LNR Research Group (1975). We use a case grammar representation, with each utterance specified as a main predicate with a set of parameters. Because this module is a conventional parser whose implementation is well understood, we have so far produced hand parses of the input utterances, following an ATN grammar.

Proteus

This is a spreading activation mechanism, which modifies the activation of concepts specified as related in LTM whenever a given concept becomes active. This mechanism provides a way to integrate top-down and bottom-up processing within a uniform framework (Levin, 1976). The Dialogue-Game Model uses Proteus to activate a concept, given tha a number of closely related concepts (Components, features, instances, etc.) are active.

Proteus operates on all current activations to modify their "salience", a number associated with each activation that generally represents the importance or relevance of the concept. Two kinds of influence relations can exist between concepts: *excite* or *inhibit*. If an *excite* relation exists, then Proteus increases the salience of the activation of that concept in proportion to the salience of the influencing concept. The higher the salience of an activation, the larger its influence on directly related concepts. If an *inhibit* relation is specified, then Process decreases the salience of the activation of the neighboring concept.

Match

This Process identifies concepts in LTM that are congruent to existing activations. The Dialogue-Game Model contains a number of equivalence-like relations, which Match uses to identify a concept in LTM as representing the same thing as an activation of some seemingly different concept. Once this equivalent concept is found, it is activated. Depending on how this concept is defined in LTM, its activation may have effects on other processes (for example, if the concept is part of a rule, Deduce may be invoked).

Match can be viewed as an attempt to find an activation (A) in WS and a Concept (C) in LTM which correspond, according to some set of criter The basic tactic is to attempt

to find a form of equivalence relationship between A and C, without delving into their structure at all. Only if this fails are their respective substructures examined. In this second case, the same match which was attempted at the top level is tried between corresponding subparts of A and C. Match proceeds in five steps:

1. Is it already known that A is an activation of C? If so, the match terminates with a positive conclusion.

2. Is there any other activation (A') and/or concept (C') such that A' is known to be a view of A, C is known to be a kind of C', and A' is known (by step 1) to be an activation of C'? The relations (... is a view of ...) and (... is a kind of ...) represent stored relations between pairs of activations and concepts, respectively. One concept "is a kind of" another concept represents a superclass inclusion, true for all time and contexts. (Whatever else he might be, John is a kind of human being.) On the other hand, one activation may be "a view of" another only under certain circumstances--a conditional, or tactical relationship. Under different conditions, it is appropriate to view John as a Husband, Father, Child, Help-seeker, Advice-giver, etc.

3. A list of matched pairs of activations and concepts represent correspondences found elsewhere, with which match must be consistent. (N.B.: this Match, as we will see later, may be in service of another Match called on structures containing the current A and C.) If the pair [A,C] is a matched pair, then these two have been previously found to match, so we may here conclude the same thing and Match exits.

4. On the other hand, if there is either an X or a Y such that [A,X] (or [Y,C]) is a matched pair, then replace this match with an attempt to match C and X (or A and Y).

5. Finally, if the match has neither succeeded nor failed by this point, then Match is called recursively on all corresponding subparts of A and C, pairwise. That is, e.g., if A and C have only three subparts in common (say, SUBJ, OBJ and PRED) then Match((SUBJ of A),(SUBJ of C)), Match((OBJ of A),(OBJ of C)) and Match((PRED of A),(PRED of C)) are attempted. Only if all of these subordinate matches succeed is the top-level Match said to succeed.

Clearly, for structures of significant complexity, Match may eventually call itself recursively, to an arbitrary depth. However, since each subordinate call is on a strictly smaller unit, this process must converge.

Our experience has shown us that this type of mechanism plus a collection of rewrite rules enable us to eventually map a wide variety of input parsing structures to pre-stored, abstract knowledge structures, in a way that a significant aspect of their intended meaning has been assimilated in the process.

Deduce

This operates to carry out a rule when that rule has become active. Rules are of the form (Condition)->(Action), and Deduce senses the activity of a rule and applies the rule by activating the concept for the action. Whatever correspondences were evolved in the course of creating the activation of the condition (left) half of the rule are carried over into the activation of the action (right) half. The combination of Match and Deduce gives us the capability of a production system.

The operation of Deduce is relatively simple. It is called only when a rule is active in the WS. Deduce attempts to match the left half of this rule with some other activation in the WS. (This has typically already been done by match.) Assuming this is accomplished, Deduce creates an activation of the right half of the rule, substituting in the activation for all subparts for which there are correspondences with the left half.

Dialogue-game Manager

Once a Dialogue-game has been activated (by Proteus) as possibly the communication form being bid for a dialogue, the Dialogue-game Manager uses it to guide the assimilation of successive utterances of the dialogue, through four stages:

1. establish the Parameter values and verify that no Specification is contradicted,
2. establish otherwise unsupported Specifications as assumptions,
3. establish the Components as goals of the participants,
4. detect the circumstances which indicate that the Dialogue-game is terminating and represent the consequences of this.

The first two of these phases happen in parallel. When the Manager accesses each of the Parameters, they are found either to have activations in the WS or not. If they do, the correspondences between activation and Parameter are established in the WS. This corresponds to assigning a value to the Parameter for this particular evocation of the Dialogue-game. Any Parameter that has no activation is put on a list which is

periodically checked in the hope that later activity by the Manager will lead to the creation of appropriate activations.

For each of the Specifications, a check is made to determine if it already has an activation in WS. (In most cases, the activation of some of these Specifications will have led to the activity of the Dialogue-game itself.) The Specifications having activations need no further attention.

For all remaining Specifications, activations are created substituting for the Parameters as determined above. At this stage, the Dialogue-game Manager calls Proteus to determine the stability of these new activations. Any new activation which contradicts existing activations will have its level of activity sharply reduced by Proteus. If this happens, the Dialogue-game Manager concludes that some of the necessary preconditions for the game do not hold (are in conflict with current understanding) and that this particular game should be abandoned. Otherwise, the new activations stand as new knowledge, following from the hypothesis that the chosen game is appropriate.

The Dialogue-game has now been successfully entered; the Manager sets up the third phase, creating activations of the Dialogue-game's Components, with appropriate substitutions. (By this time, any unresolved Parameters may well have activations, permitting their resolution.) This sets up all of the game-specific knowledge and goals for both participants.

Finally, the Manager detects that one of the Specifications no longer appears to hold. This signals the impending termination of the Dialogue-game. In fact, the utterance which contains this information is a bid to terminate. At this point, if the participants' initial goals are satisfied (thus contradicting the Specification which calls for the presence of those goals) the interaction ends "successfully". Otherwise, the Dialogue-game is terminated for some other reason (e.g., one participant's unwillingness or inability to continue) and would generally be regarded as a "failure". These consequences are inferred by the Manager and added to the WS. When a Dialogue-game has terminated, its salience goes to zero and it is removed from the WS.

Pronoun Processes

The Dialogue-Game Model contains a set of Pronoun Processes, including an I-Process, a You-Process, and an It-Process. Each of these is invoked whenever the associated surface word appears in an input utterance, and operates to identify some preexisting activation that can be seen as a view of the same object.

Each of these Processes search the current context, as represented by the current set of activations in the WS, using the features specified there to identify a set of possible co-referential expressions. When there is more than one possibility, the one with a higher salience is selected.

DEFICIENCIES IN CURRENT MAN-MACHINE COMMUNICATION

With the understanding we now have of the multi-sentential aspects of human communication, it is easy to see why man-machine communication appears so alien, highly restrictive, uncomprehending and awkward. This is because *major regulation and interpretation structures are missing*.

In Table 1, we compare human dialogue and typical man-machine communication with respect to some of these features. The table designates a "sender" and a "receiver" which should be identified with the person and the computer, respectively, in the man-machine communication case.

ASPECTS OF NATURAL COMMUNICATION ADDRESSED BY DIALOGUE-GAME THEORY ·	HUMAN DIALOGUE	MAN-MACHINE
SENDER'S GOALS KNOWN TO RECIPIENT	YES	NO
PARTICIPANTS CAN DECLARE THEIR GOALS	YES	NO
GOALS PERSIST OVER SEVERAL MESSAGES	YES	NO
GOALS IDENTIFIED WITH EACH MESSAGE	YES	NO
COMMUNICATION PLANS USED	YES	LITTLE
IMPLICIT COMMUNICATION TAKES PLACE	YES	LITTLE

Table 1: A comparison of man-man and man-machine communication

Conventional man-machine communication frequently gives the user a sense that the computer is operating "out of context", since he must continually respecify what is relevant to the ongoing dialogue. In human communication it is the shared awareness of each other's goal structures which permits them to retain and focus on what is relevant. Man-machine communication seems aimless and undirected because no analogous body of knowledge is being used to facilitate and interpret the communication.

The ideal interface, and the sort toward which this research is directed, would be continuously asking itself: "Why did he say that?". From answers to this, the interface would infer just what the human was expecting as a response. This would constitute a major step toward the enabling the interface to serve the actual (rather than the poorly expressed) needs of the user. Finally, such an interface would require much less adaptation on the part of the user, and so, by our original hypothesis, would significantly enhance the effectiveness of the man-machine partnership.

CONCLUSIONS

This paper has described a research effort into the modeling of human dialogue. The purpose of this research has been to uncover and describe in process models, regularities that occur in dialogue. It is hoped that the enhanced understanding of human communication which results, will facilitate the development of more natural (and thus more effective) man-machine interfaces.

The principal regularity we have discovered is a collection of knowledge and goal structures, called Dialogue-games, which seem to be crucial in understanding the structure of naturally-occurring dialogues. According to the theory we have proposed, one or more of these Dialogue-games serve as the major organizing influence on every human dialogue.

Each Dialogue-game specifies what knowledge each person must have to engage in such a dialogue, and what goals of the participants might be served by that interchange. A Dialogue-game also specifies, as a sequence of "tactical" goals, the manner in which the dialogue is conducted.

The Dialogue-game Model is a collection of cooperative processes which continuously updated a representation of each participant's attention state in a Workspace. The model recognizes when a particular Dialogue-game is being bid, accepted, pursued and terminated, and represents these states appropriately in the Workspace. A particular Dialogue-game, the Helping-game, was described in some detail. A simulation of the evocation and use of the Helping-game on a segment of natural dialogue is contained in the Appendix.

Our experience so far with the Dialogue-game Model has reinforced our hypotheses that an understanding of the goal-serving aspects of dialogue is a powerful tool in understanding the individual dialogues.

REFERENCES

Austin, J. L. *How to do things with words.* Cambridge, MA: Harvard University Press, 1962.

Bales, R. F. *Interactive process analysis.* Cambridge, MA: Addison-Wesley, 1952.

Charniak, E. Organization and inference in a frame-like system of common sense knowledge. In R. Schank & B. L. Nash-Webber (Eds.), *Theoretical issues in natural language processing.* Cambridge, MA: Bolt, Beranek and Newman, Inc., 1975.

Clark, H. H., & Lucy, P. Understanding what is meant from what is said: A study in conversationally conveyed requests. *Journal of Verbal Learning and Verbal Behavior,* 1975, *14*, 56-72.

Erman, L. D, Fennell, R. D., Lesser, V. R., & Reddy, D. R. System organizations for speech understanding. *Proceedings of the Third International Joint Conference on Artificial Intelligence.* Palo Alto, CA: Stanford University, 1973.

Gordon, D., & Lakoff, G. Conversational postulates. *Papers from the Seventh Regional Meeting,* Chicago Linguistic Society, 1971.

Grice, H. P. Logic and conversation. In P. Cole & J. L. Morgan (Eds.), *Syntax and semantics.* New York: Academic Press, 1975.

Labov, W., & Fanshel, D. *Therapeutic discourse. Psychotherapy as conversation.* Draft copy, 1974.

Levin, J. A. Proteus: An activation framework for cognitive process models. Unpublished doctoral dissertation, San Diego, CA: Univ. of Calif, San Diego, 1976.

Levin. J. A., & Moore, J. A. Dialogue-games: A process model of natural language interaction. *Proceedings of the AISB Summer Conference.* Edinburgh, Scotland, July 1976.

Levin, J. A., & Moore, J. A. Dialogue Games: Meta-communication Structures for Natural Language Interaction (ISI/RR-77-53). Marina del Rey, CA: Information Sciences Institute, 1977. Also in press, *Cognitive Sciences,* 1977, *1(4).*

Mann, W. C. Why things are so bad for the computer-naive user (ISI/RR-75-32). Marina del Rey, CA: Information Sciences Institute, 1975.

Mann, W. C., Moore, J. A., & Levin, J. A. A comprehension model for human dialogue. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.* Cambridge, MA: MIT, 1977, forthcoming.

Minsky, M. A framework for representing knowledge. In P. H. Winston (Ed.), *The psychology of computer vision.* New York: McGraw-Hill, 1975.

Newell, A. Production systems: Models of control structures. In W. G. Chase (Ed.), *Visual information processing.* New York: Academic Press, 1973.

Newell, A., & Simon, H. A. *Human problem solving.* Englewood Cliffs, NJ: Prentice-Hall, 1972.

Norman, D. A., Rumelhart, D. E. & the LNR Research Group. *Explorations in cognition.* San Francisco: W. H. Freeman, 1975.

Rieger, C. The commonsense algorithm as a basis for computer models of human memory, inference, belief and contextual language comprehension. In R. Schank & B. L. Nash-Webber (Eds.), *Theoretical issues in natural language processing.* Cambridge, MA: Bolt, Beranek and Newman, Inc., 1975.

Rumelhart, D. E. Notes on a schema for stories. In D. G. Bobrow & A. Collins (Eds.), *Representation and understanding: Studies in cognitive science.* New York: Academic Press, 1975.

Sacks, H., Schegloff, E. A., & Jefferson, G. A simplest systematics for the organization of turn-taking for conversation. *Language,* 1974, *50,* 696-735.

Schank, R. C., & Abelson, R. P. Scripts, plans and knowledge. Paper presented at the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, August 1975.

Searle, J. R. *Speech acts: An essay in the philosophy of language.* Cambridge, England: Cambridge University Press, 1969.

Searle, J. R. Indirect speech acts. In P. Cole & J. L. Morgan (Eds.), *Syntax and semantics.* New York: Academic Press, 1975.

Shatz, M. How young children respond to language: Procedures for answering. *Papers and Reports on Child Language Development*, Palo Alto, CA: Stanford University, 1975.

Thorndyke, P. W. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology*, 1977, *9*, 77-110.

Wittgenstein, L. *Philosophical investigations* (3rd ed.). New York: Macmillan, 1958.

Woods, W. A. Transition network grammars for natural language analysis. *Communications of the ACM*, 1970, *13*, 591-606.

APPENDIX -- SIMULATION OF THE DIALOGUE-GAMES MODEL

Example of the Dialogue Model in Action

In this appendix we describe an extensive simulation of the current state of the Dialogue-game Model. We make use of a particular version of the Helping-game and also explore another structure, an Execution Scene, which describes the customary events surrounding the successful execution of a particular program (Runoff).

We start by describing this more detailed version of the Helping-game, introducing names for the various aspects, to be used later. Next we show a short, naturally occurring dialogue between a computer operator and a user. Then we describe the operation of the Dialogue-game Model as it assimilates this dialogue, up to the point at which it concludes that the Helping-game is an appropriate structure through which to understand the subsequent utterances.

Once this hypothesis for the form of the dialogue has been chosen, we continue the simulation to examine how the model decides that a particular Execution Scene is appropriate for assimilating the content of the dialogue. Next, we see how this choice of scenes enhances the set of goals imputed to the speaker, thus facilitating the comprehension of what he is saying. Finally, we summarize our experience with the Dialogue-game Model so far.

A Detailed Structure for the Helping - game

What follows is the substance of the communication structure we have named the Helping-game. In the interests of clarity of presentation, the formal structures of the definition have been expressed in prose. However, the elements of the following description correspond one-to-one to those in the actual Helping-game used in the simulation.

HELPING-GAME

Parameters:
The parameters are two roles (HELPER and HELPEE) and a topic (TASK/HG).

Parameter specifications:
The HELPER and HELPEE are each a kind of person.

H1 = A goal of the HELPEE is that he perform TASK/HG.
H2 = It is not true that HELPEE is able to perform this TASK/HG.
H5 = The HELPEE wants to be able to perform the TASK/HG.
(being able to perform the task is a subgoal of performing the task)
H6 = The HELPER is able to enable the HELPEE to perform the TASK/HG.
H8 = The HELPER is willing (= is able to want to ...) to enable the HELPEE to perform the TASK/HG.
H10 = The HELPEE is permitted to perform the TASK/HG.
H11 = The HELPEE wants the HELPER to enable him to perform the TASK/HG.
(being enabled to perform the task is a subgoal of performing the task)
Game components:
HGX1 = The HELPEE knows of a particular execution scene, XS/HE.
[note: an execution scene is a flowchart-like description of the use of a particular process; more details below]
HGX2 = The HELPEE knows that his perceiving the terminal state of XS/HE would satisfy his wanting to perform TASK/HG.
HGX2C= (Thus) The HELPEE wants to perceive XS/HE in this terminal state.
(this perception is a subgoal of performing the TASK/HG)
ACTION/GOOD = an ACTION of XS/HE which was realized in the past.
HGX3 = The HELPEE knows he has perceived this ACTION/GOOD.
HGX4 = The HELPEE knows he had expected to perceive it.
HGX5 = The HELPEE knows he wants to perceive this ACTION/GOOD.
(perceiving the ACTION/GOOD is a subgoal of perceiving the [desired] terminal state of the XS/HE)
ACTION/BAD = an ACTION of XS/HE which was not realized in the past.
HGX6 = The HELPEE knows that he did not perceive ACTION/BAD.
HGX7 = The HELPEE knows that he had expected to perceive it.
HGX8 = The HELPEE wants to perceive ACTION/BAD.
(perceiving the ACTION/BAD is a subgoal to perceiving the terminal state of XS/HE.)
HGX9 = The HELPEE wants to describe what happened which was both expected and wanted, the ACTION[s]/GOOD.
(describing these ACTION[s]/GOOD is a subgoal of having the HELPER enable the HELPEE to perform the TASK/HG.)
HGX10= The HELPEE wants to describe what did not happen that he

expected, and wanted, the ACTION[s]/BAD.
(describing these ACTION[s]/BAD is a subgoal of having
the HELPER enable the HELPEE to perform the TASK/HG.)

The Dialogue to be Modeled

What follows is a transcript of a naturally occurring dialogue between a computer operator (identified as "O") and a user ("L") who has "linked" to the operator, in an attempt to solve a problem.

There has been virtually no "cleanup" of this transcript, except to remove extraneous typing that had appeared on the operator's console listing as a result of the operating system printing routine status messages. The choice of words, and even spelling, are exactly as typed by the participants. (We have segmented the text by interposing carriage-returns as we deemed appropriate.)

Dialogue OC117

LINK FROM [L], TTY 42

L: How do I get runoff to work,
I keep xeqtn it
but it just grabs my input file
and then says done
but gives me no output?
GA

O: The output comes out on the line printer

L: Throw it away
but can I get it to go to a file?
GA

O: Confirm your commands with a comma
and you'll be queried for files, etc.
GA

L: Thanx mucho
BREAK

The subsequent simulation is of the model processing the first five segments, the entire first utterance. Each utterance is ingested one at a time, by the Parser, and the assimilation proceeds until a quiescent state is reached (much more detail, below) whereupon the next segment is parsed and input for processing.

The identification of the Helping-game

How does the model know to evoke the Helping-game? To exhibit answers to this and subsequent questions, we lead the reader through a simulation of the model as it processes the beginning of dialogue OC117. We indulge in the same use of prose for formalism as above, again with the same assurances of correspondences with the actual simulation.

The simulation proceeds in cycles; in each cycle, we exhibit the operation of a single processor, performing one iteration of its function. We do not address here the issues of how the model would select which processor to call next. In fact, our design calls for these processors to be maximally autonomous and parallel in their operation, operating whenever circumstances are ripe for their function and dormant otherwise.

The format of this simulation is as follows: The cycle number is first, in the form: <segment number>--<cycle number in this segment>. Next is the name of the processor operating in this cycle. After that is a description of the nature of the processing done during that cycle. Finally, there is a list of the results for this cycle, that is, all the important changes in WS.

Initially, the description is at a very detailed level. But after a while, the operations become extremely repetitive so the description becomes less detailed, focusing only on the unique aspects of the current operation. In this example, each processor is called at least once in the processing of each segment; Match, Deduce and Proteus bear the major burden, having several invocations each per segment.

Cycle 1-1 -- Parse.

The parser reads one utterance/segment of input and translates it into the formalism for activations in the workspace. No claim is made that this translation retains all the content of the original text, only that it is adequately faithful to the level of detail we are simulating.

Results: Case/9 (= (O perceives that L asks (how do I get Runoff working?))) is activated.

Cycle 1-2 -- I-processor

Certain words (e.g. pronouns, determiners) are taken to be signals that a reference is being made to concepts introduced elsewhere. The presence of a concept in the workspace corresponding to one of these words leads to the calling of the process-specialist which attempts to resolve the implied reference. Thus, the presence of "I" in the text leads to the calling of the I-process, whose sole function is to determine the referent of the "I" and modify the stored concept to reflect this. This process judges that if L is asking a question which contains "I" as its subject, then this constitutes adequate evidence to hypothesize that "I" is being used to refer to L.

Results: O perceives that L asks (how does L get Runoff working?)

Cycle 1-3 -- Match

Match is always on the lookout for pairs of nodes, one in the WS and the other in the LTM, such that the activation (node in WS) matches the concept (node in LTM). This is taken to be evidence that the activation is also to be taken as an activation of the matched concept. It should be understood that we are examining only some of the successful matches which occurred.

Starting in this cycle, we see a pattern which recurs regularly, and which accounts for a significant piece of the action, as the model assimilates the dialogue. Match determines that a particular activation matches the left half (condition side, if part, etc.) of a production-like rule stored in LTM. This successful match leads to the identification of the correspondences between the aspects of the activation and those of the left half of the rule, as well as creating an activation of the rule itself. The activation of a rule leads to calling the Deduce processor in the next cycle, which applies the activated rule to the node in the WS responsible for the rule's activation. This application of a rule (which also results in the removal of the rule's activation from the WS) creates a new activation structure in the WS.

In other words, the introduction of a piece of knowledge suggests that a certain transformation (e.g.,"Whenever you know X, you can conclude Y.") is appropriate. This transformation is applied to the stimulus knowledge to generate a conclusion: a new piece of knowledge.

In this particular case, the above result structure is found to match the left half of

Rule0 = If O perceives a proposition,
then O knows that proposition.
with the correspondences
Case/1 (= (L asks (How do I get Runoff working?))) is activated.
corresponds to the proposition.

(This rule represents the approximation that what is perceived is accepted at face value.)

Since Case/9 is now seen to be an activation of the Left-half of Rule0, an activation for the rule itself is created in the WS.

Results: Case/9 is an activation of Left half of Rule0.
Case/1 corresponds to the proposition in Rule0.
An activation of Rule0 is entered into WS.

Cycle 1-4 -- Deduce

Since a rule is active in WS, Deduce is called in an attempt to apply the rule. The Match has guaranteed that the necessary correspondences exists between the left half of the rule and the node which is its activation. To apply the rule, Deduce creates an activation of the right half, with the corresponding sub-parts substituted.

Results: R0-1 = O knows Case/1
Activation of Rule0 deleted from WS.

Cycle 1-5 -- Match

Match finds that R0-1 matches the left half of:

Rule1 = If O knows (L asks about a proposition),
then O knows (L does not know about that proposition).

Results: R0-1 is an activation of the left half of Rule1.
Case/1 corresponds to (L asks about a proposition)
Case/2 = (How does L get Runoff working) corresponds to the proposition.
An activation of Rule1 is created in the WS.

Cycle 1-6 -- Deduce

Deduce applies Rule1 to R0-1, substituting according to the discovered correspondences.

Results: R1-1 (= O knows (L does not know Case/2), is activated.)
Activation of Rule 1 deleted from WS.

Cycle 1-7 -- Match

Match R1-1 with left half of

Rule3 = If O knows that a person does not know how to perform a task,
then O knows that that person is not able to perform the task.

Results: R1-1 is an activation of the left half of Rule3.
L corresponds to the person mentioned.
Get corresponds to Perform.
The state of Runoff working corresponds to the task.
An activation of Rule3 is created in the WS.

Cycle 1-8 -- Deduce

Deduce applies Rule3 to R1-1.

Results: R3-1 (= O knows that R3-11 = (L is not able to perform (getting Runoff working)) is activated).
Activation of Rule 3 deleted from WS.

Cycle 1-9 -- Match

Match R3-11 with H2 = Helpee is not able to perform the task.

Results: R3-11 is an activation of H2.
(getting Runoff working) corresponds to the task.
L corresponds to the Helpee

Cycle 1-10 -- Match

Match R0-1 with left 1/2 of:

Rule2 = if O knows (L asks about a proposition),
then O knows (L wants to know about that proposition).

Results: R0-1 is an activation of the left half of Rule2.
Case/1 corresponds to (L asks ...), in Rule 2.
Case/2 corresponds to the proposition.
An activation of Rule 2 is created in the WS.

Cycle 1-11 -- Deduce

Deduce applies Rule2 to R0-1.

Results: R2-1 (= O knows (L wants to know about Case/2) is activated).
Activation of Rule 2 deleted from WS.

Cycle 1-12 -- Match

Match R2-1 with left half of

Rule4 = If O knows (a person wants to know how
to perform a task),
then O knows (that person wants to perform that task).

Results: R2-1 is an activation if the left half of Rule4.
L corresponds to the person.
(getting Runoff to work) corresponds to the task.
An activation of Rule 4 is created in the WS.

Cycle 1-13 -- Deduce

Deduce applies Rule4 to R2-1.

Results: R4-1 (= O knows (L wants to perform (getting Runoff working)) is activated).
Activation of Rule 4 deleted from WS.

Cycle 1-14 -- Match

Match R4-11 with H1 = Helpee wants to perform a task.

Results: R4-11 is an activation of H2.
L corresponds to the Helpee.
(Getting Runoff working) corresponds to the task.

Cycle 1-15 -- Match

Match R0-1 with left half of

RuleVa = If O knows (a person says
(he executes a process with an instrument)),
then O knows (that person is saying
(he performs (the execution of the process)
with the instrument).

Results: R0-1 is an activation of the left half of RuleVa.
L corresponds to the person.
(getting Runoff working) corresponds to (... executes a process ...)
How corresponds to the instrument (i.e., the means).
An activation of Rule Va is created in the WS.

Cycle 1-16 -- Deduce

Deduce applies RuleVa to R0-1.

Results: RVa-1 (= O knows (L asks (how do I perform (getting Runoff working)?)) is activated)..
Activation of Rule Va deleted from WS.

Cycle 1-17 -- Match

Match RVa-1 with Left half of

Rule2a = If O knows (a person asks how to perform a task),
then O knows (that person wants O to enable him
to perform that task).

Results: RVa-1 is an activation of the left half of Rule2a.
L corresponds to that person.
(L getting Runoff to work) corresponds to the task.
An activation of Rule 2a is created in the WS.

Cycle 1-18 -- Deduce

Deduce applies Rule2a to RVa-1

Results: R2-1 (= O knows (L wants O to enable him (L) to get Runoff working) is activated).
Activation of Rule 2a deleted from WS.

Cycle 1-19 -- Match

Match R2a-1 with H11 = Helpee wants Helper to enable him to to a task.

Results: O corresponds to Helper.
L corresponds to Helpee.
(L getting Runoff to work) corresponds to the task.

Cycle 1-20 -- Proteus

H1, H2 & H11 provide Proteus with enough evidence to create an activation of the Helping-Game.

Results: An activation of the Helping-game is created in the WS.

Cycle 1-21 -- Dialogue-game Manager

The presence of an activation of a Dialogue-game in the WS leads to the calling of the processor specialized in this category of knowledge. The Dialogue-game Manager (DGM) makes use of a set of correspondences that have already been established by the matches which led to the activations of H1, H2, and H11:

Previous Results: L corresponds to Helpee
O corresponds to Helper
Case/3 (= (Runoff working)) corresponds to the task.

Once an activation of a game has led to the calling of the DGM, the Manager accesses the entire collection of information about the game from the LTM representation of it. The items of knowledge in the game, with the particular parameters of this situation substituted appropriately, fall into one of three categories:

1. Already known to hearer (e.g. H1, H2 & H11). Items in this category are simply ignored, since it serves no purpose to re-assert them.

2. Contradict knowledge already held by the hearer (e.g., if O already knew, for sure, that L knew all about Runoff). If any item falls into this category, the hypothesis that this game is active is simply abandoned as inaccurate.

3. Items neither previously known or contradicted (the majority of the content of the typical case). In this case, the DGM creates activations of these items to represent the collection of implicit knowledge that follows from a recognition of the proposed game.

Results: Activations are created for all of the following:
H5 = L wants to be able to get (Runoff working) himself.
(being able to get (Runoff working) is a subgoal
to performing (Runoff working).)
H6 = O is able to enable L to get (Runoff working).
H8 = O is able to want to enable [i.e. is willing to enable]
L to get (Runoff working).
H10=L is permitted to get (Runoff working).

The game also contains a collection of knowledge having to do with the conduct of the game, rather than what the participants need to successfully evoke it. These items of knowledge and goals are also established as activations by the DGM at this time:

Results: Activations are created or all of the following:
HGX1 = L knows of an execution scene (XS/HE).
HGX2 = L knows that if he perceives a particular
terminal state of this scene, this will
satisfy his wanting to perform the task.
HGX2C= (Thus) L wants to perceive this terminal state

of XS/HE.
An ACTION/GOOD is an ACTION within the specification of XS/HE which occurred in the past.
HGX3 = L knows that he has perceive the ACTION/GOOD.
HGX4 = L knows he expected to perceive it.
HGX5 = L wanted to perceive it.
An ACTION/BAD is an ACTION within the specification of XS/HE which has not occurred in the past.
HGX6 = L knows he has not perceived the ACTION/BAD.
HGX7 = L knows he expected to perceive it.
HGX8 = L knows he wanted to perceive it.
(perceiving the ACTION/BAD is a subgoal to perceiving the desired terminal state of XS/HE.)
HGX9 = L wants to describe the ACTION[s]/GOOD [to O].
(this describing is a subgoal to (O enables L to perform the task)
HGX10= L wants to describe the ACTION[s]/BAD [to O].
(this describing is a subgoal to (O enables L to perform the task)

Processes, procedures, ceremonies, and the like, may have an associated execution scene, which is in effect an abstract description of a complete performance of the object described. The execution scene resembles a flowchart, with the boxes being actions of one of the active agents involved.

In this case, the execution scene is for Runoff, a program which reads a file specified by the user, formats the contents of the file, and outputs this formated material onto either the line printer or another file. The execution scene of Runoff, as stored in our model, is similar to figure A-1.

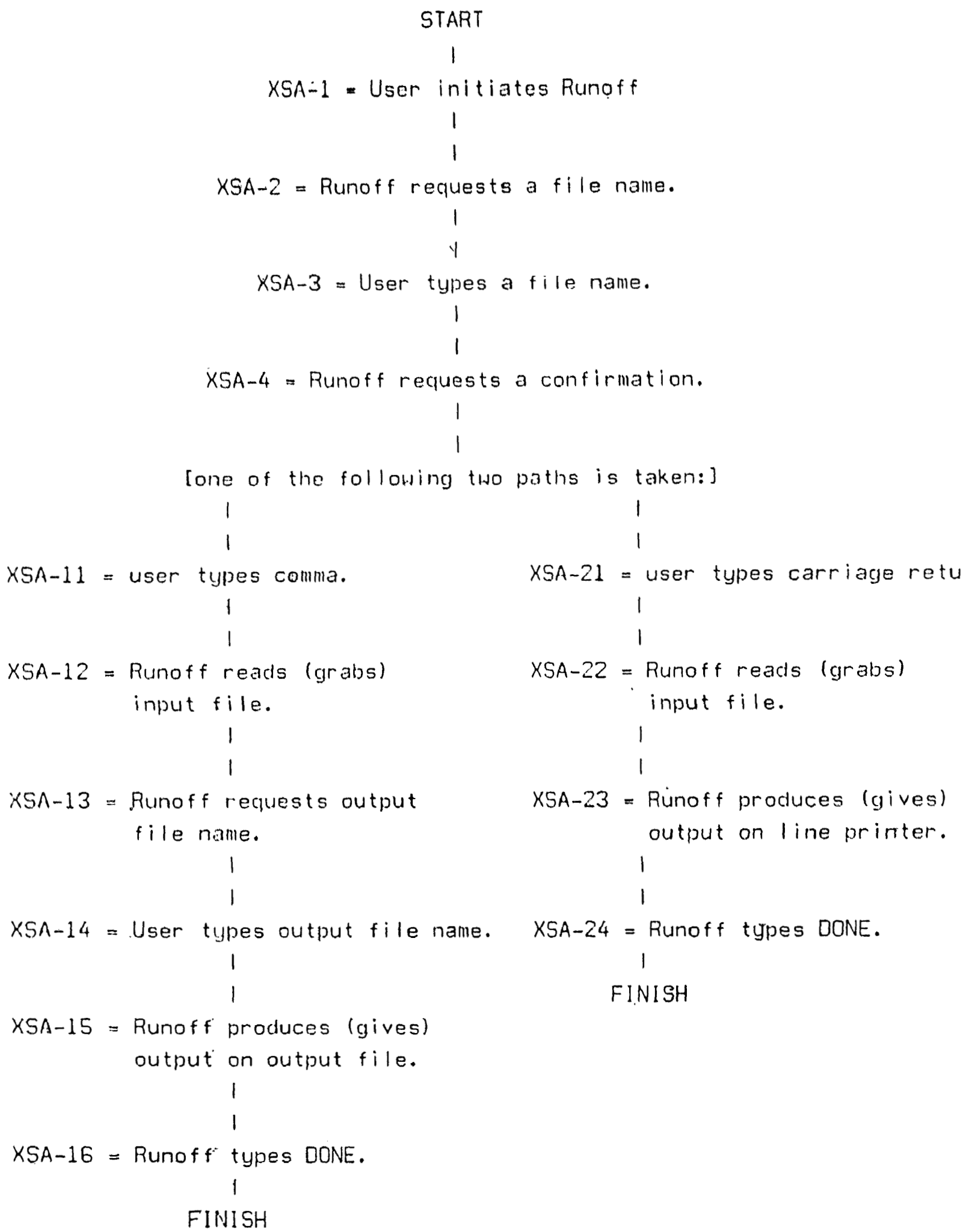

Figure A-1. XS/RO, THE RUNOFF EXECUTION SCENE.

Cycle 1-22 -- Proteus

As a result of the numerous references to Runoff and XS/HE, the activations for these two concepts are "highly active". Consequently, when Proteus is called, the concept XS/RO (the execution scene of the Runoff process) becomes active and, due to its similarity to XS/HE, is taken to be equivalent to it. Since XS/RO is more detailed (contains more information) than XS/HE, XS/RO is used in place if XS/HE in all of the expressions introduced in Cycle 1-21.

Something we passed over in the earlier examples was the issue of when the model is willing to stop processing a given piece of text and go on to the next one. It seems inappropriate to demand that the model wring all possible information and deductions out of each utterance. Yet there must be some demands made on the assimilation. An alternate form of the question is: what needs of his own does the hearer see the incoming text as potentially satisfying? We have taken the position that a hearer (tentatively) understands an utterance, when he successfully views it as serving some goal imputed to the speaker. That is, to a first approximation, the hearer has assimilated an utterance if he figures out why the speaker said it.

The model has already established (HGX9 and HGX10, above) that L wants to describe (implicitly, to O) certain actions in XS/RO that L expected to perceive, and in some cases, did. Thus, in the following utterances, we see the model matching the parsed input structure with one of these two goals, thus it is seen as being in service of a goal of the speaker, and need be examined no further (for the time being).

In the subsequent example, we use two new rules: RS (Satisfaction) and RQ (Quiescence). RS determines when an utterance is seen to satisfy a speaker's goal and RQ reacts to this detected satisfaction by marking the utterance quiescent. (Operationally, this means that in the next cycle, the Parser is called to input the next segment of text.)

We resume the example at the point where the first segment has been marked quiescent, and the Parser is called.

Cycle 2-1 -- Parser

Results: Case9a = O perceives that L declares (I executed it).

Cycle 2-2 -- I-processor

Results: Case9a = 0 perceives that L declares (L executed it).

Cycle 2-3 -- It-processor

The case frame associated with the concept "execute" specifies that the object concept is to be a process. The It-processor determines this and examines the WS to see if it contains any active concept which is a process. In this case, there is only one: Runoff. Since this case is so clearly unambiguous, this simple-minded resolution scheme is adequate to the task. (We have outlines for more ambitious resolution schemes, but the dialogues we have examined have not yet required them.)

Results: Case9a = 0 perceives that L declares (L executed Runoff).

Cycles 2-4 & 2-5 -- Match and Deduce

As in cycles 1-3 and 1-4, Rule0 is used to transform "perceive" into "know".

Results: R0-1a = 0 knows that L declares (L executed Runoff).

Cycle 2-6 -- Match

Two items in the WS are matched to the two parts of the left half of RS:

RS = If a person knows a proposition
and
he knows that a second person wants that proposition,
then the first person knows that the realization of the
proposition satisfies the second person's desire for it.

Results: R0-1a = (0 knows (L declares ...)) corresponds to
(a person knows a proposition)
0 corresponds to the first person.
(L declares ...) corresponds to the proposition.
0 knows HGX9 = (L want (L describe action/good))
corresponds to
(he knows the second person wants that proposition).
L corresponds to the second person.
(L describe action/good) corresponds to
the proposition.

(L declares (L executed Runoff)) corresponds to
(L describe action/good)
declare corresponds to describe
(L executed Runoff) corresponds to ((User initiate Runoff) past)
thus, (L executed Runoff) corresponds to action/good
An activation of RuleS is created in the WS.

Cycle 2-7 -- Deduce

Deduce applies RS to RO-1a and HGX9. Activation of Rule S deleted from WS.

Results: RS-1a (= O knows ((L declares ...) satisfies (L wants (L describe ...))) is activated).

Cycle 2-8 -- Match

Match RS-1a with left half of RQ.

RQ = If a person knows ((person2 utters something) satisfies
(person2 wants something else))
then the first person knows that he comprehends
(person2 uttering something) as constituting the
something else that person2 wanted.

Results: RS-1a corresponds to the left half of RQ.
O corresponds to the first person.
(person2 utters something) corresponds to
(L declares (L executed Runoff))
L corresponds to person 2
(L executed Runoff) corresponds to something.
(person2 wants something else) corresponds to
(L wants (L describe ...))
(L describe action/good) corresponds to something else.
An activation of RQ is created in the WS.
An activation of RQ is created in the WS.

Cycle 2-9 -- Deduce

Deduce applies RQ to RS-1a.

Results: RQ-1a = O knows (O comprehends
(L declare (L execute Runoff))
as constituting
(L describe action/good))
Activation of Rule Q deleted from WS.

Cycles 3-1 to 3-8

This set of cycles are exactly parallel to the preceding set. The structure implanted into WS by the Parser is

Case/9b (= O perceives (L declares (it grabbed file/mine)))

The It-processor translates "it" to "Runoff". Rule0 is used by Match and Deduce to replace "perceive" with "know". Match and Deduce then apply RS and RQ, to determine that Case/9b is comprehended as constituting another instance of (L describes action/good) [XSA-12 or XSA-22, Runoff reads (grabs) input file]

Cycles 4-1 to 4-8

Similarly, the Parser-produced structure:

Case/9c (= it said done)
is also found to be comprehended as constituting an instance of (L describes action/good) [XSA-16 or XSA-24, Runoff types DONE].

Cycles 5-1 to 5-10

A nearly identical sequence of cycles applies to the next Parser-input:

Case/9d (= O perceive L declare (It did not produce output),)
except an additional Match/Deduce cycle is needed to apply Rp:

Rp = If a person declares that something didn't happen,
then he is declaring he did not perceive it happen.

In this case, however, we determine that Case/9d is comprehended as constituting an instance of (L wants (L describe action/bad)) [XSA-15 = Runoff produces output on output file -- or -- XSA-23 = Runoff produces output on line printer].

What we have seen, then, is the setting up of the expectations that the speaker will (i.e. wants to) describe some things that went right, and some that didn't. The presence of these expectations has enabled the assimilation of the last four utterances, leading to the model's awareness that for L, steps XSA-1, XSA-12 or -22, and XSA-16 or -24 all proceeded as expected, but that L didn't perceive Runoff producing any output. Mechanisms outside the scope of this example determine that XSA-15 (Runoff produces output on output file) was perceivable to L (had it occurred), but that XSA-23 (Runoff produces output on the line printer) was not. This leads to the conclusion that XSA-23 probably was what had occurred, and thus to the subsequent explanation from O.