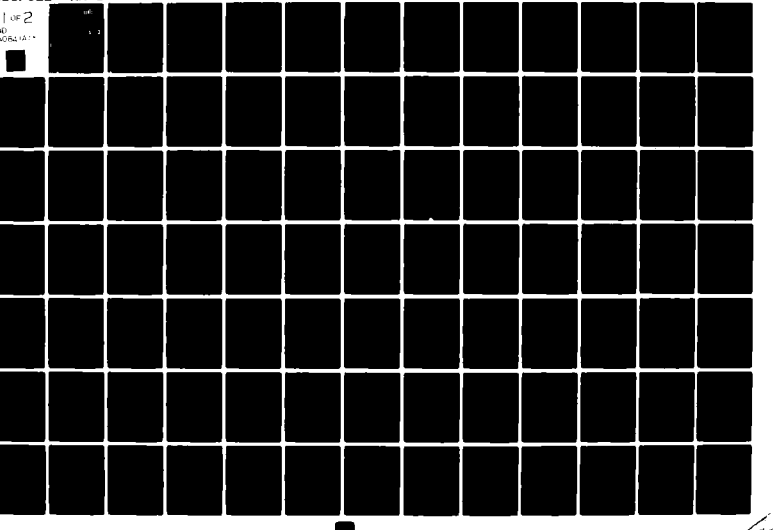
AD-A084 141 YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE F/G 5/7
MEMORY ORGANIZATION AND SEARCH PROCESSES FOR NARRATIVES.(U)
APR 80 M G DYER, W G LEHNERT N00014-75-C-1111
UNCLASSIFIED RR-175 NL



AD A084141

(6)

LEVEL II

DTIC
ELECTE
MAY 13 1980
S D
E

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DDC FILE COPY

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

80 5 12 121

Accession For
NTIS GRA&I
DDC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or special
A

MEMORY ORGANIZATION AND SEARCH PROCESSES FOR NARRATIVES

April 1980

Research Report #175

by
Michael G. Dyer and Wendy G. Lehnert

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

MEMORY ORGANIZATION AND SEARCH PROCESSES FOR NARRATIVES

April 1980

Research Report #175

by

Michael G. Dyer

and

Wendy G. Lehnert

This work is supported in part by the Advanced Research Projects Agency under contract N00014-75-C-1111 and:

National Science Foundation under contract IST7918463.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: #175
2. GOVT ACCESSION NO.: AD-A084141
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): (6) Memory Organization and Search Processes for Narratives
5. TYPE OF REPORT & PERIOD COVERED: Technical Report
6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): (10) Michael G. Dyer and Wendy G. Lehnert
8. CONTRACT OR GRANT NUMBER(s): (15) N00014-75-C-1111, NSF-IST 79-18463

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Yale University - Department of Computer Science
P.O. Box 2158 - Yale Station
New Haven, Connecticut 06520

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:

11. CONTROLLING OFFICE NAME AND ADDRESS:
Advanced Research Projects Agency
Arlington, Virginia 22209

12. REPORT DATE: (11) April 1980
13. NUMBER OF PAGES:

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
Office of Naval Research
Information Systems Program
Arlington, Virginia 22217
(12) 101

15. SECURITY CLASS. (of this report): unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report):
Distribution of this report is unlimited

(9) Research rept.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

(14) RR-175

18. SUPPLEMENTARY NOTES:

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Natural Language Processing
Question Answering
Memory Representation
Information Retrieval
Knowledge Structures
Human Information Processing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
BORIS represents the first system to integrate the knowledge-based inference techniques of scripts, plans, goals, and themes, within a single narrative-understanding program. This report discusses techniques used by BORIS for memory representation memory retrieval. Emphasis is placed on human question answering abilities, and the heuristics needed to simulate these phenomena. An example narrative processed by BORIS is discussed in detail and used to illustrate design decisions.

DD FORM 1 JAN 73 1473

407051

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

-- OFFICIAL DISTIRUBTION LIST --

Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12 copies
Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217	2 copies
Dr. Judith Daly Advanced Research Projects Agency Cybernetics Technology Office 1400 Wilson Boulevard Arlington, Virginia 22209	3 copies
Office of Naval Research Branch Office - Boston 495 Summer Street Boston, Massachusetts 02210	1 copy
Office of Naval Research Branch Office - Chicago 536 South Clark Street Chicago, Illinois 60615	1 copy
Office of Naval Research Branch Office - Pasadena 1030 East Green Street Pasadena, California 91106	1 copy
Mr. Steven Wong New York Area Office 715 Broadway - 5th Floor New York, New York 10003	1 copy
Naval Research Laboratory Technical Information Division Code 2627 Washington, D.C. 20375	6 copies
Dr. A.L. Slafkosky Commandant of the Marine Corps Code RD-1 Washington, D.C. 20380	1 copy
Office of Naval Research Code 455 Arlington, Virginia 22217	1 copy

Office of Naval Research — 1 copy
Code 458
Arlington, Virginia 22217

Naval Electronics Laboratory Center — 1 copy
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E.H. Gleissner — 1 copy
Naval Ship Research and Development
Computation and Mathematics Department
Bethesda, Maryland 20084

Captain Grace M. Hopper — 1 copy
NAICOM/MIS Planning Board
Office of the Chief of Naval Operations
Washington, D.C. 20350

Dr. Robert Engelmore — 2 copies
Advanced Research Project Agency
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

Professor Omar Wing — 1 copy
Columbia University in the City of New York
Department of Electrical Engineering and
Computer Science
New York, New York 10027

Office of Naval Research — 1 copy
Assistant Chief for Technology
Code 200
Arlington, Virginia 22217

Major J.P. Pennell — 1 copy
Headquarters, Marine Corp.
(Attn: Code CCA-40)
Washington, D.C. 20380

Computer Systems Management, Inc. — 5 copies
1300 Wilson Boulevard, Suite 102
Arlington, Virginia 22209

Ms. Robin Dillard — 1 copy
Naval Ocean Systems Center
C2 Information Processing Branch (Code 8242)
271 Catalina Boulevard
San Diego, California 92152

Table of Contents

Memory Organization and Search Processes for Narratives

Michael G. Dyer and Wendy G. Lehnert

Yale University
Computer Science Dept.
New Haven, CT 06420

April 1980

BORIS represents the first system to integrate the knowledge-based inference techniques of scripts, plans, goals, and themes, within a single narrative understanding program. This report discusses techniques used by BORIS for memory representation and memory retrieval. Emphasis is placed on human question-answering abilities, and the heuristics needed to simulate these phenomena. An example narrative processed by BORIS is discussed in detail and used to illustrate design decisions.

1. INTRODUCTION

BORIS (a Better Organized Reading and Inference System) is a the language processing system under development at the Yale Artificial Intelligence Project. BORIS is designed to read narrative texts and answer questions about its input. In this report we will present techniques of memory organization and retrieval that have been implemented in BORIS.

1.1. BACKGROUND

Previous natural language understanding projects at Yale have been restricted in the sense that each has dealt with the application of a single knowledge structure. SAM [Cullingford 78] and FRUMP [DeJong 79] were designed to understand script-based stories, while PAM [Wilensky 78] handled plan-based stories. Although [Schank and Abelson 77] has described very generally how these separate knowledge structures might interact and combine, BORIS signifies the first attempt to actually construct a system which would be able to access multiple knowledge structures in order to understand stories of arbitrary complexity.

There are three basic issues we address in the BORIS project: (1) how narrative memory structures are constructed at the time a story is understood, (2) how the internal organization of narrative memory can be specified, and (3) how these memory structures can be accessed and searched for the purposes of memory retrieval. Within each of these general problem areas there are a number of specific concerns:

- How is each sentence in the text to be represented in memory?

- How do various knowledge structures aid in the analysis of words and phrases?
- How are sentence representations integrated into the larger memory structure for the story?
- What does narrative memory look like in terms of its organization and high level structure?
- How are questions understood within the context of a story?
- What are the processes which search memory, and what operations do they perform?

In order to address these various issues, we have adopted the following research methodology:

1) Select a story containing multiple scripts, plans, goals, affective reactions, relationships, settings, themes, expectation violations, and so forth.

2) Collect some question answering data from subjects who have read the story.

3) Design a computer program which will produce the same input/output behavior of our human subjects.

4) Collect more question answering data and protocol data when design decisions can be resolved by additional data.

The first prototype story we chose for BORIS appears below:

The Kidnap Story

John left his office early Friday afternoon so he could attend a going-away party for his friend Bill. Bill had been transferred to California. When he arrived there were lots of people and plenty of beer. Three years ago Bill had borrowed $50 from John which John hoped to get back before Bill left. He was about to remind Bill about it when he was called to the phone. It was his wife, Mary, and she was hysterical. Mary had just received a call from a stranger who claimed to have kidnapped their daughter Susan. He said he would call again. John told Mary to stay at home, keep off the phone, and wait until he got there. John drove home immediately. Just as he walked in the door, the phone rang. John picked it up. But it was only Karen, one of Mary's friends. John told her Mary was out and hung up. Mary wanted to call the police but John thought they should wait. They were discussing this when Susan walked in the door. She had stayed after school for a gymnastics meet and taken a late bus home. Nothing had happened to her. It was all a hoax.

Some representative BORIS I/O appears below. The questions are verbatim input to BORIS and the answers are verbatim computer output.

SAMPLE Q/A

Q1: Why did John go to the party?
A1: He wanted to have a good time and he was Bill's friend and he wanted to get 50 dollars back from him.

Q2: Why did John leave the party?
A2: He was worried about Susan.

Q3: Why?
A3: Susan was kidnapped.

Q4: What happened to John at the party?
A4: He got a call from Mary.

Q5: Why did Mary call John at the party?
A5: She wanted to tell him about Susan's kidnapping.

Q6: Why did John want money from Bill?
A6: He borrowed 50 dollars from him.

Q7: Why didn't John talk to Bill?

A7: He was more worried about Susan.

Q8: Who called John at home?
A8: Karen.

Q9: Why?
A9: She wanted to talk to Mary.

Q10: Was Karen upset?
A10: No, She did not know about Susan's kidnapping.

Q11: Did John kidnap Susan?
A11: No. He was her father.

Q12: Where was Susan?
A12: She was attending a gymnastics meet.

Q13: What happened when Susan came home?
A13: John was relieved that she was safe.

1.2. MEMORY AND RETRIEVAL

A few general observations about the processing of BORIS can be made on the basis of this sample question answering behavior. We will highlight some of these issues briefly before we proceed with a more detailed discussion of BORIS and problems in question answering.

1. Scenarios Organize Memory

Questions Q1 ("Why did John go to the party?") and Q2 ("Why did John leave the party?") refer to the movement of a character to or from a scenario. Scenario transitions are significant because they allow us to group chains of events within the structure of a scenario. Questions that ostensibly refer to the time of an event are often handled in terms of the scenario which contains that event:

Q: When did John talk to Mary on the phone?
A: At the party.

In fact, people are notoriously bad at relating events in terms of their

temporal relationships. It does not seem that specific "time nodes" are associated with events. Instead, most temporal relationships can be inferred from scenario changes. For example, if a character X moves from scenario S1 to Scenario S2, then it follows that any event involving X which occurred within S1, occurred **before** any event involving X which occurred within S2. This is why temporal questions can produce spacial answers.

In a similar manner, questions which ostensibly request a setting can be answered by referring to the main event within that setting. For instance, Q12 asks about Susan's location and is answered by recalling that Susan was attending a gymnastics meet.

2. Context Alters Understanding and Search

In both questions Q3 and Q9, ("Why?") the previous context must be available in order to understand them. In the case of Q3, the previous answer ("He was worried about Susan.") is sufficient. But in the second case, the answer to Q8 ("Karen") is not sufficient. To understand Q9 we must go back to the previous question as well.

Context influences the interpretation of questions in many ways. For example, potentially ambiguous references can be resolved by the context of a question. If we ask "What did John do after the call?" after Q5, subjects will respond, "He went home." But if we ask the identical question after Q9, subjects will say, "John and Mary talked about calling the police," or "He waited for the kidnapper to call." The reference to "the call" is resolved in both cases by references in the immediately preceding questions. If we've been talking about a

particular phone call in one question, we expect subsequent references to be consistent with that topic.

3. Memory Must Be Accessible

In a story with many events, retrieval heuristics must be especially efficient. If we must examine every event in the story everytime a question is asked, we are not accessing memory in a very efficient manner. Many things happened to John through the story, but relatively few occur within each scenario. If we can narrow our search down to a given scenario, we are facilitating the search process. This technique is most obviously appropriate when a question like Q4 ("What happened to John at the party?") explicitly specifies a scenario.

In a story with many characters, we must be able to connect events and characters by means of their intentions reactions to events. In order to answer Q5 ("Why did Mary call John at the party?") and Q6 ("Why did John want money from Bill?"), we must know how these characters are related to one another and how an event affecting one will affect the other. In addition, we must decide how the information available in the question helps direct search. For example, there does not appear to be a direct link from the concept of money to events in our story. If we ask people:

Q: What involved money in the story?

they tend to hesitate and finally answer:

A: The ransom.

Why then, don't people consider this in order to answer Q6 as follows:

Q: Why did John want money from Bill?
A: To help pay the ransom.

Evidently, people know that John's goals involving Bill are separate from his problems with the kidnapping. In particular, this answer is inappropriate because John had a goal of getting money from Bill before he had any goals concerning the kidnapping.

Counterfactual questions such as Q7 ("Why didn't John talk to Bill?") pose special problems in a complicated story. How is an event located in memory when it never occurred in the story in the first place? To handle such questions, the intended activities of each character have to be inferred and represented even if that intention is never realized.

In Q8 ("Who called John at home?"), the appropriate event in memory is recalled, even though Karen was actually calling Mary. Search processes must be able to access events in memory even if they do not perfectly correspond to the description of the event appearing in the question.

4. Recall is Rational

People's responses to many verification questions did not conform with our initial expectations. A previous theory of question answering [Lehnert 1978] predicted that subjects would answer Q11 in one of the following ways:

Q11: Did John kidnap Susan?
A11a: No. The stranger kidnapped Susan.
A11b: No. There never was a kidnapping.

However, people overwhelmingly responded with a specification of the interpersonal relationship between John and Susan.

Allc: No. A father doesn't kidnap his own daughter.

Instead of simply accessing the kidnapping event and checking the role bindings, people appear to use their knowledge concerning the relationship between John and Susan to reason how likely such an event would be. If the story had been about a father who kidnapped his own daughter, then people would have remembered it as "that strange story about a father who kidnapped his very own daughter." Since this is so unusual, the memory representation would have treated such an event specially if it had occurred. When we are confronted with Q11, we immediately access the relationship between John and Susan. Once it is determined that a father kidnapping his own daughter would be highly unusual, we can check to see if the memory representation has flagged the kidnapping episode as being unusual with respect to its participants. When no such flag is found, we can conclude that the question harbors a false presupposition. So when a search for such a violation does not succeed, memory concludes that the event must not have occurred. Thus memory can be very certain of why its response to the question is the correct response. This kind of "rational recall" is very different from the "naive recall" of: a) finding the event and b) checking the role bindings of the event against the presumed role bindings in the question. If such heuristics were used by people, Alla and Allb would be common responses.

2. UNDERSTANDING QUESTIONS

Before a question can be answered, it must first be understood. What "understanding a question" means, however, is open to interpretation. Previous Q/A systems developed at Yale [Lehnert 78] have adopted the following general flow of control:

Phase I:

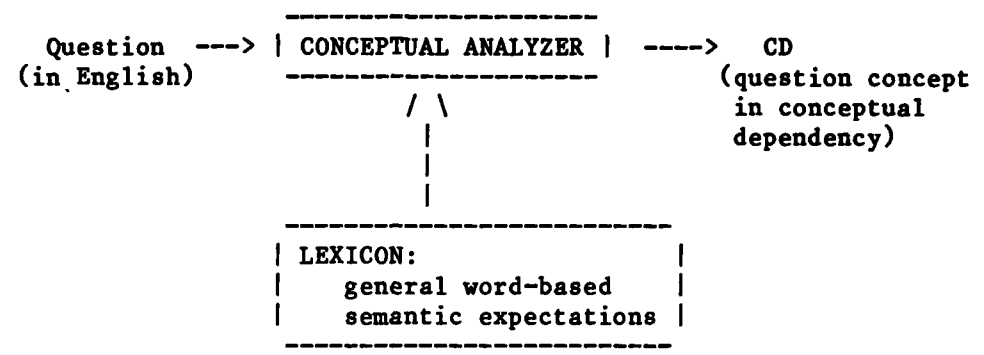


Diagram D.1

In Diagram D.1 above, the conceptual analyzer has access to semantic primitives and expectations associated with words in the lexicon. These structures and processes are used to build a language independent conceptualization of the question. Once the question is represented, a search process examines memory for a concept that matches the concept in question. If a match can be found, this structure is designated as the answer key. Once the answer key has been located, retrieval heuristics appropriate for the conceptual question type allow us to find the conceptual answer.

Phase II:

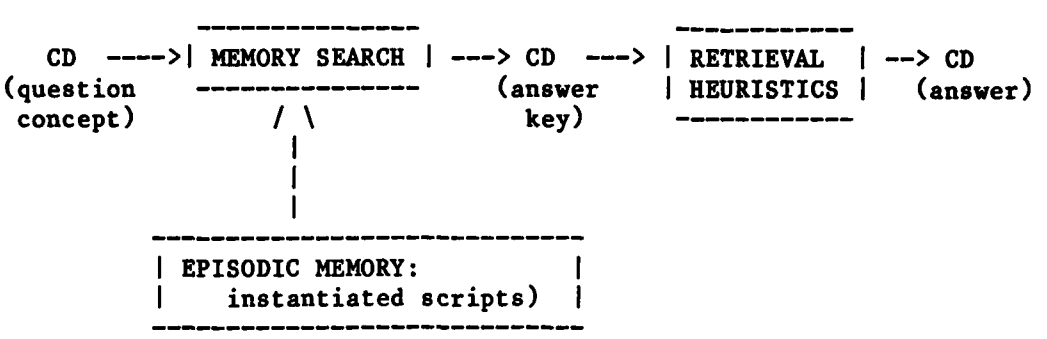


Diagram D.2

The conceptual answer can then be passed onto a conceptual generator to express the answer in some natural language (usually English).

2.1. INTEGRATED PARSING

As outlined, previous systems have relied on a highly modular division between the processes that understand the question, and the processes that find an answer. This design suggests that questions are understood without regard to the material over which they are being asked. But the following observations argue that this is not the case:

First, we asked people the following question about the kidnap story:

Q: Who called John about the stranger's kidnapping?
A: Mary did.

Then we asked:

Q: Who called John about Susan's kidnapping?
A: Mary did.

Even when both questions are presented in top of each other, people fail to notice the potential ambiguities. In fact, we had to point out to them that "stranger's kidnapping" could mean either that the stranger had been kidnapped, or that the stranger was the kidnapper. In the same way, "Susan's kidnapping" might mean "Susan's kidnapping of someone", or it might mean "Susan's kidnapping by someone".

Since people fail to notice any presupposition violations in these questions, they were evidently resolving the ambiguity as they were understanding the question. But there is no syntactic or semantic knowledge which could disambiguate either case. Only the episodic information available in the story representation could achieve this. If we have read the story, we know that Susan was the victim and that the stranger was the criminal. But if people access this knowledge in order to disambiguate the question, then they must be searching episodic memory during question-understanding time (in addition to question-answering time).

It follows that the parsing processes that analyze the question should be integrated with memory processes that search the story representation. Consequently, our program takes the following approach:

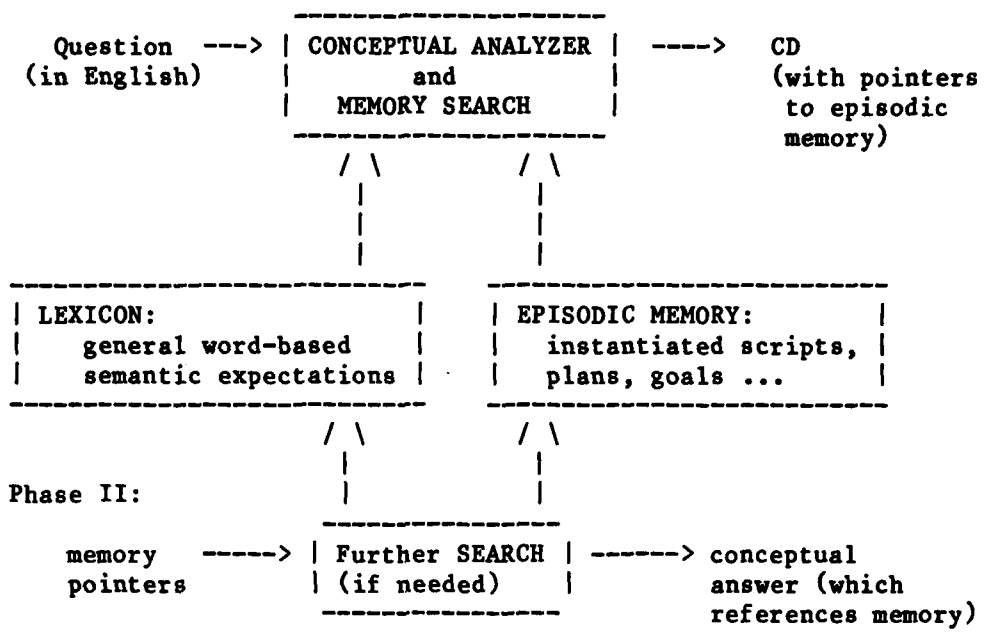


Diagram D.3

This approach has several consequences, of which the most dramatic are:

1. The answer to the question may be known the moment the entire question has been analyzed, thus obviating the need for a second search phase.

2. The answer to the question may be known even BEFORE the entire question has been understood.

3. During the parse of a question, episodic information from the story may be used to aid in understanding the question (such as in resolving ambiguous references).

These consequences are actually quite natural. For instance, when people are asked the following question (in the context of the kidnapping story):

Q: Who did Mary call at the party Friday afternoon?

they commonly claim that the answer "John" came to them the moment they either heard "Who did Mary call", or by the time they heard "at the party". In fact, it would be rather strange for people to refrain from searching their specific memory of the events in question until the entire question had been "understood", and only then start to work on answering it.

This more integrated scheme has an advantage insofar as once an episode in memory has been found, the information within it can become available to aid the understanding process. For example, if Mary's event of phoning John is recalled before "at the party" has been heard, then "at the party" can simply be checked against the scenario information attached to that phoning episode in memory. In the same way, "Susan's kidnapping" is disambiguated at question-understanding time by noticing that Susan was the victim of a kidnapping as soon as the kidnap episode has been found.

This scheme does **not** imply that episodic memory should guide parsing in a strictly top-down fashion. General semantic, syntactic, and word-based expectations continue operating during conceptual analysis. For instance, it is syntactic information about word ordering that tells us when a question's presuppositions violate episodic memory:

Q: Why did Susan kidnap the stranger?

Thus, the interaction between general knowledge and episodic knowledge during question understanding is heterarchical rather than hierarchical.

2.2. REPRESENTING QUESTIONS

The difference between integrated and non-integrated understanding can be seen in the output of the parser. For example, a non-integrated parser would analyze "Who called John at the party?" to produce:

```
(MTRANS ACTOR (PERSON SPEC (*?*))
        FROM  (PERSON SPEC (*?*))
        TO     (PERSON NAME (JOHN)
                       SEX (MALE))
        OBJECT NIL
        LOC  ($PARTY)
        MODE (VERIFY))
```

Example 1

In contrast, our integrated parser produces:

```
($PHONE  EVENT ($PHONO)
         CALLER (PERSON SPEC (*?*)
                        HUMAN (MARYO))
         CALLEE (PERSON NAME (JOHN)
                        SEX (MALE)
                        HUMAN (JOHNO))
         CALLEE-LOC  ($PARTY EVENT ($PARTYO))
         MODE (VERIFY))
```

Example 2

Notice that, in Example 2, the parser has already found both the event in memory and the person who made the call. In this case, the answer is therefore determined as the question is being understood.

The other information in the question concept (such as "(PERSON NAME (JOHN) SEX (MALE))") is also maintained to represent whatever information was explicitly stated in the question. This information is needed things as knowledge state assessment during answer generation. For example, if one asks "Who is John's friend?" then "Bill" a

reasonable answer because BILL0 is the conceptual answer and BILL0's name was not mentioned in the question. However, if we had been asked "Who is Bill?", then "Bill" would not be a reasonable answer, even though BILL0 is still the conceptual answer. Thus, each question concept is represented as a combination of structures built by parsing processes and accessed by both parsing and memory processes.

3. KNOWLEDGE REPRESENTATION

The process of understanding a story results in the construction of an internal, conceptual representation for that story. This representation consists of two major components: a) episodic memory about the story itself, which includes instantiated scripts, plans, and goals involving the various characters and events in the story, and b) indexing structures which allow retrieval processes access the story representation.

3.1. STORY MEMORY

The system's knowledge of the story is encoded as a conceptual graph composed of memory nodes and conceptual links between them. Each node represents the instantiation of a script (i.e. event), plan, goal or CD structure. Each conceptual link is uni-directional and is constrained by the kind of nodes it may connect. These constraints are summarized in D.4 below:

(E = event, G = goal, P = plan)

E forces E forced-by	G motivated-by E thwarted-by achieved-by	P realizes E
E motivates G thwarts achieves	G suspends G suspended-by	P intended-by G enabled-by
E realized-by P	G intends P enables	P _____ P

Diagram D.4

Links are grouped into pairs -- that is, for each link "L" from node N1 to N2, there is a corresponding link "L-by" from N2 to N1.

While the ultimate defense of any system of semantic categorization must depend upon its usefulness in processing, the intuitive meaning that each link-pair is intended to capture is described below:

1. Natural Events

Many cause-effect relations do not involve mental reasoning. For example, if an earthquake makes a building fall down, it is represented as:

```
$EARTHQUAKE0 -----forces-----> $COLLAPSE0
             <--forced-by-----
```

These links are designed to capture the cause-effect relationships that arise when natural forces operate in the world which are not mediated by plans or goals. Since natural forces do not occur in the kidnap story, these links will not be mentioned further.

2. Goal Motivation

Since events are actions that cause changes in the state of the world, and goals are descriptions of desired states. The occurrence of an event may cause the creation of goals in the characters affected or involved in that event. For example, the kidnapping of Susan by the stranger causes John to want to save Susan. This is represented as:

```
$KIDNAP0 -----motivates-----> P-HEALTH0
         <--motivated-by----
```

Where P-HEALTH0 refers to John's goal of preserving Susan's health.

The processing that arose during the creation of this link is saved as a processing trace (PT) associated with this link. In this case it contains the following information:

1. Kidnapping threatens the victim's health.
2. Threatening y's health causes y to have a P-health goal.
3. If x loves y, then y's goals are x's goals.

This information will be useful during reconstructive reasoning.

3. Goal Failure

Just as events can motivate goals, they can also thwart the achievement of a goal. For example, when John tells Karen that Mary is not home, this thwarts Karen's goal of communicating with Mary. It is represented as:

```
$PHONE2-TERM -----thwarts-----> D-KNOW2
             <--thwarted-by---
```

where D-KNOW2 represents Karen's goal of telling something to Mary.

Notice that these links say nothing about which character is aware that a goal has been thwarted. It is enough for the understander to realize that Karen's goal was not achieved and why.

4. Goal Satisfaction

Events can also bring about states that are desired by one or more characters in the story. In such cases, the event is said to have "achieved" the goal. For example, once Mary finds out about Susan's kidnapping, she wants to inform John. She calls John at the party and tells him. Her goal is achieved by the successful completion of a phone

call. This situation is represented as:

$PHONE0 ----achieves----> D-KNOW0
<--achieved-by--

Phone calls by themselves do not necessarily satisfy D-know goals. It is the fact that the message (describing this goal) was successfully transmitted from the caller to the intended recipient that accounts for the goal achievement.

5. Goal Suspension

Whenever a crisis goal, or high priority goal is motivated by some event, the goal-holder may temporarily drop his currently active goals in order to attend to the more important goal. For example, when John finds out that Susan has been kidnapped, he quits thinking about enjoying the party or getting his money back from Bill. This situation is represented by marking each goal as being suspended by the crisis goal:

P-HEALTH0 ----suspends----> {E-ENTERTAIN0,D-CONT0}
<--suspended-by--

Crisis goals will intend plans, but sometimes the plans may not be well specified. In such cases they are vague plans. For instance, we are not sure exactly what John is going to do to save Susan (and maybe neither is John), but we know that John's actions are the result of his P-HEALTH goal.

6. Event Realization

When plans are actually executed they bring about (or help to bring about) the occurrence of events. For example, the phone call that John

receives at the party is the direct result of Mary's plan to phone him. This is represented as:

PLAY-CALLER0 ----realizes----> $PHONE0
<--realized-by--

PLAY-CALLER0 instantiates a plan which contains information about how to make a phone call, such as: finding the number to dial, dialing, listening to the dial tone, saying "hello", asking for the person being called, etc. Many of these "plans" are in fact event sequences defined by script roles. This is always the case when the event realized is a script instantiation. We will nevertheless refer to these sequences as plans, with the understanding that scriptal behavior is often used in the service of planned activities.

7. Plan Intention

Once a character has an active goal, that character may attempt to satisfy that goal by means of a plan whose execution is intended to achieve it. For example, at one point Mary has a goal of notifying the police. Her intended plan is to call them. This is represented as:

D-KNOW4 ----intends----> PLAY-CALLER4
<--intended-by--

It doesn't matter that Mary never actually executes this plan. In fact, it is important to be able to distinguish plans as intentions-to-act from plans as the execution-of-actions in the service of a goal. If Mary had actually called the police, then D-KNOW4 would also have achievement links connecting it to a $phone instantiation and there would be a realizations link from PLAY-CALLER to the instantiated

$PHONE script.

8. Plan Enablement

Many times, before a plan can be executed, there are enablement conditions which must be satisfied. For example, John can not enjoy the party simply by having been invited to it. John has to actually be at the party. (Otherwise he could stay at the office and enjoy the party there.) This precondition on E-ENTERTAIN [party] sets up a sub-goal of D-PROX, which represents John's goal of being at the party. This "delta prox" (i.e. change in proximity) will itself set up some intended plan (e.g. PLAY-DRIVER) whose execution will satisfy this precondition. This situation is represented as:

D-PROX0 ----enables----> E-ENTERTAIN0
<--enabled-by--

D-PROX0 ----intends----> UNSPECIFIED PTRANS
<--intended-by--

Thus, the connection between goals and sub-goals is mediated by enablement conditions on plans. This is done since the understanding of a story involves being able to fit the explicit actions made by characters in the story with the implicit intentional structures that gave rise to these actions.

Notice, in this representational system, that plans do not directly achieve goals. Rather, the execution of an intended plan may bring about (or help to bring about) the occurrence of an event. The fact that an event has occurred may then result in satisfying some character's goal.

There are several reasons for setting things up in this way:

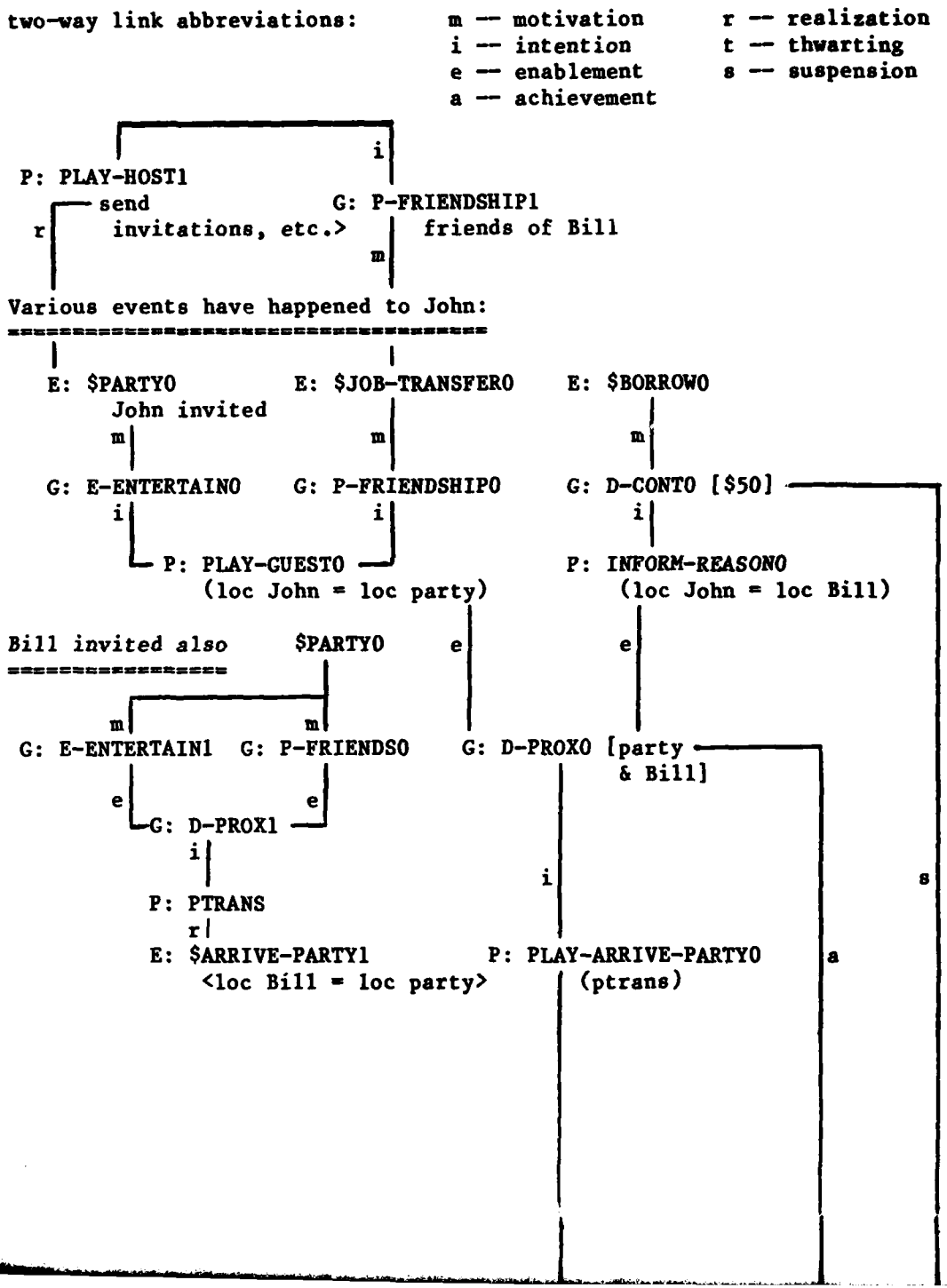
a. Memory can keep track of the status of any goal or plan simply by checking these conceptual links. If a goal has been marked motivated, suspended, and achieved, then memory knows the 'life' of that goal. For example, John's goal of getting $50 from Bill is first motivated and then left in a state of suspension, while Mary's plan to call the police never gets past the state of an intention.

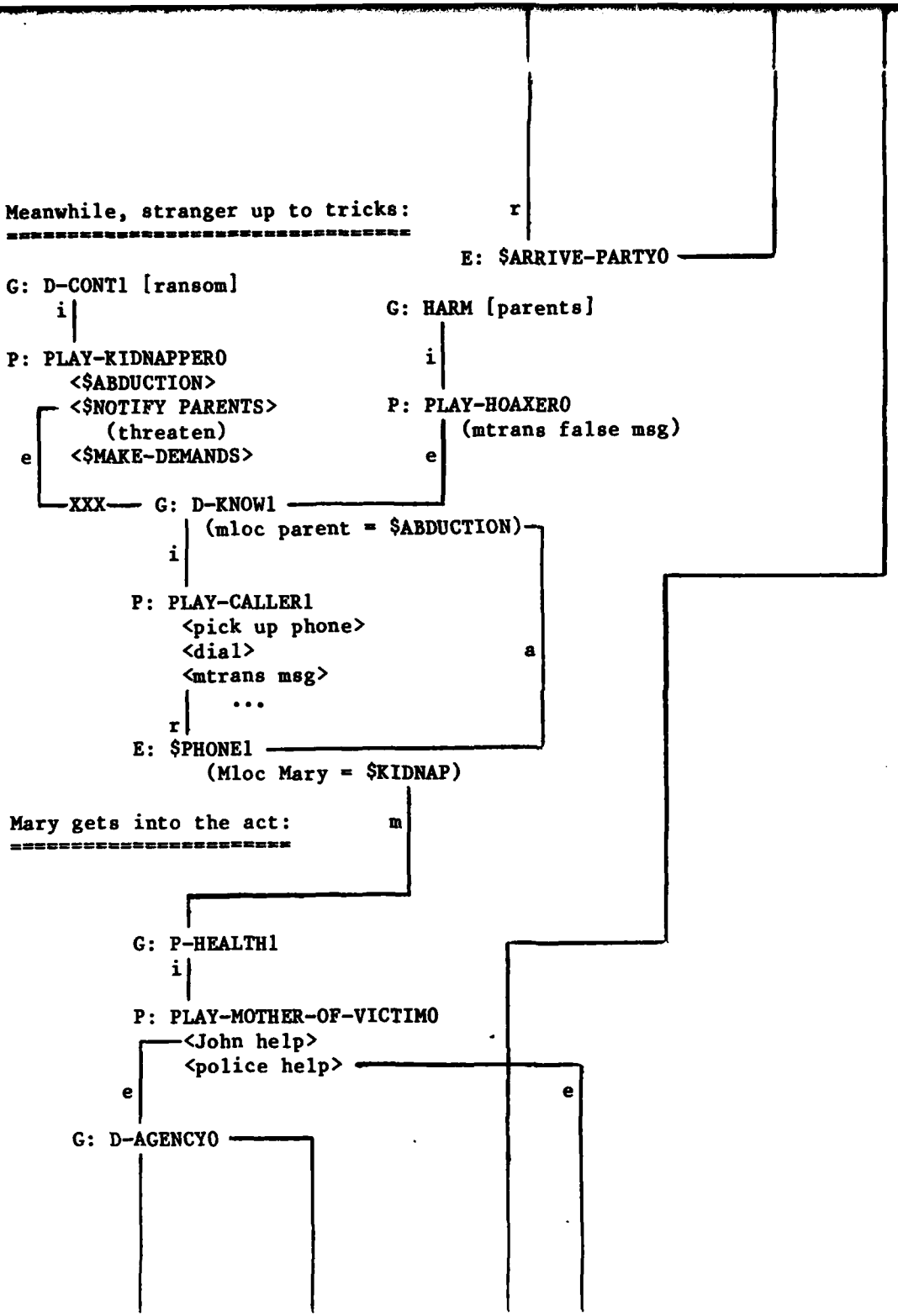
b. Memory can distinguish the intention to achieve a goal from the intention to execute a plan to achieve a goal. This is important in the case of **fortuitously** achieved goals. In such cases the event which achieves a goal is **not** realized by the plan that was intended by the goal holder. For example, John has the goal of acquiesing to the stranger's expected ransom demands. However, Susan walks in and and John's goal to save Susan has been achieved. But John is surprised. Why? The answer seems obvious enough, but how do we capture our understanding of this situation? John's goal has been achieved by Susan's arrival, which we represent as:

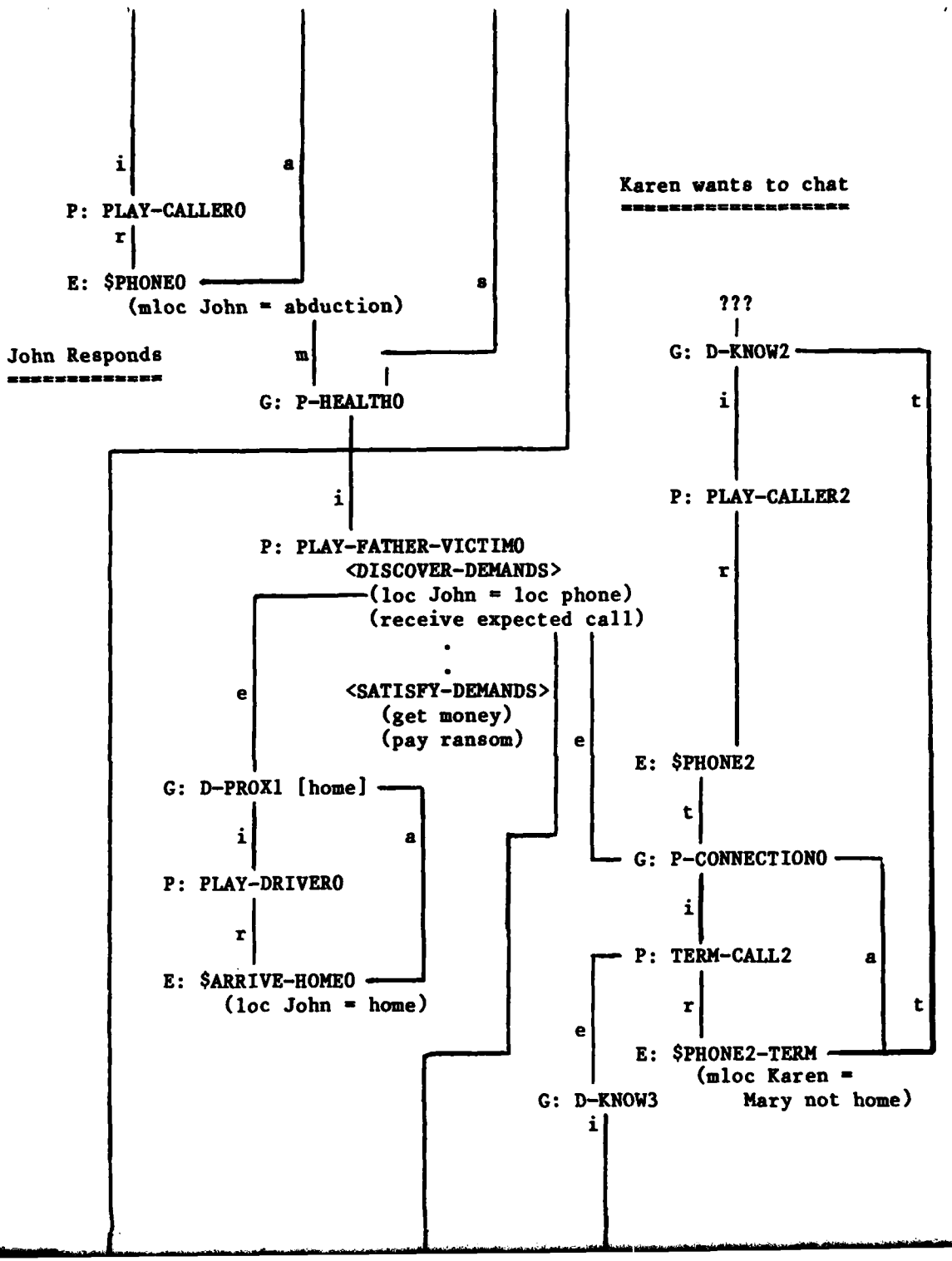
$ARRIVE-HOME1 ----achieves----> P-HEALTH0
<--achieved-by--

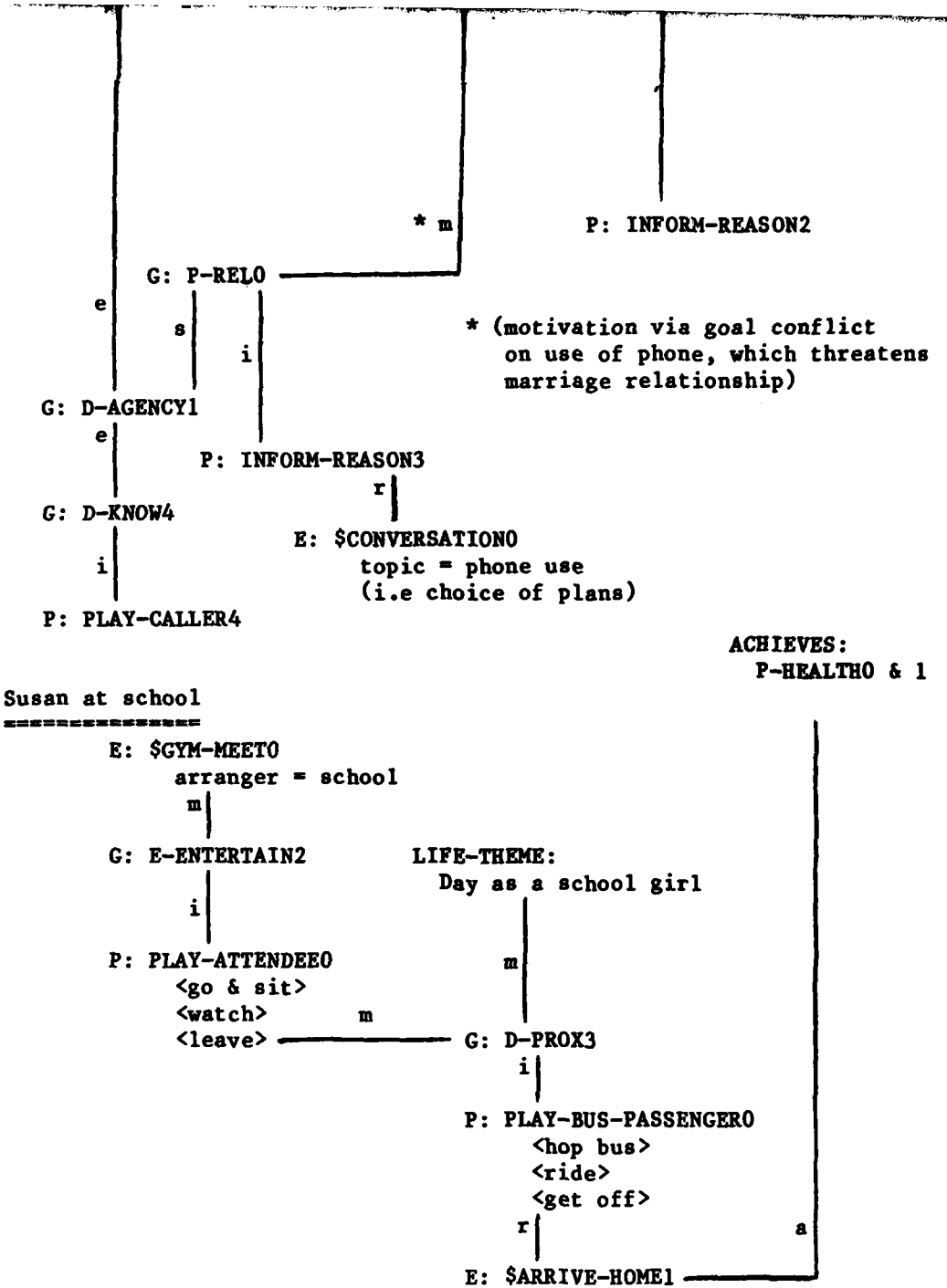
However, John's plan, that was intended to satisfy his P-HEALTH0 goal, had nothing to do with the realization of Susan's arrival home. **That** is why John is surprised. By using the scheme presented here, we can represent this state of affairs.

Now we are in a position to examine the foldout which presents a picture of the overall structure of the story representation.


two-way link abbreviations:
m -- motivation
i -- intention
e -- enablement
a -- achievement
r -- realization
t -- thwarting
s -- suspension
P: PLAY-HOST1
send invitations, etc.>
G: P-FRIENDSHIP1
friends of Bill
Various events have happened to John:
E: $PARTY0
John invited
E: $JOB-TRANSFER0
E: $BORROW0
G: E-ENTERTAIN0
G: P-FRIENDSHIP0
G: D-CONT0 [$50]
P: PLAY-GUEST0
(loc John = loc party)
P: INFORM-REASON0
(loc John = loc Bill)
Bill invited also
$PARTY0
G: E-ENTERTAIN1
G: P-FRIENDS0
G: D-PROX0 [party & Bill]
G: D-PROX1
P: PTRANS
E: $ARRIVE-PARTY1
<loc Bill = loc party>
P: PLAY-ARRIVE-PARTY0
(ptrans)


Meanwhile, stranger up to tricks:
r
E: $ARRIVE-PARTY0
G: D-CONT1 [ransom]
i
P: PLAY-KIDNAPPER0
<$ABDUCTION>
<$NOTIFY PARENTS>
(threaten)
e
<$MAKE-DEMANDS>
G: HARM [parents]
i
P: PLAY-HOAXER0
(mtrans false msg)
e
XXX
G: D-KNOW1
(mloc parent = $ABDUCTION)
i
P: PLAY-CALLER1
<pick up phone>
<dial>
<mtrans msg>
...
a
r
E: $PHONE1
(Mloc Mary = $KIDNAP)
Mary gets into the act:
m
G: P-HEALTH1
i
P: PLAY-MOTHER-OF-VICTIM0
<John help>
<police help>
e
e
G: D-AGENCY0


i
a
P: PLAY-CALLER0
r
E: $PHONE0
(mloc John = abduction)
s
John Responds
m
G: P-HEALTH0
i
P: PLAY-FATHER-VICTIM0
<DISCOVER-DEMANDS>
(loc John = loc phone)
(receive expected call)
e
<SATISFY-DEMANDS>
(get money)
(pay ransom)
G: D-PROX1 [home]
i
a
P: PLAY-DRIVER0
r
E: $ARRIVE-HOME0
(loc John = home)
Karen wants to chat
???
G: D-KNOW2
i
t
P: PLAY-CALLER2
r
E: $PHONE2
t
G: P-CONNECTION0
i
P: TERM-CALL2
a
r
t
e
E: $PHONE2-TERM
(mloc Karen =
Mary not home)
G: D-KNOW3
i



FOLDOUT: EPISODIC MEMORY

Some points of interest in the foldout are discussed below:

- John's d-prox goal of being near Bill (in order to say "goodbye" to Bill and to ask Bill for the $50), and of being at the party, are one and the same goal. This can only be realized by having built a representation of Bill's goal structure. That is, John knows that Bill will also want to be at the party in order to enjoy himself. Otherwise, it would make no sense for John to go to the party in order to obtain the money owed.

- The connection between Bill's going to California and the existence of a party lies with the reasons for why Bill's friends have decided to throw the party. This is represented here as a P-FRIENDSHIP goal (preserve one's friendship with someone) on the part of Bill's friends. Thus, some (unmentioned) friends of Bill have thrown a party for Bill in order to express their friendship for him, given that they won't be seeing him as often in the future. Playing the role of host(s) in a $PARTY script is represented by executing a PLAY-HOST plan, which contains information about how one realizes a party. One action in this plan involves sending out invitations to guests. This action realizes the event of John's being invited to the party, which is represented as $PARTY0 with John bound to the role of GUEST.

- Since the story does not specify how John got to the party, his plan for getting there is represented by the primitive CD act of PTRANS, which captures the notion of change in location without specifying how this was accomplished.

- $ABDUCTION is the actual event of grabbing Susan that is part of

the stranger's plan for getting a ransom. $KIDNAP, however, is the script which represents ALL the knowledge that memory (and the characters) have about all the roles each person in the kidnapping will play. Since John knows about $KIDNAP, this allows John to know the role that the stranger is playing, and therefore the plan the stranger intends to use. So John can expect demands to be delivered when the stranger calls back (without the stranger having to tell John that this is what he will do **when** he calls back). In addition, since the PLAY-KIDNAPPER plan specifies abducting the victim **before** notifying the victim's parents, it is assumed that Susan has, in fact, been abducted. Thus, a plan is a sequence of enabling goals and acts. Each plan, when part of a script, functions as a role or part to be played. Scripts serve as knowledge structures (which both the characters and the story-understander have) that organize the interaction of plans or script (roles).

- The expectation violation which occurs at the end of the story is represented by the fact that the D-KNOW1 goal of the stranger to inform the parents is first understood as serving an enablement condition within the PLAY-KIDNAPPER plan. At the end of the story, this D-KNOW1 turns out to serve the enablement condition of a different plan (i.e. PLAY-HOAXER).

- In general, there are two possible responses a relative of a kidnap victim can make: a) get help (usually from the police and/or other relatives), or b) follow the kidnapper's instructions. These plans are not mutually exclusive. At first it is not immediately clear what specific plans John (or Mary) is invoking to save Susan. It is

assumed, however, that whatever John does is part of his plan. This is represented by PLAY-FATHER-VICTIM0, which refers to John's role as the father in $KIDNAP0. As the story progresses, we realize that Mary wants help from the police (i.e. D-AGENCY1), while John wants to hear the ransom demands and (possibly) satisfy them. In order to do this, John has to be near the phone. Thus, John's plan is enabled by satisfying the goal of being home. The plan associated with D-PROX1 is to drive (i.e. PLAY-DRIVER0). This plan is executed and realizes the event of $ARRIVE-HOME0.

- When the phone rings (i.e. $PHONE2), the fact that it is Karen interferes with an enablement condition on receiving the stranger's call. This interference causes John to want to keep the phone available (i.e. P-CONNECTION0) for the stranger's expected call. Like most preservations goals, it is only activated when some specific event interferes with its maintenance. John can satisfy P-CONNECTION0 by thwarting Karen's goal of talking to Mary (i.e. D-KNOW2). John terminates $PHONE2 by telling Karen that the enablement condition (i.e. Mary's being there) on D-KNOW2 is not satisfied. The specification of this goal derives from the object-primitive decomposition of a telephone as a CONNECTOR object [Lehnert 1978].

- Meanwhile, Mary's goal of contacting the police (i.e. D-AGENCY1) conflicts with John's plan of receiving the ransom demands. The conflict is represented by P-REL0, which refers to Mary's goal of maintaining her relationship with John. Since this requires goal agreement, Mary decides to convince John (i.e. INFORM-REASON) that they should call the police. It is this plan of Mary's that realizes the

$CONVERSATION0 in which they are engaged when Susan arrives.

- Meanwhile, Susan finishes attending a gym meet and takes a bus in order to get home. The reason why Susan wants to go home is related to her life-theme of being a student. (I.e. that's what students do at the end of a school day.) Her plan to take the bus achieves her D-PROX3 goal and realizes the event of $ARRIVE-HOME1. This event achieves both John and Mary's P-HEALTH goals. As mentioned earlier, the unusualness of this situation manifests itself in the fact that John and Mary's P-HEALTH goals are achieved by an event which is not realized by any of their own plans.

3.2. ACCESS STRUCTURES

1. Why A Discrimination Net Won't Work.

One obvious candidate for use as an access structure is the discrimination net (d-net). For instance, one could set up a d-net each of whose "leaves" would point to some node in episodic memory. Answer keys to questions would then be found simply by using the information in each question to trace a path through the net to a leaf. An example of one such d-net is presented in Diagram D.5 below:

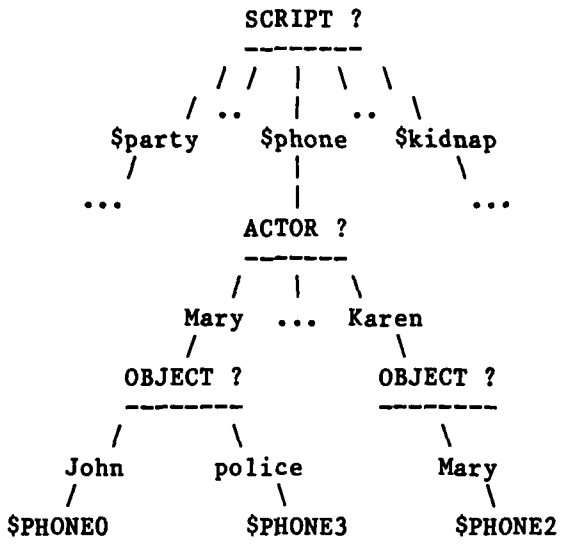


Diagram D.5

The way this d-net might work is as follows: Suppose we are given the question:

Q: Did Mary call John at the party?

Once we have parsed this question into its semantic constituents:

(script = $phone
actor = Mary
object = John)

we could then use the d-net above to retrieve $PHONE0 as an answer key. Thus, a d-net could serve as a structure which "sits on top of" episodic memory and acts as a general access mechanism. Since, at first glance, this general mechanism seems to work so well, we must explain why it has rejected.

There are several reasons why d-nets are inadequate in the area of

question-answering:

a. The Categorization Problem.

If we use a d-net, we must then decide which discriminations will be made first (for these, by definition, will be the most important). For instance, in Diagram D.5, we chose scripts as the initial category for discrimination. The next discriminations involved the actor, and then the object. However we gave no reason why the d-net couldn't have been set up to discriminate initially on the basis of some other category (e.g. the actor, the object, the actor's goal, emotional state, on scenario). Simply stated, the "Categorization Problem" with respect to d-nets consists of the following observation:

> Whichever category we choose for the initial discrimination, the resulting d-net will fail to work for those questions seeking an answer involving that very same category. (This problem will occur **at every level** within the d-net.)

For instance, if the d-net uses **scripts** as a discrimination category, then it will fail to handle such questions as:

> Q: What was Susan doing?
>
> Q: What happened between John and Bill three years ago?

This failure occurs because the d-net is relying upon having a script as a means of pathing into the story. It can not very well be expected to handle questions which leave out any mention of a script.

Alternatively, if we set up a d-net to use the **actor** as a

discrimination category, then it will be incapable of handling the following kinds of questions:

> Who called John at the party?
> Who drove home?
> Who borrowed $50 from Bill?

which seek the actor involved.

Again, if we choose to discriminate off of the **object** or **scenario**, then we won't be able to use this net on the following kinds of questions:

> Who did Karen call?
> Where was Susan?

Thus, d-nets fail as a general access mechanism.

b. The Combinatoric Problem.

One approach to the Categorization Problem involves using many discrimination nets to create multiple indexing. The problem with this approach arises from the combinatoric explosion of paths in the d-nets. That is, a path which discriminates over, say, scripts and then actors must, in some other place, be making essentially the same discriminations, except this time over actors before discriminating over scripts. In general, we get a path we get a redundancy of n! for a tree of depth n, as depicted in D.6 for n-2:

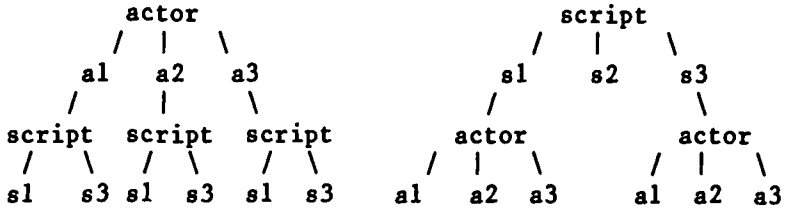


Diagram D.6

As can be seen in this diagram, in order to be able to discriminate over either the actor or the script, we have to double the number of equivalent paths in the nets.

c. The Closure Problem.

Finally, d-nets simply fail to take into account certain kinds of questions. This is because d-nets assume that each episode in the story will exist somewhere in the net as a leaf. This "closure" assumption, however, is violated by the following types of questions:

- Questions which ask about events that never happened in the story. For example:

Why didn't John talk to Bill?
Why wasn't Karen worried?

People can answer these kinds of questions with no trouble, yet how can we set up categories for discriminations which involve non-existent events?

- Questions whose presuppositions violate the story. For example:

Why did Karen want to talk to John?
Why did John call the police?

Again, people are pretty good at noticing when a question's assumptions are in violation with what happened in the story. Yet they are able to recall appropriate events so as to be able to correct and comment upon these presupposition violations. With a d-net, however, such questions will simply fail to path to any leaf in the net.

- Verification questions which have negative answers. For instance:

> Did Mary call Karen?
> Did John kidnap Susan?
> Did Karen want money from Bill?

These kinds of questions are similar to presupposition violation questions. However, they pose a problem for a general d-net mechanism. For instance, given the question, "Did John kidnap Susan?", with **which** d-net (or **where** in a single d-net) do we start? As we have currently defined the d-net mechanism, we would only be able to answer "No." by failing to find a path to a leaf in all of the nets. However, people are usually good at augmenting a negative answer with an elaboration which indicates WHY they know the answer to be negative. The Q/A example below shows a common answer people give to this question:

> Q: "Did John kidnap Susan?"
> A: "No. Fathers don't kidnap their own daughters."

This answer does not seem to involve a general d-net mechanism in operation. This is **not** to say that discriminations are useless. In fact, many forms of discrimination are a necessary part of any model involving cognitive processes. The proliferation of the LISP COND (itself a d-net!) is proof enough of this. We are simply saying that

d-nets fail, as a general mechanism, in handling memory search processes. So what is the alternative?

To find the answer, we have been studying what **people** do, how **people** search their memory in answering different questions about various events in a story. We hope to discover the organizing principles that people use and then model own search/retrieval heuristics on those strategies.

There are some important consequences to this approach:

- Such a program will exhibit the same recall that characterizes the way people recall stories they have read. This means that the program will produce answers which are sometimes incorrect or incomplete.

- Such a program will occasionally give conflicting responses, such as when people change their minds, by first saying "Yes", and then "No" in answer to the same question. For instance:

Q: Who kidnapped Susan?
A: The stranger did.
No. Wait...
She never was kidnapped. It was a hoax.

What is important, then, is not some general mechanism expected to work in all cases, but a psychologically reasonable set of search heuristics and memory organization principles which closely model people's performance in these tasks.

This view of question-answering process design is based upon the

assumption that people are the best mechanisms known for answering questions about stories. Exactly how people do it should therefore be well understood before any attempt is made to improve upon these mechanisms.

2. Using Relationships

Interpersonal relationships are important organizing principles in the story, and their influence can be seen in several ways. First, people tend to forget the characters' names, and recall, instead, their status within some interpersonal relation. For example, John is referred to as "the father" or "the husband"; Bill is "John's friend", etc. In addition, people seem to traverse relationships in order to recall the characters in the story. They usually recall John's family and friend, then Mary's friend. Many times the stranger is left out. This oversight can be explained by observing that the stranger is not involved in any interpersonal relationship, and by assuming that those processes which attempt to recall characters make use of some map of interpersonal relationships.

The relationship map of the story is shown in Diagram D.7 below:

Relationship Map

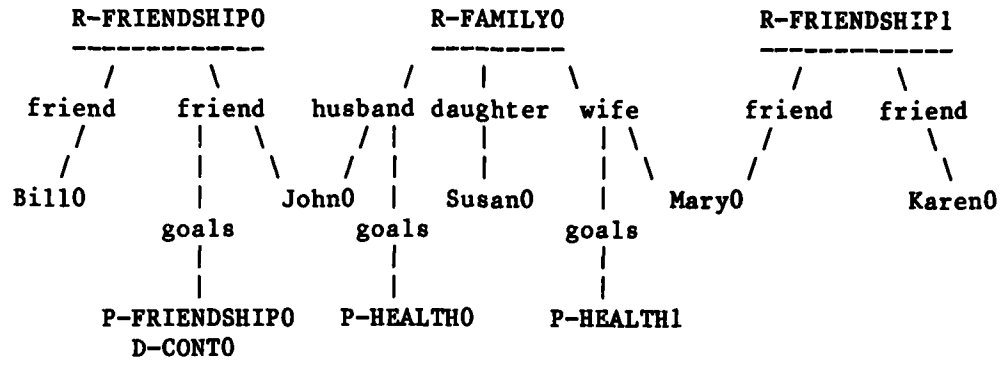


Diagram D.7

As can be seen, there are three interpersonal relationship structures in the story, and these serve to connect the characters to one another. It is important to know the relationship between characters because it can be used to infer both themes (such as LOVE), and certain goals directly related to relationship maintenance. For example, when people are asked:

Q: Why did John want money from Bill?

They invariably answer:

A1: Because Bill had borrowed money from John.

They never think of the following reasonable alternative:

A2: To help pay the ransom.

However, when we ask people a general question about money, such as:

Q: Did the story involve money?

they tend to think of the ransom instead of the $50. How do we explain this?

One likely possibility is that when people read that Bill borrowed $50 from John, they realize that this act is appropriate, given that John and Bill are friends. (After all, what are friends good for if not for such things are resource sharing?) We can also predict that, if Bill fails to pay John back, that could possibly harm their relationship. Thus, John's goal of getting $50 back is associated with his friendship with Bill. When people are asked about a goal involving John and Bill, they can access these goals and find one which matches the question concept. In this way they naturally think about one friend borrowing $50 from another without thinking about the kidnapping (which involves John's family, but not John's friend, Bill).

In the same way, the relationship map is used to answer such questions as:

Q.1 Why was John worried about Susan?

Q.2 Did John kidnap Susan?

In processing each of these questions memory can't help but think of the relationship between John and Susan. In the case of Q.1, the relationship map is used to recall John's goal of saving Susan from the kidnapper, while in Q.2, the map is used to infer a LOVE theme which is violated by the presupposed act of kidnapping mentioned in the question.

3. Scenario Participation

In order to make use of the representation described above, there must exist access paths which allow processes entry into memory. These access paths are built (along with the episodic memory) at story understanding time. For example, we can not begin to answer the question: "Why did Mary call John at the party?" unless we can get from the English words "Mary", and "John" to the characters JOHN0 and MARY0 in the story, and from the words "call" and "party" to the appropriate $PHONE event within the party scenario. Some of these access paths are presented in the form of rules and will be discussed later. Other access paths are better thought of as "maps" of memory which are examined by various search processes seeking entry points into the story. One such map is called a "Scenario Participant Map" and is depicted in Diagram D.8. The existence of this map was postulated to handle such questions as:

Q.1 Why did John go to the party?

Q.2 Why did John leave the party?

Q.3 Where was Susan?

The conceptual representation of "go" and "leave" is PTRANS. But which PTRANS? There are numerous PTRANSs in the story, both explicit and inferred. For example, John PTRANSed to two phones, to a car, to the numerous doors, etc. In order to find the appropriate D-PROX goal being referenced in the question, we must make use of the scenario information available in the question. By first 'locating' John at the party and then 'recalling' where John was next, we can access John's d-prox goal in episodic memory. Once we have this entry point, other

search processes can retrieve the higher goals (or events) which cause John to want to be home. But how is this 'locating' and 'recalling' accomplished?

The scenario participant map accomplishes this task by containing information concerning the movement of characters in the story, and the situations surrounding the contacts which occurred between characters. We believe the scenario (such as the party, home, school) serves as an important organizing principle in memory. That is, events are associated with the scenarios within which they occurred, and goals are associated with the scenarios within which these goals were motivated or achieved. Furthermore, at story-understanding time, changes in scenario are important for recognizing such things as changes in character perspective and the termination of events. Whenever a character switches from one scenario to another, or whenever a character tries to contact another character across scenarios, the reason for this can be formed associated with the appropriate delta-goal in the scenario map.

As can be seen in D.9, the Scenario Participant Map (SPM) summarizes the general actions which occurred in the story: i.e. John went to a party. At the party he attempted to contact Bill. Mary contacted John. At home, a stranger contacted Mary. John came home. Mary wants to contact the police. Susan was at school. Then Susan came home:

SCENARIO PARTICIPANT MAP:

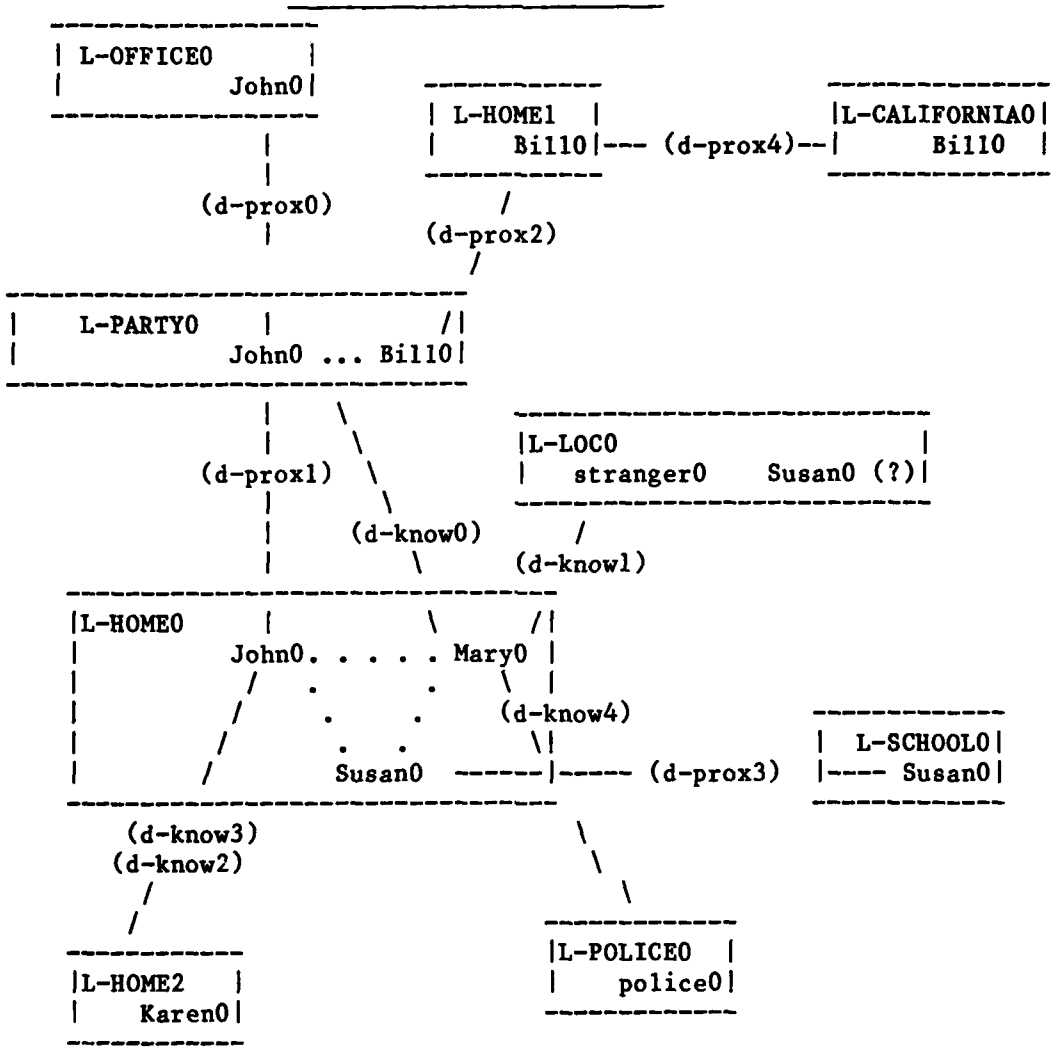


Diagram D.8

Thus, the SPM serves as a mental 'map' of the story, keeping track of each setting for each character, and for the contacts between them. Furthermore, each scenario has events associated with it, and there are search processes operating upon scenarios to recall those associated events. These processes are use to answer such questions as: "What

happened at home?" and are discussed in section 4.

4. SEARCHING MEMORY

The processes which search memory may be thought of as divided into three general categories:

a) One set of processes have the task of finding a node in the story representation, given only pointers to those general knowledge structures which were referenced by words or phrases in the question. Such a process is called **Instantiation Recognition** because it attempts to find an appropriate instantiation (in episodic memory) of some knowledge structure (such as a script or CD) available in general memory. Once memory nodes have been found, processes from the next two categories can perform their operations.

b) Once an instantiated structure is located in the story representation, another set of processes are responsible for examining this structure. These processes are called **Primitive Access Functions**, because they perform standard retrieval tasks specific to the general structure at hand.

c) Primitive Access Functions are usually executed in the content of a larger search strategy. **Heuristic Search Processes** are complex retrieval strategies that make repeated use of the Primitive Access Functions in order to arrive at conceptual answers to questions.

4.1. INSTANTIATION RECOGNITION

A very basic type of search process involves getting from an English word (or description) involving a person or situation to its associated character or event in story memory. These tasks are handled

by "instantiation" searches which try to find an instantiation in memory of each general knowledge structure referenced in the question.

1. Name Instantiation

People often maintain a direct associative link between a character's name and the unique memory token which serves to instantiate that character in the story. In such cases, Name Instantiation Access is used to immediately recognize a character from a given name. This function traverses associative links from specific names to specific memory tokens for characters.

But sometimes these associations are lost. Then the subject forgets the names of the characters in the story, or forgets which names refer to which characters. This difficulty does not generally stop people from answering questions which mention characters' names. In such cases, people rely on character search heuristics in order to derive a likely character for a given name. Derivations of this sort require a more complex memory access strategy, and will therefore be described in section 4.3.

2. Unique Event Instantiation

When a script (such as a party or kidnapping) is mentioned, people are capable of accessing the unique events in memory associated with that script. Unique Event Instantiation is responsible for recognizing script instantiations.

Unique Event Instantiation accounts for the answers people can give to questions such as:

Q: Was there a party in the story?
A: Yes.

Q: Was there a kidnapping in the story?
A: Yes. ... Well, no, it was a hoax.

Q: Was there a phone call in the story?
A: Yes. Several.

In the second example above, Unique Event Instantiation has found a kidnapping event in memory, so a positive response occurs. The contradicting answer: "Well, no, it was a hoax." then results from a primitive access function which examines the status of the memory node located. In the third example, memory uses Unique Event Instantiation to realize that multiple phoning events exist in memory. When more than one instantiation of a given script exists in memory, then a more complicated search heuristic must be invoked to decide which event is being referenced.

3. Unique Scenario Instantiation

Given a setting, people have no difficulty recalling the scenario of events associated with that setting. This happens because most settings refer to script-based or relationship-based structures. For example, "home" implicitly references family relationships while "party" and "office" refer to scriptal structures. Specific processes sensitive to such structures can then be evoked. (These will be discussed later.) When a setting serves as a unique backdrop for a scenario of events in the story, then the Unique Scenario Instantiation can access the scenario associated with it in the story representation. Such access occurs in the following kinds of questions:

Q: Whose home was it?
A: John's and Mary's.

Q: Who was at school?
A: Susan.

In the second example above, there is only one school setting mentioned in the story. In the first example above, "home" could refer to Karen's or Bill's home, but John's home was the only setting in the story that served as a scenario for events occurring in the story.

4. Summary

Names, scripts, and settings are not the only things that get mentioned in questions. Many words reference general knowledge structures such as primitive acts ("go", "talk", "give") and physical objects ("money", "door", "bus", "beer").

For each of these entities, their appropriate instantiations must be found in episodic memory. Similarily, a question may refer to a character's goal or plan, as in:

Q: Why did John want money from Bill?
A: Bill had borrowed $50 from John.

How do people get from their knowledge about money in general to the $50 owed in the story? One possibility would be "Object Instantiation Access." But this seems unlikely since objects like money may be used by many characters in numerous events. In such a case, it would be difficult to decide which node "money" is refering to out of context.

In fact, people have great difficulty instantiating "money" out of context. For example, when given the question:

Q: What involved money in the story?

they look blank for several seconds, and then finally respond:

A: The ransom.

This is an interesting response, since no ransom was ever actually mentioned in the story. Evidently, people were employing reconstructive search techniques while using the main event as a default starting place. (Reconstructive recall will be discussed in section 4.4).

We will therefore take the approach that the instantiation of a physical object can only be directly accessed within some specific context, be it scriptal, goal-based, or relationship-based. Once a context has been supplied, then the object referenced will be located by search processes specific to that context.

The same approach is taken with primitive acts, such as "go" or "talk", which refer to PTRANS and MTRANS, respectively. Again, John PTRANSed to various places: to several phones, to the party, to the car, to the door, etc. John also talked to Mary on several occasions. Thus, a context must be found by other search processes before a primitive act instantiation can be located. That is why the following question:

Q: What events in the story involved talking?

sounds odd, and is difficult to answer. To answer this, people have to ask themselves "Who were the characters in the story?"; "What were they doing?", and so forth. That is, people must first search for a context before tackling the original question.

4.2. PRIMITIVE ACCESS FUNCTIONS

There are 26 Primitive Access Functions. These retrieval mechanisms operate on instantiated memory structures within the story representation (characters, scripts, plans, objects, etc.). Within this class of retrieval operations, there are two basic types of Primitive Access Functions: (1) Standard Link Traversals, and (2) Narrative Link Traversals. The standard links are those outlined in Diagram D.4 on page 16. We have implemented primitive access functions for 14 of these 16 standard links (the "forces" and "forced-by" links have not occurred in any of our BORIS stories). The last 12 Primitive Access Functions rely on narrative links which will be described as each function is presented.

1. Motivated Goal Access

Given an event and a character involved in that event, people are able to recall the goals that were initiated by the event.

Q: How did the kidnapping affect John?
A: It made him worry about
his daughter's safety. (P-health)

Q: How did Bill's loan affect John?
A: It made him want his money back. (D-cont)

2. Event Motivation Access

Given a goal on the part of a specific character, people can remember the event which gave rise to that goal.

Q: Why was John concerned about Susan?
A: Because she had been kidnapped.

Q: Why did John want $50 from Bill?
A: Because Bill had borrowed $50 from John.

The hard part here is finding an instantiation in memory of the goal mentioned in the question. How this is done will depend upon other information in the question, and is discussed in section 4.3.

3. Goal Intention Access

Given a character's plan of action, memory can retrieve the goal which this plan is intended to achieve. For example:

Q: Why did John drive home?
A: To get home.

Q: Why did Susan take a bus?
A: To get home.

4. Intended Plan Access

Given a character's goal, memory can retrieve the plan of action intended by the character to achieve that goal. For example, once we recall John's goal of being at home, we are then in a position to try to remember the plan John used to get home:

Q: How did John get home?
A: He drove.

Q: How did the stranger let Mary know about the kidnapping?
A: He phoned Mary.

Q: How was the stranger going to get money?
A: By kidnapping Susan.

The last question above concerns retrieving a plan that was intended to accomplish a goal, but which was never actually invoked (or completed). The "going to" in the question informs us that the goal was never achieved. This information is used by processes trying to instantiate a goal of the stranger's. Once this goal has been

instantiated, intended plan access can be invoked to recall the plan that was being used by the stranger.

Many "how" type questions are ambiguous. That is, "how" may request either a) the manner in which an act was performed, or b) the plan that was used to realize some event brought about by the act in question. The way in which a "how" question is answered, therefore, will depend upon the access route use into memory. For example, when people were asked:

Q: How did Karen talk to John?
A: By phone.

they tended to answer by supplying the plan attempted by Karen to achieve her goal of talking to Mary. However, when people were asked:

Q: How did John talk to Karen?
A: Abruptly.

they tended to answer by supplying the manner in which John talked in order to get Karen off of the phone. What is causing this difference in response? The answer lies with who initiated the event in the first place. Since the first question focuses upon Karen, and since Karen had the goal that initiated her call, it is relatively straightforward to access the plan she used to attempt to achieve her goal. On the other hand, when the focus is on John, search processes do not find that John had the goal of talking to Karen. Thus, the manner in which John talks to Karen is retrieved.

5. Goal Precondition Access

Given a plan, memory can retrieve a sub-goal whose achievement is necessary before the plan can be executed.

Q: What did John have to do before he could remind Bill about the money?
A: John had to get to the party.

6. Enabled Plan Access

Given a goal, memory can retrieve a plan serves the goal.

Q: Why did John want to be at home?
A: To be able to receive the stranger's demands.

Q: Why did Mary want to let John know about the kidnapping?
A: To get John's help.

7. Suspending Goal Access

When people remember a suspended goal, they can recall the higher priority goal which caused the suspension to occur.

Q: Why didn't John remind Bill about the $50?
A: Because John was worried about Susan.

8. Suspended Goal Access

When people recall a crisis goal, they can often recall those goals which had previously been active before their suspension by the crisis.

Q: How did the kidnapping interfere with John?
A: John didn't remind Bill about the $50.

9. Achieved Goal Access

Given an event, memory can retrieve the goal which that event achieved.

Q: What happened when Susan came home?
A: John was relieved.

Q: What happened when the phone rang at the party?
A: John found out about Susan's kidnapping.

Thus, when Susan came home, John's goal of saving Susan was achieved, and this resulted in his feeling of relief. Likewise, the phone call at the party achieves Mary's goal of telling John about the kidnapping.

10. Event Achievement Access

When people recall an achieved goal, they are able to recall the event that achieved that goal.

Q: Why was John relieved?
A: Because Susan came home.

Notice that this event can be recalled even though it was not caused by any plan of John's. This suggests that events achieving goals have direct associations with those goals.

11. Plan Realization Access

If an event was caused by a character executing some plan, then people are able to recall the plan which realized that event.

Q: How did John get home?
A: He drove.

Q: How did Susan arrive home?
A: She took the bus?

Q: How come the phone rang at the party?
A: Mary was calling John.

Sometimes people will have forgotten the plan although they remember that the event was caused by a successful plan. In these cases, people use reconstructive reasoning to derive a plausible plan, and they express uncertainty in their answer.

Q: How did John get to the party?
A: I'm not sure. He probably drove.

12. Realized Event Access

When people remember the plan that some character used in the story, they are able to then recall any event which came about as a consequence of that plan's realization.

Q: What happened when Mary called the party?
A: John received the call.

13. Thwarted Goal Access

If people recall an event which thwarted some character's goal, they will be able to recall that failed goal.

Q: How did John's lie affect Karen?
A: Karen was not able to talk to Mary.

14. Thwarting Event Access

If people recall a goal that failed, then they will remember the event which caused this goal to be thwarted.

Q: Why didn't Karen get to talk to Mary?
A: Because John told Karen that Mary wasn't home.

The following chart summarizes the 14 Standard Link Traversals:

ACCESS FUNCTION	INPUT	OUTPUT	LINK TRAVERSED
Motivated Goal	event	goal	motivates
Event Motivation	goal	event	motivated-by
Goal Intention	plan	goal	intended-by
Intended Plan	goal	plan	intends
Goal Precondition	plan	goal	enabled-by
Enabled Plan	goal	plan	enables
Suspending Goal	goal	goal	suspended-by
Suspended Goal	goal	goal	suspends
Achieved Goal	event	goal	achieves
Event Achievement	goal	event	achieved-by
Plan Realization	event	plan	realized-by
Realized Event	plan	event	realizes
Thwarted Goal	event	goal	thwarts
Thwarting Event	goal	event	thwarted-by

15. Event Involvement Access

Once an instantiated event node has been found, people are able to recall the characters that were involved in the event, and each character's role in that event. Once accessed, this information can be used by other processes which answer concept completion questions and check question presuppositions. For instance:

Q: Who was being transferred to California?
A: Bill.

Q: What did Bill lend John?
A: It was John who lent Bill money.

In the first question above, the role to be completed is TRANSFEREE, while in the second question above, the role bingings for the LENDER and BORROWER in the question violate their bindings in story

memory.

16. Next Event Access

Given an event node, memory can recall an event which occurred subsequently. For instance:

Q: What happened after John arrived at the party?
A: Mary called him.

Q: What happened after John arrived home?
A: Karen called.

Q: What happened next?
A: John told Karen Mary wasn't home.

What complicates the use of Next Event Access is the fact that "What happened after ..." and "What happened next?" are ambiguous questions. Such questions request that memory recall some INTERESTING or salient subsequent item, rather than just any minor event immediately following the event given. Such an item of interest might include: emotional reactions to event, new goals created by the event's occurrence, what actions a character was performing within the given event,etc. Each of these alternatives is handled by other search processes, and therefore, will be discussed later. Also, in a story with multiple characters, there may be several "next" events. Next Event Access, however, searches only for the next immediate event effecting the main character. Therefore, "What happened after Karen called?" will not focus upon what happended next to Karen, but upon the next event as mentioned in the story.

17. Prior Event Access

This access rule complements rule 2. That is, it retrieves the preceding event which lies along the "story line" -- i.e. the sequence of events as told during story understanding time. For example, in the question below, Prior Event Access produces answer A1:

Q: What happened before Susan came home?
A1: John and Mary discussed calling the police.

rather than the following alternative answer:

A2: Susan attended a gym meet.

As with rule 2, Prior Event Access is not sensitive to character perspective. Therefore, this access rule produces A1 instead of A2 below:

Q: What happened before Karen called?
A1: John came home.
A2: I don't know what Karen was doing.

Prior Event Access is also used to handle time questions. For example, answering "When did John leave the party?" involves (among other things) accessing the prior event of Mary's calling, and then referencing this event in the answer: "After Mary called."

18. Main Character Access

While reading a story, people keep track of which character's perspective is dominant. For stories with only one dominant perspective, that character becomes the "main" character in the story. This character can be recalled directly, and is used in the following types of questions:

Q: Who was the story about?
A: John.

Q: Who was the main character?
A: I don't remember his name, but he was the father of the kidnapped girl.

Several search processes rely upon an ability to recall the main character. For example, actor completion questions ask us to recall the character who performed a given action. If the act mentioned is primitive, then memory has to recall a character in the story and see if that character did (or could have) performed the act in question. In these cases, the search starts with the main character.

Another use of Main Character Access shows up during the task of generating the answer to definitional questions. For example, when people are asked:

Q: Who is Mary?
A1: John's wife.

they answer with A1 above rather than with the following alternative:

A2: Karen's friend.

even though A2 seems equally descriptive. A2 is not given because John is the main character while Karen is not. More information is imparted by giving Mary's relation to the main character, rather than her relation to some minor character.

19. Relational Information Access

Once people have recalled a character in the story, they are able to remember basic personal information about the character, such as sex,

name, rough age, and so forth. This information is used in presupposition checking, to make sure that the question makes sense in the context of the story.

The interpersonal relationships that character is involved in are very important. These relationships are central organizing features because many goals are motivated by interpersonal issues.

Q: Who was Susan?
A: The daughter.

Q: Who was the friend of John's wife?
A: Karen.

Q: Why did John hate Susan?
A: He didn't. John was Susan's father.

Susan can be characterized uniquely as "the daughter" since she is a unique member of the only family structure in the story. In the last example above, memory processes use this information to check the compatibility of attitude with a father-daughter relationship.

Q: What did John want?
A: To save Susan.

Q: What did Bill want?
A: I don't know. To have a good time.

Q: What did Karen want?
A: To talk to Mary.

In the examples above, the most salient goal is recalled. In Bill's case, since there was no mention of any of Bill's goals in the story, what is recalled is reconstructed from general knowledge about why people attend parties. (Reconstructive recall will be discussed later.)

20. Major Goal Access

Some goals are more important than other goals. For example, crisis goals will be recalled first, especially when little context is given.

Q: What did John want?
A: To save Susan.

Q: What did Bill want?
A: I don't know. To have a good time.

Q: What did Karen want?
A: To talk to Mary.

In the examples above, the most salient goal is recalled. In Bill's case, since there was no mention of any of Bill's goals in the story, what is recalled is reconstructed from general knowledge about why people attend parties. (Reconstructive recall will be discussed later.)

21. Relational Goal Access

Interpersonal relationships are important in the story because they help to organize the characters' goals. Therefore, goals which directly arise from a relationship (or effect the maintainance of a relationship) are associated with that relationship and can be retrieved as such. For example, the fact that Bill borrowed money from John is understood in the context of their friendship. If Bill doesn't repay John, this could damage their relationship. Therefore, the goal of being repaid is associated with their relationship.

Q: What did John want with Bill?
A: To get his money back.

22. Scenario Containment Access

Characteristic locations which serve as settings for one or more events are instantiated in episodic memory as scenarios. Scenarios are memory nodes which organize whatever events occurred within that setting. Whenever people hear of an event occurring within a given setting, they tend to associate that setting with the event. In such cases, the setting can be retrieved from the event.

Q: Where was the gym meet held?
A: At school.

Q: Where was the party?
A: I don't know.

Scenarios may or may not have locational information associated with them. For example, the party serves as a scenario for both the party event and Mary's phone call. However, the actual location of the party is not known. Some events, such as phone calls, serve to connect one scenario to another. In such cases, a decision has to be made as to which scenario to access. This will depend upon which character is in focus. For example,

Q: Where was Mary when she called John?
A: At home.

Q: Where was John when Mary called him?
A: At the party.

In the above examples, each scenario is accessed according to the character it contains.

23. Event Containment Access

People commonly retrieve events by first recalling a scenario in the story, and then by using this scenario to recall the events which occurred within it. If the number of events within a scenario is large (e.g. more that 3 or 4), then usually the first event within that scenario will be recalled. In order to retrieve the remaining events, other search processes must be invoked. (See section 4.3.)

Q: What happened at the party?
A: John got a call from Mary.

Q: What happened at home?
A: Mary got a call from a stranger.

Q: What happened at school?
A: Susan attended a gym meet.

24. Delta Goal Scenario Access

Since scenarios organize events, it is important to know whenever a character changes settings. In addition, characters usually have a reason for changing setting. At the lowest level, this reason must include a D-PROX goal. The D-PROX goal usually serves to satisfy some precondition within a character's plan for achieving a higher goal. For example, John can not play his role of party guest until he is actually at the party. Thus, D-PROX [party] enables PLAY-GUEST [John]. If the higher level goal is not known, then only the d-prox goal will be recalled. Since this does not usually convey much information, it may sound funny. This is the basis behind the humor in:

Q: Why did the chicken cross the road?
A: He wanted to be on the other side.

D-prox goals are organized in the Scenario Participant Map (see section

3), where they are indexed by character and scenario. Given a character and a source or destination scenario, people can access the associated d-prox goal and use this to recall higher level plans and goals.

Q: Why did John go to the party?
A: Because John wanted to get to the party.

Q: Why did John leave the party?
A: Because he wanted to get home.

Q: Why did Bill leave his home?
A: Because he wanted to get to the party.

The answers above are similar to the answer to the chicken joke insofar as they are too low level. It is up to other search heuristics to realize when an answer is too low level (see section 4.3).

25. Scenario Contact Access

When a character in one scenario communicates with a character in another scenario, the link across these two scenarios is maintained in the Scenario Participant Map as a D-Know goal on the part of the character which initiated the communication. This d-know goal can then be recalled by means of the Scenario Contact Access, which searches the scenario participant map.

Q: Why did Mary call at the party?
A: She wanted John to know about the kidnapping.

Q: Why did the stranger call John's home?
A: He wanted John and Mary to know about their daughter's abduction.

Thus, scenarios act as one way of discriminating one phone call from another.

26. Main Event Access

When a question supplies no context, search processes seeking an event must have some available node in memory from which to proceed. Usually this node will be the most salient event in the story (if there was one).

Q: What happened?
A: There was a kidnapping.

Q: What was the story about?
A: John's daughter was kidnapped.

4.3. HEURISTIC SEARCH PROCESSES

Heuristic Search Processes rely on both instantiation recognition and primitive access functions. There are 18 search heuristics which can be categorized into four types of mechanisms: (1) **Scans**, (2) **Derivations**, (3) **Checks**, and (4) **Searches**.

A **Scan** is usually a constructive process (with the exception of the Goal Feature Scan) which compiles a list of entities and examines them in a serial fashion. Four scanning heuristics have been implemented for BORIS.

A **Derivation** searches multiple instantiations in order to find one which is the most likely referent. There are three Derivations which correspond to the three techniques of Instantiation Recognition described in section 4.1. When unique referents are available, Instantiation Recognition is sufficient. But when ambiguous references are encountered, a Derivation must be invoked.

A **Check** is a process which examines memory in order to detect false

presuppositions in a question. Three Checks have been implemented thus far. Checks are executed as the question is being understood, as a technique of integrated memory access.

A **Search** is a process which provides a flow of control for multiple access functions. There are 8 Searches that have been implemented in BORIS. These functions are built on top of Instantiation Recognitions, Derivations, Primitive Access Functions, and Scans.

1. Relationship Scan

Protocols indicate that people do not maintain a mental "list" of all the characters in a story. Subjects cannot just rattle off names or descriptions of every character, and they will inadvertently omit one or more characters during this task. Therefore, people must be relying on some method of character construction. One such method is called a Relationship Scan. It involves a traversal of the Relationship Map, and starts with the main character unless a specific character has been supplied by the current context.

Description:

For each character recalled, invoke a Relational Information Access to retrieve the interpersonal relationships this character is involved in. Do this for each additional character recalled until no new characters are recalled.

Example:

Q: How many characters were there in the story?
A: Five: John, Mary, Susan, Bill and Karen.

John is recalled first by means of Main Character access. Then Relational Information Access on John leads to recalling both his family

and his friendship with Bill. Relationship Information Access is then applied to each family member and friend. Since Mary has a friendship with Karen, Karen is recalled. Once no new interpersonal relationships are found, the process stops.

Actually, there are 6 characters in the story. People commonly forget to mention the stranger when performing this constructive task. Whenever the stranger is mentioned he is usually recalled last. This indicates that he is being retrieved by means of some different search heuristic.

We have conducted some experiments on the task of character listing which indicate that the Relationship Scan is a central mechanism in deriving a list of characters from a narrative. For example, two groups of subjects were asked to read two versions of a narrative, where one version specified two characters as brothers, and the other version specified the relationship between the same two characters as one of friendship. The two versions were identical in all other respects. The resulting character lists showed a statistically significant difference in the distance between these two characters across the two groups. Lists resulting from the "brother" version placed these two characters closer to each other than the lists resulting from the "friend" version. This phenomena was then duplicated with a second narrative designed to control for other contributing factors. This result suggests that character lists are generated by an ordered expansion from the main character via Relationship Information Access. The Relationship Scan is designed to model this process of constructive search.

2. Goal Feature Scan

Sometimes an instantiation process returns more than one goal. In such cases, any additional information in the question can be used to to decide which goal is the one being referenced in the question.

Description:

If the question concept indicates the type of goal being sought, then only consider goals of this type. Select the goal whose features match the features described in the question. If the question specifies an act or plan, then return a goal which either was achieved by a plan of that type, or (for goals not yet achieved) could have been achieved by a plan or act of that type.

Example:

Q: Why did John want money from Bill?
A: Because Bill had borrowed $50 from John.

Q: Why didn't John talk to Bill?
A: Because he was worried about Susan.

In the first example, Relationship Goal Search has returned two goals: a) John's goal to say farewell to Bill at the party, and b) John's goal to get money back from Bill. Since the question mentioned money, this information is used by the Goal Feature Scan to select the appropriate goal.

In the second example, Goal Feature Scan tries to select a goal of John's which could have been achieved by John having MTRANSed something to Bill. In this case, both John's D-CONT [money] and preserve friendship goals satisfy this criteria. Thus, it is up to Interference Search to select the goal which was suspended and return the suspending goal. (In this case, however, both goals were suspended by the same crisis goal, so the choice does not matter.)

3. Scenario Events Scan

This process is used to retrieve all of the events which happened within a given scenario. If the number of events is small (i.e. less than four) then they will be directly associated with the scenario and can be recalled directly through an Event Containment Access. Otherwise, all of the events must be constructed by searching the chain of events occurring in the story and selecting those contained by the given scenario. For example, many things happened at John's home. Thus a Scenario Events Scan would be used to answer the following question:

Q: What events occurred at home?
A: The stranger called Mary and then Mary called John.
John arrived and then Karen called.
John and Mary discussed calling the police and
then Susan arrived.

When only a single event is requested, then the first event found that's associated with the scenario will be retrieved. Which event is recalled first may depend upon the context of the question. For instance, if the context is John's arrival at home, then the event recalled first will be Karen's call instead of the stranger's call. (See section 5 for a discussion of the influence of context.)

4. Event Chain Scan

This process is invoked whenever a list of the events following a given event must be constructed. The construction is done by successively calling Next Event Access. If Next Event Access fails to return the next event (possibly due to forgetting), then Motivated Goal Access, Intended Plan Access, and Realized Event Access are used to find an event which might have occurred because of the last event retrieved.

The scan terminates once no subsequent event can be recalled.

Example:

Q: What events happened after Mary called John at the party?

A: John arrived home.
Karen called.
John and Mary discussed calling the police.
Susan arrived home.

5. Character Derivation

Whenever Name Instantiation fails, people either give up on the question, or retrieve a character compatible with the name mentioned.

Description:

Use a character construction strategy (such as Relationship Scan) to build a list of the characters in the story. Then scan this list, using Relational Information Access to find a character whose description is compatible with the information inferable from the name mentioned in the question. If a compatible character is found, then the name mentioned becomes associated with the character so that subsequent Name Instantiations will succeed.

Example:

Q: What did Karen want?
A: To save her daughter.

If Name Instantiation on "Karen" fails, then Character Derivation invokes A constructive strategy (such as Relationship Scan) to recall the characters in the story. Since Mary's name has been forgotten also, and since her sex is compatible with the default sex of the name "karen", Character Derivation mistakenly assumes that "Karen" refers to the character Mary in the story.

This kind of error is commonly made by people since they are

willing to answer questions in spite of name confusions.

6. Event Derivation

This search is invoked whenever Event Instantiation Access fails. Such failures occur when the script mentioned in the question concept is not a unique event in the story. The nature of the search will depend upon the type of script and upon any additional information available in the question. For example, if a scenario is mentioned in the question, then a Scenario Events Scan will be invoked to recall the events that occurred within that scenario. If one of these recalled events matches the script in the question, then Event Instantiation will succeed. For example, the phrase, "the call at the party", would cause Event Instantiation to retrieve events which occurred at the party. Since one of these was a phone call, that phone call would be instantiated rather than, say, Karen's call, which is associated with the home scenario.

When a scenario is not given, but the initiator of the event is mentioned, then a search of that actor's goals can often lead to a unique event in memory which matches the script mentioned in the question concept. For example, "the call at home" is ambiguous because there were two calls: the stranger's and Karen's. If Karen is mentioned as the actor then a goal search can find Karen's D-KNOW goal, which will lead to a unique phoning event through plan intention and event realization links.

7. Scenario Derivation

Normally, scenarios are instantiated by accessing the events contained within them. For example, the party event makes the party

scenario immediatedly available. However, in those cases in which a scenario is not uniquely referenced, other heuristics must be employed. These heuristics are usually associated with the type of reference mentioned. For example, the word "home" refers to a scenario involving the interpersonal relationship structure of R-FAMILY. Thus the reference to "home" can be found by noticing which character has been mentioned who is a family member. This search heuristic explains why people always assume that "home" in the kidnap story always refers to John's home. It is because there is only one family mentioned explicitly in the story. For example, when people are asked:

Q: What happened at home?

or even:

Q: What happened to Karen at home?

they claim that they think of Mary's home first. Some people even assume that "Karen" refers to the character Mary, even though they have successfully answered prior questions that refer to Mary as the wife.

8. Scenario Compatibility Check

This process is invoked by questions involving actions between two characters which have scenario prerequisites. In such cases, it checks to make sure that the presuppositions concerning the characters whereabouts are compatible with the scenarios the characters were members of.

Description:

If the question refers to an MTRANS between two characters

within a given scenario, then search the Scenario Participant Map to make sure that both characters participated in that Scenario. If the question refers to a character involved in an event, then use Event Containment Access to recall the scenario containing that event, and check the Scenario Participant Map to see if this character was involved in this scenario. If not, reject the question, and for verification questions, use this rejection as the elaboration in the answer.

Examples:

Q: Did John see Susan at school?
A: No, John was never at school.

Q: Did John talk to Mary at the party?
A: No, Mary was never at the party.

In the first example above, the question is rejected since John was never at school. The second question is interesting because the answer to it is wrong. John did talk to Mary at the party, but it was over the phone. However, people commonly answered "No" for the same reason that Scenario Compatibility rejected this question. Unless people recall Mary's phone call, they notice instead that "talking" requires the two participants to be in the same scenario. A quick search of the Scenario Participant Map reveals that Mary never was at the party, so the question is rejected on these grounds.

9. Role Compatibility Check

This process is invoked whenever a question refers to an event involving one or more characters. Once an instantiation search has found the corresponding episode, Role Compatibility Check compares the role bindings in the question against the role bindings in the episode to see if there have been any presupposition violations. If so, the question is rejected and the correct role bindings are used in the response.

Examples:

Q: Why was there a party for John?
A: John was not the guest of honor.
Bill was.

Q: Who transferred Karen to California?
A: Karen was not transferred to California.
Bill was.

In both cases above, the answer given does not address what the question asks. Instead, the violated presupposition is mentioned, along with the corrected role binding. This is reasonable since people also quit trying to answer the question once they start correcting the false presuppositions.

Verification questions are handled by those mechanisms which check for violated presuppositions. The major difference is in generation of the English response. In the case of violated presuppositions, these violations must be pointed out. In the case of verification questions, however, the question is simply answered in the negative if the verification has failed, and the reason for the failure may be generated as an elaboration to the negative.

10. Thematic Compatibility Check

This process is invoked by questions involving certain actions between two characters. The action must have a strong positive or negative consequence for the object of the action. In such cases, the Thematic Compatibility Check accesses the relationship between the characters involved and compares this against the default thematic relationship implied in the action.

Examples:

Q: Did John kidnap Susan?
A: No. He's her father.

Q: Did John hate Bill?
A: No. They were friends.

In the first example above, the act of kidnapping implies a strong negative thematic relationship between the kidnapper and the victim. Therefore, the Thematic Compatibility Check examines the relationship between John and Susan. It discovers that they are father and daughter. Without specific contradictory information, it assumes that fathers love their daughters. This is a strong positive thematic relationship. Therefore a contradiction is found, and the relationship is used as the elaboration response to the question. Notice that this process can operate without having instantiated the kidnapping in memory, which means that is may be done more quickly than other processes which are also operating. For instance, another process which will be active is Role Compatibility Check. Once the kidnapping has been instantiated to a node in memory, the roles in the question can be checked against those in the event. Once this occurs, the response generated is:

Q: Did John kidnap Susan?
A: No. It was the stranger that kidnapped Susan.

The vast majority of people we sampled responded "No. He was her father." In addition to supporting the description given above, this data also supports the view of "Rational Recall" presented in section which argued that the search processes used to find and answer are sensitive to the reasons for why an answer is correct or incorrect. With these cases, the way the answer to the verification was

found also immediately supplied a reason as to why the answer found was reasonable.

11. Interference Search

This search heuristic is invoked by question which ask "Why didn't...?" For example, "Why didn't John get his money back?" The processing of this question assumes that the character did not actually perform the action, but had intended to perform it at some point in the story.

Description:

> Interference Search first invokes a goal search in order to recall the goal underlying the intention to perform the act. The nature of this search depends upon the nature of the act, and is handled by multiple search heuristics. Once this goal has been found, it is examined to see if it was thwarted, suspended, or achieved. If it was achieved, the question is rejected for harboring a false presupposition. If the goal was thwarted, Thwarty Event Access is used. If the goal was suspended, Suspended Goal Access is invoked. Answers to reasonable interference questions will derive from either thwarted or suspended goals.

Examples:

Q: Why didn't Karen get to talk to Mary?
A: Because John told her that Mary wasn't home. (thwarted)

Q: Why didn't John talk to Bill?
A: Because he was worried about Susan. (suspended)

Q: Why didn't John go to the party?
A: John did go to the party. (rejected)

In the first example above, Karen's goal of talking to Mary was thwarted by John. The event was John's act of MTRANSing to Karen that Mary wasn't home. Since the thwarting was caused by another character,

an alternative answer could have involved producing the goal behind John's act. This would have caused the response

A: Because John wanted to keep the phone free.

This answer could be found by executing Achieved Goal Access after Thwarted Event Access.

In the second example above, once John's goal of D-CONT [money] is retrieved, Interference Search can recall the P-HEALTH crisis goal which suspended it. In the third example above, John's D-PROX [party] goal is found. However, Interference Search discovers that it was achieved, so the question is rejected.

In previous question-answerers at Yale [Lehnert 78], interference questions were handled by following a "Ghost Path" in a script. An example was Jack's going to a restaurant and ordering a hamburger. The hamburger was burnt, so Jack ordered a ham sandwich. The script included two paths (or tracks). In one, Jack would have eaten the hamburger since this is what normally happens when hamburgers are ordered in restaurants. When the system was asked:

Q: Why didn't John eat the hamburger?

it checked for default paths in the script in which the hamburger would've been eaten. It then backed up to the choice point which had caused the alternative path to be taken. This choice point in the script indicated that the alternative was taken when the food served had been burned. Thus, the system could respond:

A: Because it was burnt.

There are several problems with using this approach in a more complex story domain. First, there are many things that can go wrong or happen unexpectedly in a more complicated story. To handle this in terms of Ghost Paths would cause an enormous proliferation of alternate tracks in each script. Second, one script might simply serve as a setting for some other event which is actually the cause of the interference. Finally, the Ghost Path notion was used to avoid the need for directly representing the goals and plans of the character being effected within the script. But in a system of multiple knowledge structures, the interactions between settings, events, and the characters' effected goals must be represented explicitly. These representational innovations allow us to design a more general retrieval heuristic - which is not structure-dependent.

12. Higher Goal Search

Whenever a delta goal is found by some search process, Higher Goal Search may be invoked. Delta goals usually serve as enablement conditions on plans being executed to achieve more important goals.

Description:

Whenever a delta goal is returned, use Enabled Plan Access to retrieve the plan which the delta goal enabled. If there is such a plan, then use Goal Intention Access to recall the higher goal which the plan was intended to achieve. If there is such a goal, then return it, otherwise return the delta goal.

Examples:

Q: Why did Bill go to the party?
A: He wanted to enjoy himself.

Q: Why did Karen call Mary?
A: She wanted to tell Mary something.

In the first example, we are given Bill's d-prox goal. Then Higher Goal Search finds that Bill's reason for being at the party is to enjoy the party. In the second example, we get Karen's d-know goal of telling mary something from the $phone script. However, no higer goals are found, for Karen so the d-know itself is returned.

13. Goal Search

This search is invoked by both Interference and Causal Antecendant questions. It is up to Goal Search to find one of more of the goals which the question is implicitly referencing. The nature of the search will depend upon the kind of information available in the question.

Description:

If the question concept involves two characters, and the action between them is too "low-level" to use for event instantiation, then Relational Goal Access is invoked on the character performing the act. If, however, the question involves a scenario change, then Delta Scenario Goal Access is invoked, followed by Higher Goal Search, in order to find the goal(s) which motivated the character to change scenarios.

Examples:

Q: What did John want with Bill?
A: To see Bill off, and to get the $50 back.

Q: Why did John leave the party?
A: Because he was worried about Susan.

In the first example above, all we know is that John has a goal and that it involves Bill in some way. Since there is no act or event to be instantiated in the question, Goal Search examines John's relationship with Bill to see if John had any goals involving this relationship. In the second example, the party scenario is given, so Goal Search can use the Scenario Participant Map (via Delta Scenario Goal Access) to find

John's goal of going home. This D-PROX [home] is then used by Higher Goal Search to find the reason why John wanted to be home.

14. Actor Completion Search

Questions which seek the actor of a goal (or state) are generally more difficult than questions which seek to specify the object. When a question mentions an actor, then his goals and plans can be retrieved to aid in searching for an answer. With actor completion questions, however, this information is not directly accessible. For instance, Q1 below is harder for people to answer than Q2:

Q1: Who wanted money from Bill?

Q2: Who did John want money from?

This can be explained by assuming that, in order to answer Q1, people simply search John's goals for a goal involving a D-CONT [money]. In Q1, however, a search of Bill's goals will not be helpful since John is not available. In such cases, unless an instantiation search locates the goal, plan or state in the question concept, search processes must resort to a "generate and test" mode. This is what Actor Completion Search does. In other words, it uses a character construction search to recall the various characters in the story and then searches each character's goals for a match against the goal in the question. This seems analogous to the way people answer such questions. That is, people answer Q2 by doing something like the following:

1 Who might have wanted money from Bill?
2 Did Mary want something from Bill? --> No.
3 Did Karen want something from Bill? --> No.
4 Did John want something from Bill? --> Yes.

Although the example above gives the "flavor" of the process involved, the actual process to handle Q2 will find the answer much faster than shown above. A character construction search will start looking for characters by searching the interpersonal relationships Bill is involved in. Thus John will be the first candidate whose goals are tested. This makes sense, since (all things being equal) the most likely characters to have a goal involving Bill will be characters which have some relationship already established with Bill.

15. Affect Reaction Search

This process is invoked whenever a question refers to a strong positive or negative affective reaction on the part of a character. If the affect is positive, then the search looks for an important goal of the character that was achieved and recalls the event that achieved it. If the affective reaction is negative, then crisis goals are sought.

Examples:

Q: Was Mary upset?
A: Yes. She thought her daughter had been kidnapped.

Q: Why was John happy?
A: Because Susan came home.

Of course, context will effect the search. If Susan's arrival home was the last event mentioned, then "Was mary upset?" will be answered with a "No". (See section 5 for a discussion of the role of context.)

16. Expectation Violation Search

Some questions directly refer to violated expectations. For example:

Q: What surprised John at the party?
A1: Mary called about the kidnapping.
A2: That Susan was kidnapped.

Q: Who did John expect when Karen called?
A: The kidnapper.

difficulty recalling violated expectations out of context. To recall such events, they must resort to an Event Scan of the story, all the while checking each scenario or event to see if something unexpected had occurred at that point in the story. There are several heuristics which the Expectation Violation Search employs to recognize such events. One is to check the scenario in the question to see if the character in the question was involved in an event which he did not have a plan for, and which he was not the initiator of. Answer A1 above is handled in this way, since people who receive phone calls do not usually expect them. Another heuristic involves looking for suspended or thwarted goals, since the suspending goal is usually not anticipated (or it would have been taken into account during planning). Answer A2 above is handled in this way. Since John's current goal of reminding Bill about $50 was suspended by a goal to save Susan, the event which caused the suspending goal is retrieved.

Yet another heuristic involves fortuitous goal achievement. This occurs when a character has a goal which is achieved by an event not realized by any of the character's plans. This is how the system can infer that John is surprised when Susan comes home.

Does goal suspension really have anything to do with violated expectations? Some events are surprising whether they cause goal suspension of not. For example, events which cause crisis goals are

surprising to characters who learn about them -- even if those characters are not directly involved. Notice that the second question is of a totally different order than the first. Here, it must be realized that: a) John had the goal of hearing the demands, b) that Karen's call threatened an enablement condition on this goal. Thus, we could attempt to handle the second question by examining goal threats. However, it is more than the threat on keeping the phone clear. The moment the phone rings (and before we hear it is Karen) we are expecting the stranger. This is a very strong expectation, and its violation should be represented explicitly, whether or not it threatens some other goal of John's. At present, the system is not capable of handling expectation violations which are not linked to goal suspension or fortuitous goal achievement.

17. Scenario Plan Search

Some questions request the actions performed by a character within a scenario. Such actions are usually part of some plan. Plans are represented in the system in two ways: a) as acts in the service of a goal, or b) as a script role (in the case of stereotypic activities). For instance, playing a guest role at a party is actually a plan for how to act as a guest.

Examples:

Q: What did John do at home?
A: He talked to Karen on the phone and he argued with Mary about calling the police.

Q: What did John do at the party?
A: He mingled, and he talked to Mary on the phone.

In each of the above examples, the activities performed by John can

be retrieved from his role in each event. That is, a guest mingles, a receiver of a call talks to the caller, etc. Scenario Plan Search, therefore, uses Event Containment and Event Involvement Access to recall the role bindings of the character in each event. The activities associated with the role is then recalled.

18. Causal Antecedent Search

This process is invoked by question concepts of the form: (LEADTO ANTE (*?) CONSE X). If memory search instantiates X to a character's goal, then Event Motivation Access is used to recall the event which gave rise to this goal. If X instantiates a character's plan in memory, then Goal Intention Access is used to retrieve the goal which gave rise to this plan. If X refers to an event, then the actor in the event is retrieved since this will usually be the character who initiated the event. The initiator's plan (i.e. role) can then be used by Goal intention Access to recall the character's reason for having played this role in the event.

Examples:

Q: Why did John want $50 from Bill?
A: Bill had borrowed $50 from John.

Q: Why did Mary call John?
A: To let John know about the kidnapping.

Q: Why was there a party?
Al: Because Bill's friends wanted to see Bill.

In the first example, John's d-cont [$50] is used to recall the motivating event, which was the prior $borrow. In the second example, Mary's plan of calling John was intended to achieve her goal of telling John about the kidnapping. The last example is more difficult. The

party was thrown by Bill's friends. Their reason is to preserve their friendship with Bill (who they won't be seeing as much of). However, A1 is not as good an answer as:

A2: Because Bill was being transferred to California.

Although A2 is a better answer, it is a more difficult answer to come up with since it involves answering a question about one event by mentioning yet another event. That is, the party event came about because of a plan to throw a party. This plan (of Bill's friends) was executed to maintain contact with Bill. Why do relationships need to be maintained? Because something "threatened" these relationships. This "threat" came from Bill's job transfer. But what tells memory that Bill's job transfer is a better answer than intermediate goal of seeing Bill at the party? The solution to this problem lies with the kind of party it is. The story specifically referred to the party as a "farewell party". Thus, there is information directly associated with this kind of script, which essentially says that it was brought about by a transfer of some sort. With this additional information, the transfer event can be sought by accessing the motivation behind the relationship-preservation goal of Bill's friends.

This kind of knowledge about parties in general (or farewell parties specifically) is closely intertwined with the episodic knowledge about the actual party event in story memory.

4.4. RECONSTRUCTIVE RECALL

In general, there are several possible outcomes which can result from attempts to answer questions about a story:

- An answer is found in the story representation
- No answer is found, resulting in an "I don't know" response.
- No answer is found, so reconstructive recall is invoked, and general memory is searched to find a plausible answer.

Consider the following answers to the question below:

Q: Why did John go to the party?

A1: To remind Bill about the $50.
A2: I don't remember.
A3: Probably to have a good time.
A4: Probably to say "goodby" to Bill.

Answer A1 requires episodic memory access. There is no way of knowing this answer without knowing the story. Answer A2 simply indicates that episodic memory search has failed. Answer A3, however, could have been answered without even having read the story. People generally go to parties to have a good time. So A3 could have been generated soley via reconstructive search. Answer A4 is the most interesting answer because it involves an **interaction** between both episodic and reconstructive access. That is, answer A1 is not possible without retrieving the party mentioned in story memory. But once this has been done and it has been recalled that Bill was the guest of honor, the reason (i.e. saying "goodby") can be generated reconstructively from what we know about **farewell** parties in general. That is, people go to farewell parties to wish farewell to the guest of honor. Therefore, A4 involves a combination of episodic and reconstructive search.

Whenever episodic search fails, reconstructive search processes could be invoked to find a plausible answer, using general memory structures. In these cases, the English response would contain the word

"probably" within it to indicate that reconstructive search had been used. For example:

Q: Did John mingle with people at the party?
A: Probably. That's what guests do.

Q: Did John talk to Susan when she got home?
A: Probably. That's what family members do.

Reconstructive retrieval has not implemented in the current question answering system for BORIS, although it is a target problem area for our next implementation. For a discussion of reconstructive retrieval techniques in a expert retrieval system see [Schank and Kolodner 79]. Information retrieval in an expert system is very different from information retrieval in a narrative understander, but the problems of reconstructive memory across these two task orientations share a number of similar issues.

5. THE ROLE OF CONTEXT

People exploit the context of question-answering dialogues whenever they respond to a question. They use context both to aid their understanding of a question, and to direct the processes which search memory for an answer. We tend to be conscious of these contextual factors when the dialogue presents questions in a funny order:

Q: Why did Mary call John?
A: Because of the kidnapping.
Q: Did Mary call John?
A: Yes.

When questions are presented in a reasonable order, we sense a contextual continuity:

Q: Did Mary call John?
A: Yes.
Q: Why did Mary call John?
A: Because of the kidnapping.

This continuity can be explained in terms of two contributing process issues: (1) knowledge state assessment which builds a model of the questioner's knowledge state, and (2) context-sensitive memory retrieval mechanisms. We will briefly describe a few of the context sensitive factors that direct retrieval.

5.1. USING CONTEXT TO UNDERSTAND

Questions do not usually appear in isolation. Rather, questions are asked in the context of scriptal interactions, conversations, and other, more restricted, "test" sessions. In fact, many questions cannot be understood without relying on the previous question-answer pair. For example:

AD-A084 141 YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE F/G 5/7
MEMORY ORGANIZATION AND SEARCH PROCESSES FOR NARRATIVES,(U)
APR 80 M G DYER, W G LEHNERT N00014-75-C-1111
UNCLASSIFIED RR-175 NL

Q: Why did John leave the party?
A: He was worried about Susan.
Q: Why?
A: He thought she had been kidnapped.

Here, the question "Why?" makes no sense without reference to the answer before it. To handle this situation, the system makes use of the following contextual heuristic:

1. Previous Answer Reference

When the system parses a question to the form: (LEADTO ANTE (*?*) CONSE X) where the consequent X is not specified, it then accesses the answer to the previous question. If the previous answer is an event, goal, or plan, then the system hands the question to memory processes as usual to search for an answer.

Notice, however, that this scheme will **not** work in the following case:

Q: Who was the party for?
A: Bill.
Q: Why?
A: He was being transferred to California.

since Previous Answer Reference would pass the following structure to the memory search:

```
(LEADTO ANTE (*?*)
        CONSE (PERSON NAME (BILL)
                      SEX (MALE)
                      I-HUM (BILL0)))
```

But a question of the form: "What caused Bill?" makes little sense. Therefore, if the previous answer is anything other than a goal, plan or event, the following heuristic is used:

2. Previous Question Reference

Both the previous question and answer are accessed, and if the previous question was a concept completion type of question, then the previous answer is used to complete the previous question. This new structure is then bound to the consequent slot in the LEADTO structure, and this new question is passed to memory search processes. Thus, occurring after:

Q: Who called John at home?
A: Karen.

"Why?" would basically be understood as:

```
(LEADTO ANTE (*?*)
        CONSE ($PHONE CALLER KAREN0
                      CALLEE JOHN0
                      LOC  HOME0))
```

Using these two heuristics, many question fragments can be understood by referencing the most recent question-answer pair.

5.2. USING CONTEXT DURING RECALL

People also seem to use the current memory context to aid subsequent search. The most obvious example occurs when people answer "What happened next?" This question essentially tells us to resume the memory search for subsequent events at the point in memory where we last left off. A similar version of this occurs in many exchanges that are somewhat more subtle.

For example:

Case I:

Q: Why did John leave the party?
A: He was worried about Susan.
Q: How did Mary feel?
A: She was hysterical.

Case II:

Q: What happened when Susan came home?
A: John was surprised and relieved.
Q: How did Mary feel?
A: She was relieved also.

In case I, the answer to "How did Mary feel?" is very different from the answer in case II although the questions are identical. Obviously, the current memory context in the second case refers to the event at the end of the story, when the hoax is discovered, while the first case has a different context. Likewise, people are good at automatically supplying a context when a reference is left ambiguous. For instance:

Case I:

Q: What did John want from Bill?
A: $50
Q: What did John do after the call?
A: He rushed home.

Case II:

Q: Why did John lie to Karen?
A: He wanted to keep the phone free.
Q: What did John do after the call?
A: He argued with Mary about whether to call the police.

We get very different answers to the same question: "What did John do after the call?" Here, "the call" is ambiguous since it could refer

to any one of a number of calls. Yet people do not notice this ambiguity. They automatically supply a context and then use this context to aid in locating an instantiated phone call. The establishment of context should be a natural side-effect of previous memory searches. This way it will always be available as a constraining factor to delimit potential ambiguities and narrow the possible search space for a question.

6. CONCLUSION

It is easy to write a computer program that makes mistakes. What is difficult is the design of a system that makes the same mistakes people make. But could we design a system that never made mistakes? If we could do that, why should we bother with a design that does no better than a person? The answer is methodological.

Very little research has been conducted on information retrieval from narrative texts. Given the complexity of the memory representations needed, it is not at all clear how one should proceed to build a "perfect" system. For all we know, perfect systems might be theoretically impossible (there are no perfect human systems to refute this possibility). But we do know that slightly imperfect systems are possible (people exist). So the only strategy open to us is a simulation of human information processing. If we could build an accurate simulation of human retrieval processes, we would be in a much better position to examine the question of whether or not a perfect system is possible.

The mistakes that people make and the preferred responses they supply should reveal something about underlying cognitive processes. So we have studied and emphasized actual human responses in order to better understand the information processing needed. We have not yet specified a complete process model for narrative question answering, but we have specified a preliminary set of processing fragments which are needed to account for various examples of question answering behavior. We must now piece these fragments together with an overall flow of control to

produce a complete process model. We have a set of building blocks; we can now build something.

To build a model that simulates human processing, we must study human question answering behavior with special care. Actual question-answering data can reveal important clues to the overall process.

For example, subjects tend to provide the following response:

Q: Why did John lie to Karen?
A: Because he wanted to keep the phone clear.

In order to get this response, we must traverse the immediate plan of terminating the phone call and arrive at the intended preservation goal of keeping the phone clear. This can be achieved by executing Goal Intention Access (he wanted Karen to think Mary was out) followed by Goal Precondition Access (he wanted to end the phone call) followed by Goal Intention Access (he wanted to keep the phone clear).

But how general is this search strategy? Which why-questions can be handled by this heuristic? Suppose we wanted to answer "Why did John leave the party?" using this heuristic. Goal Intention Access will take us to the immediate goal: he wanted to be near a vehicle. Goal Precondition Access then tells us what plan this goal enables: he wanted to travel some significant distance - probably by car. And finally Goal Intention Access tells us why that plan was executed: he wanted to be at home. We would therefore arrive at the following response:

Q: Why did John leave the party?
A: Because he wanted to go home.

This is a reasonable answer, but it is not the only one people come up with. While about a third of our subjects mentioned going home, most of those subjects also reference the kidnapping. For example:

Q: Why did John leave the party?
A: He wanted to go home and deal with the kidnapping.

And a number of subjects went beyond both of these concepts to mention Mary:

Q: Why did John leave the party?
A: To comfort Mary.

While there was strong agreement across the answers to "Why did John lie to Karen?" there were a number of conceptually distinct responses to "Why did John leave the party?" The heuristics needed to provide these alternative responses could have been applied to the first question (He wanted to get the call from the kidnapper - he wanted to protect Mary from Karen's inquiries - he wanted to cooperate with the kidnapper) but these answers appear (if at all) only with very low frequency. If we can account for the actual responses that people give, we will be able to propose a general control structure for the task of narrative memory retrieval. This control structure can then be tested by comparing its predictions against human question answering behavior. Eventually, the theory should be refined to the point where it will be possible to study individual differences.

The notion of "computer memory" is typically praised for its superior speed and accuracy over human memory. This assessment will

persist as long as memory retrieval is confined to task of retrieving information that is typically found in computers. No person can match a computer for information retrieval when the information consists of payroll data or statistical analyses. But in fact, human episodic memory is vastly superior to the database design of a computer. This superiority is especially apparent in a task orientation like story understanding, where the complexity of episodic information must be tackled by techniques that construct information by processes of conceptual interpretation and inference.

REFERENCES

[Cullingford 78]
Cullingford, R. E.
Script Application: Computer Understanding of Newspaper Stories.
Technical Report 116, Yale University, 1978.

[DeJong 79] Dejong II, Gerald F.
Skinning Stories in Real Time: An Experiment in Integrated Understanding.
Technical Report 158, Yale University. Department of Computer Science, 1979.

[Lehnert 78] Lehnert, Wendy G.
The Process of Question Answering.
Lawrence Erlbaum, Hillsdale, New Jersy, 1978.

[Schank and Abelson 77]
Schank, Roger and Abelson, Robert.
Scripts, Plans, Goals, and Understanding.
Lawrence Earlbaum Associates, Hillsdale, New Jersey, 1977.
The Artificial Intelligence Series.

[Schank and Kolodner 79]
Schank, R. C. and Kolodner, J.
Retrieving Information from an Episodic Memory or Why Computers' Memories Should be more like People's.
Technical Report 159, Yale University. Department of Computer Science, 1979.

[Wilensky 78] Wilensky, Robert.
Understanding Goal-Based Stories.
Technical Report 140, Yale University, 1978.