UNIVERSITY
OF TORONTO

DEPARTMENT
OF
COMPUTER SCIENCE

ON KNOWING WHAT TO SAY: PLANNING SPEECH ACTS

Philip R. Cohen

Technical Report No. 118

January 1978

A Thesis submitted in Conformity with the
Requirements for the Degree of
Doctor of Philosophy
in the Department of Computer Science
University of Toronto

ABSTRACT

The goal of this thesis is to model some of the cognitive structures and processes involved in how people decide what to say in purposeful conversation. The main concern is to show how a speaker's knowledge of his hearer influences what he says.

Utterances in such dialogues, where speakers can be presumed to be speaking for reasons, can best be viewed as the performance of "speech acts" (e.g., requesting). By modelling the process of deciding what to say as one of problem-solving, we show that the same mechanisms used to plan non-linguistic actions are applicable to the planning of speech acts.

The methodology of this research employs a computer program that integrates beliefs, goals, problem-solving, and reasoning, with the use of language. The program does this by planning speech acts when it determines that to achieve its situational goals, it needs to influence someone else's behavior.

The system is able to plan request speech acts to influence its user's goals, and inform speech acts to influence his beliefs. In addition, it is shown that speech acts can be used to achieve subgoals of other such acts. For instance, in addition to requesting that the user perform some action, the system can also decide to supply him with information necessary to his performing that action -- information that it believes he does not already have. Our model is extendable to the more general problem of planning multi-party speech acts -- ones where acts requested by the speaker involve acts to be performed by parties other than the hearer.

The system's range is also demonstrated by its ability to plan numerous kinds of questions, viz. WH, yes/no, existential, teacher/student, when it wants to know something. This unique ability relies on the development of a representation for knowing <u>that</u> someone knows what the value of an expression is without having to know <u>what</u> he thinks that value is.

In order to plan speech acts, the process-model incorporates means for dealing with beliefs and goals in a fashion that is both formally precise and yet implementable. The implementation employs a set of nested and overlapping belief and want "spaces", embedded with a partitioned semantic network knowledge representation. These spaces provide the scoping mechanisms necessary to maintaining a model of someone else's beliefs.

The thesis presents a formal semantics for a logic of "belief" and "want". The model theory was specially designed to give an interpretation to formulas containing existential quantifiers whose scopes include beliefs. Our having interpretations for these formulas is crucial since such formulas are the representation for "knowing that", the concept involved in planning questions.

The program deals with such beliefs by distinguishing between existentially quantified variables whose scopes do not include a belief from those whose scopes do. A parallel distinction is made in the formal semantics by restricting the set of objects over which variables quantifying "into" beliefs can range. We demonstrate that a natural language understanding system would need a mechanism for handling quantified beliefs by showing that such formulas are the logical forms for referential definite and "specific" indefinite noun phrases. This thesis provides one such mechanism. We also show how such a system would acquire quantified beliefs in conversation.

Finally, we discuss how the model can be extended, by a concept called "mutual belief", to deal with the unbounded number of beliefs produced by any communication act. We suggest how this concept may be of use in understanding the planning of referring phrases.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the friendship, supervision, curiosity, thoroughness, patience, and wit of Professor Ray Perrault. The substance and results of this thesis are directly attributable to our countless discussions and his searching questions. I am thankful for my good fortune in having been associated with him. He, along with the rest of the AI group and the entire department of Computer Science combined to make my graduate education thoroughly enjoyable and intellectually rewarding.

James Allen's contributions, by way of our many discussions, our joint implementation efforts, and our friendship must be acknowledged with gratitude. Without his altruistic assistance, this research could not have been completed. I would also like to thank Corot Reason, Professor John Mylopoulos, and Walter Berndl for their invaluable help in providing the appropriate tools upon which OSCAR was built.

Special thanks are extended to David Elliott, whose editorial comments greatly improved the readability of the thesis. Throughout the course of the research we engaged in many "helpful" conversations (he knows what I mean) in addition to discussions on the relevance of Artificial Intelligence to Philosophy and Computer Science. Ed Lazowska also played a very important role as editor, devil's advocate, and "brainstormer", while Alex Borgida's provocative comments were most helpful.

Many other influential discussions were held with members of the AI group, of which those with Hector Levesque (re. the nature of "known constants") and Corot Reason (re. reference) deserve special mention. The theoretical and practical results of this thesis were also influenced by

conversations with members of the AI Center at the Stanford Research Institute, especially with Gary Hendrix.

I would like to thank my committee: Professors Derek Corneil, H. A. Gleason, Peter Lindsay, John Mylopoulos, Ray Perrault, Dionysios Tsichritzis, and Terry Winograd, for their diligence in reading and commenting upon this thesis in the face of tremendous time pressures.

Financial assistance was gratefully received from the National Research Council of Canada.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

CHAPTER 2:

SPEECH ACTS

CHAPTER 3

ORGANIZATION AND REPRESENTATION OF KNOWLEDGE

CHAPTER 4
USER MODEL AND MEMORY ORGANIZATION

CHAPTER 5

PLANNING SPEECH ACTS

CHAPTER 4
USER MODEL AND MEMORY ORGANIZATION

CHAPTER 5

PLANNING SPEECH ACTS

CHAPTER 6

PLANNING QUESTIONS AND THREE PARTY SPEECH ACTS

CHAPTER 7

A FORMAL MODEL OF OSCAR AND ITS RELATION TO OPACITY

CHAPTER 8

EXTENSIONS OF OSCAR'S BELIEF REPRESENTATION FOR MUTUAL BELIEF

CHAPTER 9

EXTENSIONS AND APPLICATIONS

CHAPTER 10

CONCLUDING REMARKS

[Page 285]

BIBLIOGRAPHY

[Page 290]

CHAPTER 1

INTRODUCTION

1.0 On Knowing What to Say

There are times when I will accidentally say something that I had no intention of revealing and blurt out my thoughts without considering my audience. People are disturbed by such remarks because they expect someone addressing them to plan his utterances with them in mind. What I would like to discover is how such planning occurs and why.

I am interested in showing what the process of designing an utterance might look like, and what might be needed in the way of cognitive structures to support such processing. This thesis proposes both structures and processes aimed at illustrating what influences our deciding what to say in goal-oriented (purposeful) conversation situations.

The thesis has complementary theoretical and practical goals. On the theoretical side, it approaches the modelling of the use of language by modelling parts of the conversation process. The practical goal is to demonstrate techniques that could be used in constructing helpful computer systems that engage their users in natural language dialogue.

Previously, there have been only meager techniques for getting machines to reason explicitly about their utterances. Existing computer systems that "communicate" with users in natural language have the means for generating utterances, but do not themselves have the motivation. Their production of surface structures presumes some representation of the meaning of what is to be said. What

ought to be said, however, has typically been decided in a simplistic and inflexible manner. Paraphrase or question-answering programs, for instance, do not themselves decide to paraphrase or answer for explicit reasons -- they simply perform the function for which they were designed.

This thesis integrates aspects of philosophical theories of language, knowledge, and action into a model of how people decide what to say in certain kinds of dialogues. In order to aid in establishing consistency, major aspects of the model have been formalized as a computer system. For expository purposes, I will blur the distinction between system and model except where such distinction is crucial.

The system plans its utterances to achieve specific goals, such as getting its user to believe or do something, that arise in the course of solving some other problem. The central aim of this thesis is to show how deciding what to say is a function not only of the beliefs, situational goals, processing strategies and limitations of the speaker, but also of the knowledge the speaker maintains about the hearer.

With the above goals in mind, let me add an important caveat. The current program produces semantic representations of utterances, not their English realizations. Rather than produce actual utterances, the program has reasoned what it wants to say and why. I make no claims for having solved the related philosophical and linguistic problems, nor have I fully implemented a helpful system. I have, however, developed techniques that I believe are critical to progress on the problems as a whole.

An important step in designing the form of an utterance is to plan the referring phrases. This thesis makes some suggestions for further research on this problem in the hope of connecting it to the functional approach to the study of language, which deals with why we say things the <u>way</u> that we

do. The thesis also suggests how techniques developed here could be used for building systems whose objectives are quite different.

In the rest of this chapter, I outline a point of view from which to develop a process-model of language use. I present capabilities any such model should have, and demonstrate, by way of the computer system, that the model presented here possesses some of them. Finally, I contrast this methodology for studying language with that of the currently dominant school of linguistics.

1.1 Conversation

A study of dialogue is necessary if we are to understand how language is used. A purposeful dialogue will be regarded as a sequence of actions performed by two people, each attempting to affect the other's beliefs, goals, and obligations. By concentrating on dialogue situations where the speakers of utterances have non-linguistic goals, it can be presumed that speakers are speaking for reasons related to the situation rather than for other motives. Specifically, the reasons for a dialogue participant's utterances are related to the incorporation of the other person into his (developing) plans. In such circumstances, a person's utterances can be viewed as produced by his more general problem-solving processes. When those processes propose goals of influencing the beliefs or behavior of others, the person may attempt to communicate.

It is claimed by philosophers that the essence of communication is the hearer's recognition of the speaker's intentions. Thus slamming a door communicates a person's feelings only when the hearer realizes the slammer's intentions. However, we will only be concerned here with communication occurring via the use of language. Linguistic communication thus takes place when the hearer discovers the

speaker's reason(s) for an utterance, not just when he understands its content. For instance, the hearer must recognize "It's cold in here" as an attempt to get him to close the window rather than as a statement of fact. in order for the speaker to succeed in communicating his desires.

Given this emphasis on intentions, the question becomes how intentions lead to language at all. How does a speaker's goals influence what he says? What options are available to him in designing utterances, and how are his choices influenced by his knowledge of his hearers?

Considering the speaker's intentions in analyzing utterances considerably expands the range of phenomena that may figure in the concept of "meaning". Proponents of truth-conditional semantic theories tell us that when one knows the conditions under which a sentence is true, one knows the meaning of that sentence. Apart from the fact that not all declarative sentences can be characterized as true or false, various philosophers would claim that an analysis of how the speaker is trying to use a sentence also plays a part in determining meaning. Examples such as the following do not fit naturally into truth-conditional semantic theories but require analysis of the speaker's intentions.

Scene: Office
Participants: S = a secretary
E = a shady executive
Action: Phone rings. S answers, places caller on hold and buzzes E on intercom.

E: Yes
S: Excuse me, Internal Revenue Service on line 1.
E: I'm not here.

Truth-conditional semantic theories would characterize this utterance as being analytically false, or even worse, as being meaningless. Do we know the meaning of this statement by knowing that under no circumstances is it true? It is unreasonable to assert that the executive did not mean anything by the utterance. He clearly intended his secretary to make a number of inferences from his reply, and to use these in answering the caller. Among others, he intended some of his goals to be recognized (e.g., his not speaking to the tax auditor), as well as some of his beliefs (e.g., that the secretary knew how to handle the situation). In order to decide that he would not be misunderstood, he had to employ his knowledge -- his mental "model" -- of the secretary, which included his knowledge of the secretary's model of him.

1.2 Language and Action: Speech Acts

The starting point for this thesis is Austin's observation ([Austin 1962]) that a theory of language is part of a theory of action -- linguistic actions are special cases of communication acts and can be incorporated into analyses of intentional behavior. The concept of a speech act was developed to account for what speakers are doing with their utterances. For instance, speakers request, assert, suggest and thank. One needs to be able to recognize the force of a speech act, how the act is to be taken (e.g. as a request), in addition to understanding its propositional content (e.g., a request that someone do something).

The same proposition may be used in the performance of different speech acts. Thus, "The ice is thin" can be either an assertion or a warning. To complicate matters, different surface structures can be used for the same speech act. "Can you close the door?" and "Weren't you going to close the door?" could both be considered as requests. From

this general perspective, the generation and recognition of speech acts become part of the language understanding problem. Chapter 2 will discuss some of the issues of speech acts and intention in the philosophical literature.

To illustrate how some of the problems of using speech acts can arise when modelling a dialogue situation, let us consider an example.

1.3 A Sample Conversation

Imagine going to an unfamiliar supermarket to purchase Brillo. After fruitlessly performing a search of the shelves, you decide to ask the nearest stockboy for information. Your conversation might proceed in the following manner.

Shopper: (1) "Excuse me."
Stockboy: (2) "Yes"
Shopper: (3) "I'm looking for Brillo."
Stockboy: (4) "We're all out.
" (5) How about SOS?
" (6) It's two aisles down, on the bottom shelf."

In explaining what is happening in such conversations, we look for plausible reasons for the utterances. We must consider such things as the past conversation and the speaker's current goals and beliefs in order to construct a model of the process of speaking that would determine ways of issuing the observed utterances. Similarly, we should develop a model of the comprehension process that describes how one recognizes a speaker's intentions, and thus what he means, from his utterances.

With these two points of view in mind, let us look at what appears to be going on in this dialogue. Assuming you are the shopper, you clearly issue the first utterance with the intention of getting the stockboy to attend to you. The

reply in Line 2 indicates that this conversational opening is completed and that your goal has been achieved.

Note that at this point it would be quite inappropriate for you to walk away and not speak again. The stockboy might, if confronted with such behavior, reassess his analysis. Despite overt clues such as intonation, he might decide that your intention in uttering "Excuse me" was not to get his attention, but rather to get him to move or to apologize to him. Such conversational "obligations" as speaking again have been documented by sociolinguists (for instance, [Sacks et al. 1974].) Our concern in modelling the conversation process is discovering what gives rise to them.

Line 3 is not nearly as simple as the ones preceding it. Read literally, this declarative utterance simply states a fact about your goals. Certainly, not all statements of goals produce such responses as Lines 4-6. For instance, had you stated "I want to make a dentist appointment", the stockboy might have been somewhat perplexed and wondered why you were bothering to talk with him.

Recalling our original setting, you were missing a piece of information knowledge of the location of Brillo, since otherwise you would have proceeded to it and obtained the desired quantity. Presumably, you decided to ask the stockboy for help in locating the item, reasoning that stockboys place items on shelves and thus are likely to know where Brillo might be. With this reasoning you simply stated your goal, expecting that the stockboy would do something about it.

One way of interpreting this utterance is that you believed he needed to know your goal before being able to help you achieve it. You thus communicated the necessary information. But this "simple" communication act had to take into account your knowledge of his beliefs and goals,

of stockboys in general, of the helping process, and of his knowledge of you.

The stockboy's behavior is an example of what I would characterize as helpful -- behavior that is more than is literally required. Answering yes/no questions with more than just a yes or no, and reacting to statements of goals by doing something to achieve those goals, are examples of this idea.

In further examining Lines 4-6, we discover that the problem of knowing what to say again arises, this time from the stockboy's point of view. We can view his behavior as guided by helping strategies, which would often motivate utterances. Thus, his helping strategies seemed to result in his suggesting a way to overcome an obstacle (the absence of Brillo). He proceeded to state what he believed you needed to know in order to obtain SOS, namely its location. We would like to discover examples of rules for being helpful that could provide reasons for such speech acts.

1.4 Capabilities of a Helpful Computer Conversant

Let us consider some of the principal capabilities of a process-model of conversation. This model will be discussed in terms of an ideal helpful computer conversant that exhibits such behavior.

1.4.1 Problem-Solving

In order to model the ways in which humans engage in purposeful dialogues, a computer conversant must be a sophisticated problem-solver (planner). In general, the system should use its planning capabilities to reason about speech act(s) appropriate to furthering its situational goals. Based on its knowledge of the user, the system should then issue speech acts in a way that minimizes the chance that its intentions are misunderstood.

A particular speech act might be such that it satisfies a number of the system's goals simultaneously, perhaps at different levels. For instance, an act instrumental in achieving the goal of finding Brillo might also satisfy the system's conversational obligation to speak again. The system, then, should be able to plan speech acts that satisfy multiple goals at once.

1.4.2 Plan-Recognition

In order to do an adequate job of being either a speaker or a hearer, an ideal helpful conversation system would need both the ability to develop a plan and the ability to infer someone else's plans.

The system, as hearer, should try to recognize the speech act(s) that the user intended to perform by seeing which of a number of possible speech act identifications fit into a plan that can be inferred for him. Thus, the system should identify a user's utterance as the performance of a particular speech act by seeing if that speech act is consistent with other goals and beliefs he was believed to have.

1.4.3 Helping

Human communication is quite flexible. Indeed, we often become frustrated when our intentions are not recognized or our erroneous assumptions are not corrected. Current computer systems, including natural language understanding programs, however, are only capable of replying to the literal meaning of their input. For instance, a yes/no question generally receives a "yes" or "no" answer, with perhaps minimal explanation.

If we are to model natural human communication, our computer conversant must know how to be helpful. It should be able to debug someone's plans and, by such means as making suggestions, put him back on the right path to his

goal. Being helpful, however, is a business that requires tact. To avoid seeming overly solicitous, our ideal conversant should play the helping game with considerable finesse.

1.4.4 Knowledge of Others

The capabilities discussed above are dependent upon the system's using knowledge about the other party in the dialogue. To be more specific about how people use their knowledge of others, consider the following capabilities: Often, a speaker will explicitly avoid stating the obvious when he believes his audience already knows what he would be communicating. On the other hand, a speaker often explicitly realizes that his audience needs to know certain facts and consequently informs them of those facts without waiting to be asked. (Consider stating your name and phone number when placing a collect telephone call.) In either of the above cases, the speaker, among other things, determines if his audience already knows the information.

In deciding to ask someone a question, a speaker must reason that the hearer in fact knows the answer. But if the speaker already knew what the hearer thought the answer was, he would not need to ask. Hence, he must know _that_ the hearer knows something without knowing _what_ it is that he knows.

I believe that modelling how knowledge of someone else might be represented and used is critical to progress in understanding communication. The structure of the system's representation of knowledge should be rich enough to enable it to handle at least the above cases.

1.4.5 Steps Towards the Ideal System

The system presented in this thesis is only a small step towards the ideal process-model in that it only plans to influence the behavior of its user by asking questions, by informing him of facts it believes he needs to know, and by getting him to perform actions. This planning is based on a representation of the beliefs (and goals) of others that meets the criteria specified above. The representation of this knowledge has been formalized as a logic of belief.

As has been argued above, a plan recognition process is an important part of a model of language use, (and hence of conversing systems). This thesis does not deal directly with this question, nor does it address the problem of being helpful. My research strategy for modelling the intentionality of speech is to first model what speakers do before trying to recognize particular instances of their behavior. Hence, the program discussed here is only part of a dialogue system -- it only decides what to say.

1.5 Overview of OSCAR

I shall describe the program, named OSCAR, and then illustrate its speech act capabilities. In order to understand the program's operation, I must first discuss the function of planning algorithms.

1.5.1 Planning Algorithms

Most problem-solving or planning algorithms consist of a set of operators, each of which models the changes to the real world caused by a physical action. Such algorithms usually distinguish between reality and an internal model intended to correspond to reality. The model of the world contains propositions that describe a set of facts that are believed to be true. Not all actions are applicable in every state of the world and thus each operator has a set of

preconditions that state what propositions must be true in the model before the corresponding action can be executed. When an action is executed, it causes certain changes, or effects, to take place in the real world, with the appropriate changes taking place within the model.

The objective of a planning algorithm is to achieve a given goal -- a statement of what should be true in the real world after the plan is executed. The essence of the algorithm is the means for composing the available operators into a plan so that, when the associated action routines in the plan are executed, the goal becomes true. Each of the operators in the plan must be applicable in the state of the model that results from changes incurred in the initial configuration caused by the previously executed operators.

OSCAR would, in some loose sense, influence reality by speaking; the effects of its utterances would influence the hearer's beliefs and goals. In keeping track of its effects on the world, the best that OSCAR can do is to update its model of the hearer by the effects of speech act operators. We shall thus ignore the distinction between OSCAR's operators and action routines.

1.5.2 What OSCAR Can Do

The program displays a variety of verbal means to achieve its situational goals. It is able to:

- Plan REQUEST speech acts, for instance a speech act that could be realized by "Can you open the door?", when its goal is to get the user to want to perform some action.
- Plan INFORM speech acts, such as one that could be realized by "The door is locked", when its goal is to get the user to believe some proposition.
- Combine the above to produce multiple speech acts in one plan, where one speech act may establish beliefs of the

user that can then be employed in the planning of another speech act.

- Plan questions, i.e., requests that the user inform, when its goal is to believe something and when it believes the user knows the answer (without knowing what it is that the user knows). Both yes/no and WH questions can be planned. OSCAR can plan such questions when it is missing information it believes to be necessary to execute some other planned action. For instance, it could plan "Where is the key?" in the course of trying to unlock a door.

  Teacher/student questions are also plannable in the same way as real questions, but have different origins. For a real question, the questioner wants to know the answer while for a T/S question, the teacher already knows the answer and just wants to know what the student thinks the answer is. Hence, the plans for T/S questions occur as subgoals of real questions.

- Plan speech acts incorporating third parties, as in "Ask Tom to tell you where Mary is and then tell me."

1.5.3 The Program

The program incorporates a variety of "knowledge sources" to plan speech acts. First it maintains an explicit set of beliefs that represent its model of the world, including its model of the user. OSCAR's behavior is driven by a set of goals that are embedded within its beliefs. In order to achieve these goals, OSCAR uses a simple planning algorithm that, of course, has access to the system's beliefs and to the user model. Finally, an interpreter is provided that can recursively evaluate plans by evaluating their constituent operators. In a typical session, involving one domain of discourse, the user enters an initial configuration of beliefs and then generates and

executes plans. The diagram below gives a schematic of OSCAR's structure.

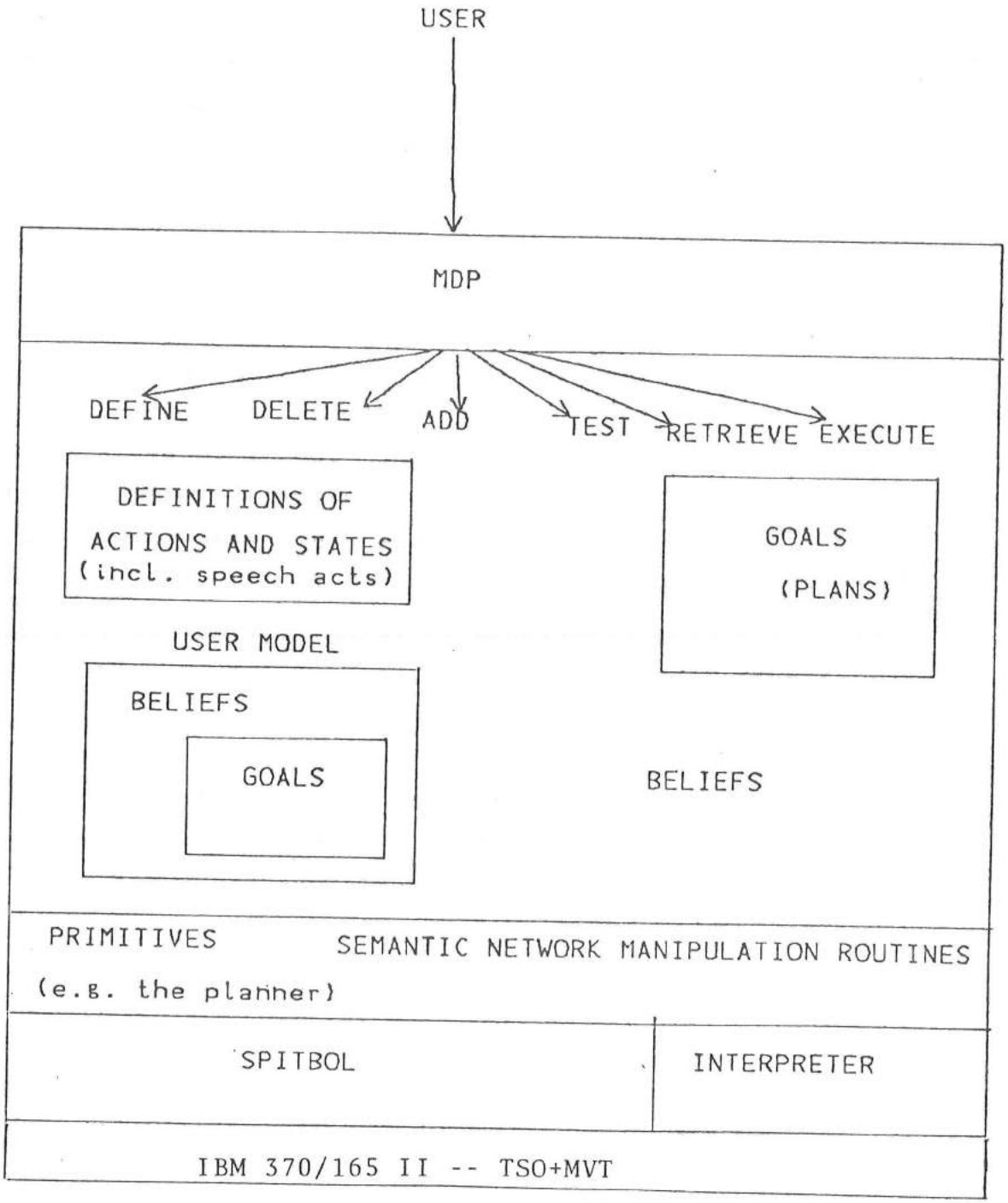

Figure 1

OSCAR

The design of the system was inspired by Norman and Rumelhart's [1975] MEMOD and VERBWORLD systems. OSCAR's user is given the capability to define a domain of discourse using a program called the Model Definition Package (MDP), written by James Allen.

The MDP allows the user to define a logical language in which he asserts (or tests) beliefs and goals. The statements in this language are translated into the system's internal representation of knowledge (a partitioned semantic network). MDP also provides the capability for defining new operators by combining previously defined operators with primitive ones, defined in the host programming language (SPITBOL). The ability to create new domains of discourse indicates that OSCAR is relatively domain independent, within the limitations imposed by the modelling formalism. Thus, it does not matter to OSCAR if the task is to get into a room or to bake a cake, provided the domain is modellable, at a satisfactory level of detail, using states and actions changing those states.

1.6 Examples of OSCAR's Behavior

OSCAR's speech-act capabilities will be outlined in a concrete situation. The domain of discourse (a variant of Power's [1974]) consists of a room with a swinging door that can only be unlocked from the inside. The initial configuration, that OSCAR believes to be shared knowledge between itself and the user, consists of its being outside the room and the user's being inside. The door is closed and locked. Where possible, I will display the actual interactions with the system in order to give the flavor of how it operates. One way of defining the initial configuration is to enter the following via MDP:

```
add (mutual_belief between system and user
    that (loc of user is inroom
```

and
loc of system is outroom
and
doorpos of door is closed
and
fastening of door is locked))

The physical actions that OSCAR believes to be shared knowledge between itself and its user include: moving through an open door, pushing the unlocked door from one position to another, and unlocking the door from the inside. OSCAR believes it and the user also share knowledge about the definitions of two (prototypical) speech acts REQUEST and INFORM. Finally, they share knowledge of acts, to be justified later, representing what it takes to get someone else to want some goal, and what it takes to get someone else to believe some proposition.

1.6.1 Possible Utterances

A minimal list of utterances that OSCAR ought to be able to produce, using only the knowledge outlined above, to achieve the goal of getting into the room is: (Utterances in parentheses are optional.)

1. Can you push the door open? [a REQUEST]
2. I want you to push the door open. [an INFORM of a WANT]
3. Can you unlock the door?
4. I want you to unlock the door.
5. I want to get in. (but the door's locked.)
6. I want to push the door open, but it's locked.

[the above two utterances are INFORMs of WANTs with optional INFORMs of obstacles.]

OSCAR can plan utterances 1 through 4. Utterances 5 and 6 require additional mechanisms that will be described, but are not implemented.

1.6.2 Planning a Speech Act

Armed with the appropriate beliefs about the state of the world and the user, the system can plan to achieve the non-linguistic goal of being in the room. As can be seen from the following annotated program output, this leads to a request: (lower-case is the user; upper-case is OSCAR)

1. system plan to achieve (loc of system is inroom)

DECIDE ON AGENT OF PUSHDOOR PLEASE.

system

[When the system cannot decide on a binding for a variable, it simply asks, but does not plan to do so. There is nothing sophisticated about this, I am simply acting as an oracle here.]

DECIDE ON AGENT OF CAUSE_TO_WANT PLEASE.

system

OK

2. generate (planof system to achieve
loc of system is inroom)

[This prints out the plan developed in Step 1 in the order in which the steps would be executed. The plan was developed in the reverse order.]

SYSTEM REQUEST USER TO DO USER UNLOCKDOOR

[The system describes the act REQUEST in the same format in which the user defined it (via the MDP). This description is not the same as executing the REQUEST.]

ACHIEVING: USER BELIEVE
SYSTEM BELIEVE
SYSTEM WANT
USER UNLOCKDOOR

[This proposition is what would result from a REQUEST. Requesting lets the user know the system wants him to unlock the door. It does not automatically cause him to want to unlock it since, conceivably, the user could refuse the request. I model this situation, as the next step in the plan shows, by a new act, CAUSE_TO_WANT, that states that in order to get someone else to adopt your goal as their own, you must get them to know what you want.]

THEN:

SYSTEM CAUSE USER TO WANT TO DO USER UNLOCKDOOR

ACHIEVING: USER BELIEVE

USER WANT
USER UNLOCKDOOR

[Notice that a person's WANTs are represented as embedded within their beliefs.]

THEN:

USER UNLOCKDOOR

ACHIEVING: FASTENING OF DOOR IS UNLOCKED

THEN:

SYSTEM PUSHDOOR FROM CLOSED TO OPEN

ACHIEVING: DOORPOS OF DOOR IS OPEN

THEN:

SYSTEM MOVETHRUDOOR FROM OUTROOM TO INROOM

ACHIEVING: LOC OF SYSTEM IS INROOM

3. execute (planof system to achieve
(loc of system is inroom))

SYSTEM REQUEST USER TO DO USER UNLOCKDOOR.

[This executes the REQUEST and asserts its effects -- that the user believes the system wants him to unlock the door -- as new beliefs.]

The following is OSCAR's plan to achieve the goal of being in the room using INFORM rather than REQUEST. (Utterances 2 and 4 are produced in this fashion).

SYSTEM INFORM USER THAT

SYSTEM WANT USER TO UNLOCKDOOR.

ACHIEVING:

USER BELIEVE

SYSTEM BELIEVE

SYSTEM WANT USER UNLOCKDOOR

[In general, one cannot directly influence someone else's beliefs. The INFORM speech act lets the hearer know what the speaker believes. Notice that the INFORM of a WANT will eventually lead to the same effects as the REQUEST above.]

THEN:

SYSTEM CAUSE USER TO WANT TO DO UNLOCKDOOR

.
.
.

[the rest of the plan, where the user will unlock the door, OSCAR will open it and move inside, is the same as the previous example.]

Choosing the user to be the agent of PUSHDOOR causes the system to request the user to push the door or to inform the user that the system wants him to push the door, rather than to unlock it. Hence, the propositional content (the act being requested or wanted) is influenced by the choice of agents during planning. Such a choice leads to utterances 3 and 4.

Utterances 5 and 6 cannot be produced by the current system since it has no reason to state such goals. What is missing, I claim, is a definition of an act HELP that would state as a precondition that the helper must know the goal (and perhaps the obstacle blocking it) of the "helpee".

1.6.3 "Need to Know" Utterances

To demonstrate additional utterances, we can expand the domain of discourse to require that the door be unlocked with a key. We need to add a new physical action GET, representing the means of obtaining some object, that requires as a precondition that the agent know the location of the object. If the system believes the key is in a closet in the room, and does not believe that the user knows the key's location (perhaps because it hid the key), then OSCAR can produce the speech act equivalents of:

- Can you unlock the door? The key's in the closet.
- The key's in the closet. I want you to get it and unlock the door.
- Can you push the door open? The key's in the closet.
- The key's in the closet. I want you to get it and then unlock the door and push it open.

It has reasoned that the user needs to know the key's location before he can successfully perform the requested

action. It thus tells him where the key is in addition to making the request. There are times when determining the user's needs may involve planning with the user's knowledge and informing the user of the facts necessary to enable such supposed plans to succeed.

1.6.4 Planning Questions

Let us assume here that the positions of OSCAR and the user with respect to the room are reversed so that OSCAR is inside and the user is outside. The system believes that the user knows where the key is but does not itself know the key's location. In this situation, OSCAR's plans result in the speech act underlying the question "Where is the key?" as follows:

```
system plan to achieve (loc of system is outroom)
DECIDE ON AGENT OF PUSHDOOR PLEASE.
system
DECIDE ON AGENT OF CONVINCE PLEASE
user
DECIDE ON AGENT OF CAUSE_TO_WANT PLEASE
system
OK.
generate (planof system to achieve loc of system is outroom)

SYSTEM REQUEST USER TO DO
    USER INFORM SYSTEM THAT
         LOC OF KEY IS A PARTICULAR PLACE

         [A Where-question is constructed as a request that
         the user inform the system about the key's
         location. As usual, the point of the REQUEST is to
         indirectly get the user to decide to do the INFORM
         speech act.]

    ACHIEVING:

USER BELIEVE
    SYSTEM BELIEVE
      SYSTEM WANT
         USER INFORM SYSTEM THAT
                  LOC OF KEY IS A PARTICULAR PLACE

    THEN:

SYSTEM CAUSE USER TO WANT TO DO
    USER INFORM SYSTEM THAT
```

LOC OF KEY IS A PARTICULAR PLACE

ACHIEVING:

USER BELIEVE
  USER WANT
    USER INFORM SYSTEM THAT
      LOC OF KEY IS A PARTICULAR PLACE

THEN:

USER INFORM SYSTEM THAT
  LOC OF KEY IS A PARTICULAR PLACE

ACHIEVING:

SYSTEM BELIEVE
  USER BELIEVE
    LOC OF KEY IS A PARTICULAR PLACE

[A speaker cannot directly influence someone else's beliefs. Rather, the hearer must decide to believe the speaker. This is represented as an intermediate act, CONVINCE, whose preconditions represent what is necessary to get someone to believe something.]

THEN:

USER CONVINCE SYSTEM THAT
  LOC OF KEY IS A PARTICULAR PLACE

ACHIEVING:

SYSTEM BELIEVE
  LOC OF KEY IS A PARTICULAR PLACE

[At this point OSCAR would know the information. It can thus proceed with unlocking the door and getting out of the room.]

THEN:

SYSTEM GET KEY

ACHIEVING:

SYSTEM HAVE KEY

THEN:

SYSTEM UNLOCKDOOR

ACHIEVING:

FASTENING OF DOOR IS UNLOCKED

THEN:

SYSTEM PUSHDOOR FROM CLOSED TO OPEN

ACHIEVING:

DOORPOS OF DOOR IS OPEN

THEN:

SYSTEM MOVE FROM INROOM TO OUTROOM

Other examples of types of questions that can be generated are the speech act equivalents of:

Is the door closed? (OSCAR must believe that the user knows the door's position without knowing what position the user thinks the door is in.)

Do you know where the key is? (Literally, this is a yes/no question about the user's knowledge. Asking it involves deciding that the user knows the truth-value of some proposition concerning his own beliefs. OSCAR always believes that if someone believes a proposition P, then he believes that he believes P.

1.6.5 Planning Three-Party Speech Acts

All of the above capabilities are applicable to conversations with more than two participants. OSCAR can plan to incorporate third parties into plans. In a situation where there are three "people", OSCAR, the user, and Tom, OSCAR plans speech acts that could be realized as:

- "Ask Tom to tell me where the key is."

  This is not a question since the user requests Tom to tell OSCAR, rather than the user, where the key is.

- "Ask Tom where the key is and then tell me."

  OSCAR requests that the user ask a Wh-question. OSCAR then requests that the user inform OSCAR of the answer.

- "Ask Tom if the key is in the closet."

  In this instance, OSCAR is requesting that the user ask Tom a yes-no question. However, OSCAR does not want to know the answer itself, or else it would have asked to be told the answer, as above. OSCAR cannot plan the above as an indirect question.

All these are plannable provided OSCAR believes, and believes the user believes, that Tom knows where the key is. Neither OSCAR nor its model of the user need contain any additional information as to the key's location. However,

if OSCAR did not believe that the user model contained the fact that Tom knew where the key was, OSCAR would plan to tell the user that fact and then ask the appropriate questions. For instance, it could plan the speech acts underlying:

- "Tom knows where the key is. Ask him to tell you where it is and then tell me."

The plans for this and the other utterances above are too involved to present in this chapter (they are presented in Chapter 6.)

The above (implemented) examples demonstrate how the system can reason about its speech actions. However, the decision of what to say is only part of the process of producing an utterance. In general, referring expressions will have to be established and lexical items must be chosen. I suggest that linguistic devices such as reference and focus are intentional and thus may be plannable. Such planning is beyond the scope of our current theories but ultimately will be the test of the view that language can be regarded as action.

1.7 Methodology: Pragmatics, Performance, and Processing

The model I have developed in this thesis is a first attempt at examining how speakers know what to say in certain kinds of contexts. This problem has heretofore been classified as one of linguistic "performance" (See [Chomsky 1965] for a discussion of the "competence-performance" distinction.) I believe there is a prevalent misunderstanding about the ways in which "performance" models relate to what is often termed "pragmatics".

I will attempt to present Chomsky's original distinction and argue that he is limiting his interests to "language" narrowly defined by grammar. This definition of language will be shown to be based upon a questionable isolation

(admitted as such by Chomsky) of the "language faculty" from other "faculties of mind". In this scheme of things, I argue that pragmatics is then not subject to the standard competence-performance discussion dealing with grammar. Given a different division of the problems of using and understanding language, I will suggest that another competence-performance distinction, one centrally based on processing, is more appropriate. The methodology of this thesis is then discussed in view of the goal of developing a "conversationally competent" system.

1.7.1 Competence and Performance -- Grammar

Chomsky has popularized a distinction between a person's linguistic competence -- what he tacitly knows about his language, i.e., what he can in principle understand and produce -- and his linguistic performance. In Chomsky's words:

> Linguistic theory is concerned primarily with an ideal speaker-listener, in a completely homogeneous speech-community, who knows its language perfectly and is unaffected by such grammatically irrelevant conditions as memory limitations, distractions, shifts of attention and interest, and errors (random or characteristic) in applying his knowledge of the language in actual performance...
>
> We thus make a fundamental distinction between competence (the speaker-hearer's knowledge of his language) and performance (the actual use of language in concrete situations)...
>
> A grammar of a language purports to be a description of the ideal speaker-hearer's intrinsic competence. [Chomsky 1965, pp. 3-4]
>
> The competence of the speaker-hearer can, ideally, be expressed as a system of rules that relate signals to semantic interpretations of these signals....[Such a system of rules] is descriptively adequate to the extent that this pairing corresponds to the competence of the idealized speaker-hearer. [Chomsky 1966, p. 75]
>
> The grammar is a system of rules and principles that determine the formal and semantic properties of sentences. The grammar is put to use, interacting with other mechanisms of the mind, in speaking and understanding language. There are empirical assumptions and conceptual distinctions embedded in this account, and they might be wrong or misguided, but I think it is not unreasonable,

given present understanding, to proceed with them. [Chomsky 1975, pp. 28-29.]

The competence-performance distinction, as presented by Chomsky, can be viewed as pertaining to the study of grammar. Grammar is regarded as a mechanism; linguistic performance is to be regarded as how that mechanism is used. The influence of non-grammatical factors on the speaking process is what prevents a speaker's linguistic performance from directly reflecting his competence.

One's initial reaction to such statements is that simplifying the study of language to avoid consideration of the minutiae of speech on particular occasions is a reasonable research strategy. Once we are clear on this score, it is perhaps easy to see why Chomsky would say:

> ...it is difficult to see how performance can be seriously studied except on the basis of an explicit theory of competence that underlies it, and, in fact, contributions to the understanding of performance have largely been by-products of the study of grammars that represent competence. [Chomsky 1966, p. 73, emphasis mine.]

Surely, to see how grammar is used, one would find it advisable to have a grammar to study. Hence, one might be able to claim that a grammar can be developed independently of performance issues.

In Chomsky's scheme of things, grammar certainly interacts with the rest of the mind. However, it is quite clear throughout his recent book, Reflections on Language [Chomsky 1975], that he holds a special place, perhaps even physically distinct from the rest of the mind, for a language faculty. This faculty of mind is then used to motivate a narrow (and technical) definition of language that is different from our intuitive understanding of the word.

> There has been much discussion of the so-called "innateness hypothesis," which holds that one of the faculties of the mind, common to the species, is a faculty of language that serves the two basic functions of rationalist theory: it provides a sensory system for the preliminary analysis of linguistic data, and a schematism that determines, quite narrowly, a certain class of grammars.... The language faculty, given appropriate stimulation, will construct a grammar; the person knows the language generated by the grammar....Questions related to the language faculty and its exercise are the ones that, for me at least, give a more general intellectual interest to the technical study of language". [Chomsky 1975, p. 13, emphasis mine.]

Given the above, the narrow version of "language use" does mean the use of grammar. Since linguists are not yet concerned with how the grammar is physically realized in the mind, it is a defensible methodology, though by no means universally accepted, to study grammatical competence independently of grammatical performance.

But, as in the tale of the blind men examining the elephant, there is no reason to suppose that the part of the beast that has been intensively explored to date is all there is. The tools used for examining one aspect may not by appropriate to the examination or integration of others. In this vein, Winograd points out one of the implicit assumptions made by many linguists:

> In avoiding processing models altogether, however, they [linguists] are also adopting the rather questionable assumption that it is possible to formulate a characterization of language structure which is independent of the processes of language use, and that the resulting characterization will be simpler and logically prior to any characterization which is based explicitly on processing. The truth of this hypothesis is an empirical question -- the analogous statement will be true for some sciences and false for others. There is some evidence that it is true for those details of syntax which lead to grammaticality judgements ...but very little evidence that [it] is true for language as a whole, particularly when meaning is taken into account. [Winograd 1976, p. 8]

Winograd's point pertains to a wider view of "language" that includes meaning and speakers' intentions. From this perspective, grammatical rules can be regarded as describing the behavior of cognitive mechanisms that produce and

understand language. (A similar point is raised by Morgan [1975, footnote, p. 295.]) However, description is only one level of explanation. A deeper understanding, perhaps leading to some innate properties of the mind, may be attained by understanding how the mechanisms that underlie grammar operate. To do this, we may have to describe how proposed mechanisms are influenced by information deemed to be outside of the scope of the narrow definition of "language". For instance, in modelling the process of discovering the referent of a definite noun phrase, we may have to consider the speaker's beliefs and intentions.

It is possible that the mechanisms supporting a person's use of grammar in his producing sentences may be intimately bound up in the capability to perform other intentional actions. Hence we would want a model of the processing of language that could capture such commonalities.

> A hypothesis about language production or comprehension gains plausibility if it can also be used in explaining other cognitive processes, such as perception or problem-solving. [Winograd 1976, p. 24]

We are lead to take this view of language when we question the distinct existence of the faculty of language.

1.7.2 Expanding Our Scope

When trying to study certain phenomena produced by a complex entity, one first tries to separate out those facets of that entity that, if understood, would prove to be central to an explanation of the phenomena. The facets of interest, termed "the system" are then (perhaps temporarily) excised from the entity, leaving as residue "the environment". When the system is language, the proposed environment becomes the rest of the mind. The study of the system then proceeds in a similar fashion -- one isolates and describes certain components, and then describes their interactions, in the hope that the essence of the system has been captured.

However, in separating the system from its environment, as well as in decomposing complex systems into smaller, more managable pieces, one should keep in mind that there may be other ways of decomposing the original entity. What were intercomponent interactions, or interactions with the environment, could become intracomponent ones under a different division, and vice-versa. Such decompositions of an entity are thus heavily dependent upon the phenomena that one originally wanted to understand.

A primary goal for Chomsky is to discover those aspects of the mind that are innate. As a step in the direction of describing the faculty of language, he would like to characterize "Universal Grammar" -- those aspects of grammar thought to be common to all natural languages and hence to all people. However, many properties of the use of language (in a wider sense) may also be universal and yet have little to do with grammar, and hence to his "language faculty". Hence when we expand our interests and decide to study language use, we find that what was once relegated to the environment might well need to be part of the system. The premise I then question is the usefulness of the standard distinction between language and non-language when our goal is to understand how humans communicate linguistically.

Indeed, Chomsky questions his own dichotomy as well, but adopts a different position from the one taken in this thesis.

> It is a coherent and perhaps correct proposal that the language faculty constructs a grammar only in conjunction with other faculties of mind. If so, the language faculty itself provides only an abstract framework, an idealization that does not suffice to determine a grammar.
>
> Suppose that there is no sharp delimitation between those semantic properties that are "linguistic" and those that form part of common-sense understanding, that is, the cognitive system dealing with the nature of things named, described or discussed. Thus, lexical items might be related by principles that form a kind of central core for a system of common-sense beliefs, with no sharp distinction between analytic and synthetic propositions....Under this assumption, lexical

> items are located in "semantic space" generated by the interaction of the language faculty and other faculties of mind....
>
> These are by no means implausible ideas. If they are correct, the language faculty does not fix a grammar in isolation, even in principle....
>
> The place of the language faculty within cognitive capacity is a matter for discovery, not stipulation....it may well be impossible to distinguish sharply between linguistic and nonlinguistic components of knowledge and belief. Thus an actual language may result only from the interaction of several mental faculties, one being the faculty of language. [Chomsky 1975, pp. 41-43.]

The proposal that Chomsky believes is "coherent and perhaps correct" suggests that we regard the language faculty as intimately, perhaps indistinguishably, interacting with other faculties of the mind, such as problem-solving and common-sense. Hence, he concedes that a more holistic approach could be a serious competitor to his theory of cognition.

A primary goal of this thesis is to show that many of the interactions that have been claimed to be of secondary interest become crucial to a study of language use. In particular, I wish to demonstrate interactions among representations of beliefs and goals, common-sense reasoning and problem-solving capabilities with what has come to be called "language". However, the research is not at the stage where it can be claimed that boundaries need to be redrawn. That must await further inquiry into the use of these new sources of knowledge and problem-solving mechanisms to influence the production of surface structures. (See [Morgan 1975] and [Ross 1975] for some examples of the influence of pragmatics on syntax.) The more intertwined we can show the environment to be with the system, the better will be the case for cutting the language pie differently.

1.7.3 Pragmatics and Conversational Competence

It has been suggested (e.g., [Stalnaker 1972]) that syntax deals with the study of sentences, semantics deals with the study of propositions, and pragmatics attempts to account for a speaker's use of language for his own ends. Speakers, motivated by their goals and influenced by their beliefs and expectations, thus employ propositions, realized as sentences, to perform actions.

While there may not be clean divisions between syntax, semantics, and pragmatics, the studies of speech acts and intention-based meaning would fall under this heading of pragmatics. One of the central problems for a theory of speech acts has been a characterization of the necessary and sufficient conditions on their successful performance -- conditions presumably written in terms of a speaker's goals and beliefs. There is no reason to suppose, however, that we can discover such conditions without investigating the processes that operate on speech acts. One reason for concerning ourselves with such processes is that they may impose criteria of adequacy on speech act definitions. In this thesis, I will show that the positing of even a minimal mechanism for planning speech acts leads to a reorganization of a list of such conditions proposed in the literature.

Certainly the conditions on speech acts may be described as pertaining to "competence", but we may be able to go further; we may be able to incorporate the processing underlying the use of speech acts into the notion of competence.

Conversational competence, when viewed in the light of the problem of knowing what to say, is a characterization of a speaker's capacity to produce speech acts. A way of describing such a capacity for action is to model the processes that produce the behavior. One might then be tempted to call those factors that influence these processes

to issue one particular speech act over another as pertaining to conversational performance. But as we come to understand more about the use of speech acts (and, in general, about problem-solving), many phenomena that are classified as performance by the above definition would be accounted for by the competence theory. What would ultimately be the subject of a performance theory of conversation would then be a characterization of the means by which aspects of linguistic behavior are affected by human frailty and whim (just as Chomsky originally defined the term).

To discover if a speaker is conversationally competent, we would situate him in the pragmatic situation of uttering. We would have knowledge of (or, when judging a program, would specify) the speaker's beliefs about a certain situation, the methods he has at his disposal, as well as what goals he is now pursuing. We would then like to see if he can generate a certain class of speech acts that we would claim would be reasonable for competent people to say, given the same knowledge and intentions.

With regard to OSCAR, a rigorous test of conversational competence would require our setting up an experiment to compare subjects' utterances to OSCAR's behavior under the corresponding circumstances. The program is in no way ready for such a test. While it can plan a number of different ways to linguistically further its situational goals, some of its speech acts can immediately be judged as unreasonable since it can be claimed that a competent adult human would not produce them. We need to explain why such naive speech acts arise and what can be done to generalize the planning algorithm to avoid the unnatural behavior.

Before one can subject the program to such a test, it would have to be capable of a wide variety of behavior. Hence, the goal for this thesis has been to demonstrate such

a range. The planning routines contain certain decision points regarding selection of goals, operators, bindings, etc. What I shall do is to show how if a certain decision is made, a certain behavior is observed. By systematically varying the choices at these decision points, I hope to show OSCAR's range of speech act behavior -- its capacity to use speech acts -- and hence what conversational competence it has.

What I am not doing is giving algorithms for intelligently making those decisions. Since researchers of problem-solving have had difficulty in formulating such algorithms and since I am not researching problem-solving per se, but rather its application to language, I feel it is methodologically justifiable to personally intervene at certain stages in the program's operation to make appropriate choices. Consequently, the issues of why OSCAR should choose one legitimate way or another of achieving its situational goals are not addressed here. Unlike many linguists, though, it is not suggested that such issues are inherently of secondary interest.

1.7.4 Models of Cognitive Processing

Many of the problems that are raised in this thesis have to do with cognitive processes. The computational approach to the study of intelligence maintains that interesting generalizations can be made in our understanding of cognition by developing algorithmic models of cognitive processing. This approach is as interested in the mechanisms that use proposed representations of linguistic structures (semantic, deep, and surface) as it is in the structures themselves. Hence, the computational approach is aimed at studying interactions of process and representation.

While a descriptive characterization of observed linguistic behavior may be interesting, the methodology

lacks the machinery necessary to enable us to understand how or why various observed phenomena occurred. In this sense, descriptive theories that ignore underlying processing are not sufficiently explanatory. We may be able to discover phenomena that can be described grammatically but are direct results of cognitive processes of certain sorts. By having a model of such processes, we would be in a better position to explain why the phenomena are the way they are and how they interact with others.

Demonstrating that these algorithms meet their theoretical specifications often requires our developing complex, vertically-integrated process-models that produce the desired behavior. Such algorithms serve, in Winograd's [1976] terminology, as "blueprints" for the specifications. Unfortunately, it is often hard to determine by hand the interactions of algorithms in applications as complex as those found in cognitive modelling. Consequently, we often prefer to demonstrate and debug our theories by implementing our algorithms on computers.

We cannot, however, claim such programs to be cognitive models in the strict sense of being point by point simulations of the actual ways in which humans produce language. We do not know how to test such a model. Our programs are intended to illustrate some of the kinds of knowledge and processes we believe speakers use in their design of utterances. The ways in which we propose the programs generate utterances, at some level of detail abstracted from the particular implementation, ought to be ones that meet the theoretical specifications and agree with our intuitions about what people might be doing, based upon our observations and introspections. Until cognitive scientists develop better testing methodologies, we will not be able to determine if our models describe the actual ways in which humans produce and understand language. This is not a difficulty of which we should be embarrassed. On the

contrary, the first and most important step is to arrive at a model that produces the desired behavior. OSCAR, then, is to be viewed as a feasibility study of the view that a theory of language can be regarded part of a theory of action.

1.8 Document Outline

The thesis begins by surveying the philosophy of language dealing with speech acts and intention. Many of the philosophical ideas discussed here will have parallels within the system.

Chapter 3 is a description of the representation of knowledge that OSCAR employs. I first present the tool that is used as a means for representing knowledge in the computer, partitioned semantic networks (patterned after [Hendrix 1976]). Once the basics are clear, I employ this tool to develop an object-centered modelling formalism, incorporating many ideas taken from [Levesque 1977].

Chapter 4 extends this formalism to deal with "believe" and "want" in order to structure OSCAR's memory. The concepts discussed here will be of fundamental importance to the system and to the rest of the thesis.

Chapter 5 then explains how OSCAR plans speech acts from non-linguistic goals. I discuss what influences the planning of simple REQUEST and INFORM speech acts. This chapter shows how OSCAR can employ an explicit model of its user's beliefs and goals in deciding what to say.

Chapter 6 discusses how OSCAR can plan to acquire information by planning questions. The fundamental result of this chapter is a means for representing facts like "the user knows where Mary is", allowing it to plan to ask the user a question about Mary's location. Once these basic mechanisms are presented, revising those of Chapter 5, I

proceed to demonstrate the planning of yes/no and teacher/student questions. To show that the results are not limited to just two people, I show how OSCAR plans three-party speech acts.

Chapter 7 develops a formal model theory for the logical language that OSCAR encodes -- a model theory very close to the structure of the system. The formal semantics for the language is then used to clarify a number of problematic constructs -- specifically, problems of referential transparency, and definite and indefinite noun phrases. Certain kinds of noun phrases are then shown to be the means by which the system would acquire information like "the user knows where Mary is".

Chapter 8 extends OSCAR's representation of belief to include the concept of mutual belief. This concept enables OSCAR to deal with the potentially unbounded number of beliefs caused by any communication act, and may prove to be vital to the planning and understanding of referring phrases.

Chapter 9 proposes ways in which the system might be extended to deal with the planning of referring phrases and with helping, by viewing both referring and helping as acts. The chapter also compares the methodology of this thesis to others in the literature and discusses some potential practical applications.

Finally, Chapter 10 will summarize the accomplishments of this thesis.

CHAPTER 2

SPEECH ACTS

Contents:

2.0 Introduction

This chapter presents a brief sketch of some of the philosophical ideas underlying this thesis. In particular, it will discuss the theory of speech acts as developed primarily by Austin and Searle. While the concepts and terminology developed here will be used extensively in the rest of the thesis, there is no attempt to be quite as precise as the original authors, and thus any inaccuracies are my own. Readers somewhat familiar with the speech act literature may want to proceed to section 2.3 or directly to Chapter 3.

2.1 Austin's Performatives

Austin [1962] points out that speakers do things with their utterances apart from simply stating facts. For instance, "I bet you five dollars that the Dodgers will win the series" is the offering of a bet (when taken up) and not the reporting of a bet. "I warn you that the building is condemned" is clearly an act of warning. Austin named declarative utterances of this type performatives, since they constitute the performing of an action.

Unlike the usual declaratives, such sentences are not true or false, but rather are subject to the same kinds of successes and failures as non-linguistic actions. They can

fail for any number of reasons, such as being applied in the wrong circumstances or being performed insincerely (e.g., requesting that someone fly a kite when you don't want them to).

A speaker, in uttering something, performs three kinds of speech acts: locutionary, illocutionary, and perlocutionary acts. A speaker performs a locutionary act by making noises which are the uttering of words in a language satisfying its vocabulary and grammar, and by the uttering of sentences with definite meaning. Such acts are used in the performance of illocutionary acts (which relate directly to this thesis) that are those acts performed in making utterances. Different types of illocutionary acts are said to differ in illocutionary force. For instance, stating, requesting, warning, ordering, apologizing, are claimed to be different illocutionary acts. Each illocutionary act contains propositional content that specifies what is being requested, warned about, ordered, etc. By this theory, communication is said to take place when the hearer understands the intended illocutionary force of an utterance rather than just its content.

New distinctions, however, bring new problems. In particular, illocutionary acts are not always recognizable as such from their form. They need not be explicitly stated in the first person singular, present indicative active forms as in "I warn ..." Simple cases such as "Patrons are requested to please remove their shoes" are easily characterized as requests. While the use of explicit performative verbs simplifies the hearer's determination of the utterance's illocutionary force, when no performative verb is used, ambiguities can arise that require a great deal of contextual interpretation on the part of the hearer. For instance, "Open the door" can be unambiguously determined to be a command, while "Can you open the door?" is potentially ambiguous. For a hearer to understand the

force of the utterance "The door", however, he would need to use knowledge of the speaker's desires and of the physical context of the utterance. For instance, the facts that the door is currently closed, that the speaker has two arm-loads of groceries, and that he wants to be on the other side of the door might all be relevant in determining that the speaker wants the door to be opened.

While a speaker may appear to be performing one illocutionary act, he may actually be trying to do something else. "We have to get up early tomorrow" may simply be an assertion or, when said at a party, may be intended as an excuse to the host for leaving, and, perhaps, as a request that the hearer leave. Such indirect speech acts are a current topic of research. (See, for instance, [Searle 1975], and [Gordon and Lakoff 1971].)

The last major kind of act identified by Austin is the perlocutionary act -- the act performed by making an utterance. For instance, with the illocutionary act of stating a sentence, I may convince my audience of the truth of the corresponding proposition (or insult or frighten them). Perlocutionary acts produce perlocutionary effects: convincing produces belief and frightening produces fear. While a speaker has performed both illocutionary and locutionary acts with the goal of achieving certain perlocutionary effects, the actual securing of these effects is beyond his control. It is entirely possible for a speaker to make an assertion, and for the audience to recognize the force of the utterance and yet not be convinced.

Though I believe people make a distinction between the illocutionary and perlocutionary aspects of an utterance, I do not believe we maintain a repertoire of perlocutionary acts as we do illocutionary acts. Instead, intended perlocutionary effects are our reasons for planning illocutionary acts. I thus regard a perlocutionary act as

the doing of something (illocutionary) that leads to a perlocutionary effect.

2.2 Searle's Theory of Speech Acts:

In an attempt to further refine Austin's distinctions, Searle [1969] presents a formulation of the structure of illocutionary acts (henceforth referred to simply as speech acts). Searle's goal is to discover necessary and sufficient conditions on the successful performance of various speech acts. The conditions determine rules constitutive of a speech act -- rules that define behavior as being of a certain sort. Any action that satisfies a particular set of constitutive rules for a speech act then is said to be a speech act of that type.

As an example, let us consider Searle's set of conditions for a speaker S, in uttering T, to request that some hearer H do action A. The conditions are grouped as follows:

- Normal Input/Output Conditions

These include such conditions as: H is not deaf and S is not mute, joking, or acting. (This is analogous to Chomsky's idealization of the conditions of uttering.)

- Propositional Content Conditions

Particular speech acts only use propositions of certain forms. The restrictions on these forms are stated in the propositional content conditions. For a request, the proposition must predicate a future act of H, while for a promise, the proposition must predicate a future act of S.

- Preparatory Conditions

A preparatory condition states what must be true in the world before the speaker can successfully issue the speech act. For a request, the preparatory conditions include:

- H is able to do A
- S believes H is able to do A
- It is not obvious to S and H that H will do A in the normal course of events (the "non-obviousness" condition).

Searle claims the non-obviousness condition is not peculiar to illocutionary acts. This thesis will support his claim by showing how the condition can be applied more generally to rational, intentional behavior.

- Sincerity Condition

A sincerity condition distinguishes a sincere performance of the speech act from an insincere one. In the case of a request, S must want H to do A; for a promise, S must intend to do the promised action; for an assertion, S must believe what he is asserting.

- Essential Condition

An essential condition specifies what S was trying to do. For a request, the act is an attempt to get H to do A.

- Force Condition (my terminology)

The purpose of the force condition is to require that the speaker utter a speech act only if he intends to communicate that he is performing that act. "Intending to communicate" involves having certain intentions regarding how the hearer will recognize the force of the utterance. The basic idea is that it is intended that the hearer recognize that the speaker is trying to bring about the satisfaction of the essential condition by means of the hearer's knowledge of the meaning of the utterance.

However, not just any means for recognizing the speaker's desires will do -- the hearer is to recognize the speaker's intentions by realizing that the speaker intends for him to understand the utterance's force as the speech act in question. [1]

By incorporating a force condition into the analysis, Searle requires the speaker (S) to determine if, to the best of his knowledge, the hearer (H) can recognize the speech act as the one intended. The recognition of intention problem is that the hearer will recognize an utterance as a particular speech act only if he believes the speaker intends for him to recognize the utterance in that way. Hence, for S to check the force condition, he must concern himself with H's recognition abilities that depend upon H's knowledge of S's intentions, that depend upon H's recognition abilities, etc. In this way, S may involve himself in an infinite regress of recognition of intentions. Schiffer [1972] proposes a potential way around this difficulty by mapping this regress to an infinite regress of knowledge using the concept of "mutual knowledge". Chapter 8 deals with OSCAR's means for representing such knowledge.

The force condition will not appear in the formulation for speech acts given in this thesis. The interested reader should refer to the note at the end of this chapter and to Chapter 9 for a brief discussion of its importance to understanding communication.

2.3 Searle's Taxonomy of Illocutionary Acts

Searle [1976] attempts to discover axes of variability along which one finds various illocutionary acts. He proposes twelve differences between speech acts (some obviously more central than others) and then employs them to create five classes of acts. Two of the dimensions of variability, illocutionary point and expressed psychological

state, and two of the classes are of primary interest to this thesis.

The directive class contains such acts as request, command, entreat and suggest. The goal of an act in this class, its illocutionary point, is to attempt to get the hearer to do some action. The expressed psychological state is "want", as found in the sincerity condition: the speaker wants the hearer to do the action. Members of the class may differ from one another along other dimensions; for instance, by the necessity of there being a social relationship between speaker and hearer (e.g., a command) or by the way in which the action relates to the interests of speaker and hearer (e.g., entreaties, suggestions). These differences would be reflected by the addition of new preparatory conditions to the various acts in the class.

The acts in the representative class all share the illocutionary point of transmitting the speaker's belief about the state of affairs described in the propositional content. (For instance, "inform" is such an act.) The significant feature of the acts in this class is that their propositional content can be characterized as true or false. For a speaker to sincerely inform someone that something is true, he must believe it is true and hence "belief" is the expressed psychological state. Acts in the class will differ in the gradations of belief they convey (e.g., stating that P is true vs. insisting that it is).

The other three classes are: commissives, expressives, and declarations. The point of a commissive act (e.g., promise or threat) is to commit (or oblige) the speaker to a future course of action. Such acts express the speaker's intention to do the act mentioned in the propositional content. The illocutionary point of an expressive act (e.g., thank or apologize) is to communicate the speaker's psychological attitude towards the propositional content. Unlike representatives, the truth of the proposition is not

asserted but is presupposed. Declarations are the performatives that Austin originally discussed -- those utterances, usually issued with respect to certain extra-linguistic institutions, that cause the world to reflect the propositional content simply by the process of uttering. For instance, betting, firing, resigning, christening are examples of this class.

2.4 Summary and Comments

Austin's analysis of a peculiar class of declaratives led him to viewing language as action. Speech acts can then be regarded as performed for reasons similar to those motivating the performance of non-linguistic acts. In addition, he observed that linguistic acts can be defective in ways similar to failures of non-linguistic acts. His conception of a locutionary act extends to the acts of referring and predicating. By treating reference as a speech act (as suggested by Strawson [1950] and Searle [1969], among others), we may be able to account for failures of reference, and for the comprehension and production of referring phrases, by understanding a speaker's intentions to refer. Such topics will be discussed briefly in chapter 9.

For Austin, a speaker, in uttering something, performs at least one illocutionary act. He claims that the force of the utterance can always be made explicit via the use of a performative verb. This raises a major (unsolved) question of whether the deep structure (and thus, to some, the meaning) of every sentence ought to include a performative verb (see, for instance, [Sadock 1974]). My belief is that a full understanding of an utterance requires an understanding of its intended illocutionary force.

Searle attempts to understand the nature of some of these actions by proposing necessary and sufficient conditions on illocutionary acts. Some of these conditions

will be explicitly found in the speech act definitions of Chapter 5, while others will be rephrased. In addition to criteria for judging the successful performance of speech acts, however, we also need criteria for deciding upon when we have done an adequate job in describing such acts. The methodology of this thesis is that speech act definitions will be organized and specified in enough detail to enable them to be incorporated into a process-model of speech act use.

While I will question some of Searle's conditions, I do feel that a taxonomy based upon the illocutionary point of the act to be an important step. By categorizing speech acts according to their illocutionary point, Searle is claiming there are five kinds of things people typically do with language. While one may disagree with his particular classes, such a taxonomy is useful in that it enables us to deal with prototypical acts representing a small number of speech act types. We can then expect similar, though not identical, effects for the issuance of other acts in each class. This thesis will deal solely with exemplars of the directive and representative classes. For the directive, request will be employed, while for a representative, an inform speech act will serve as the prototype. It will become clear that the process of incorporating these speech acts into plans, based upon their illocutionary points, should extend to other acts with the same point but different preparatory conditions. However before we proceed to such planning, we need to discuss the structure of OSCAR -- most importantly, its representation of knowledge.

2.5 Notes

1. Searle's force condition is based on Grice's [1957] definition for meaning and, by extension, for communication, wherein a person S means something by an utterance X iff S uttered X intending:

I. that the utterance X should produce a certain response r in the audience H,

II. that H recognizes S's intention I.,

III. that H's recognition of S's intention should be, at least in part, H's reason for adopting the response r.

(Grice includes in the term "utterance" such non-linguistic behavior as gestures, grunts, singing, humming, etc.)

According to the above definition, what S means is determined by, and only by, the intended response r. In particular, it is not determined by the meaning of the utterance X. The definition is intended to capture examples like the following (from Schiffer [1972]):

S intends to get H to leave the room by singing, in his usually awful manner, "Moon over Miami". S could have a few options in mind:

a. To get H to leave by annoying him, without getting H to realize that S intended for him to leave.

b. To get H to leave by getting him to realize that S intended for him to leave. S intends to bring this about by getting H to realize that S is deliberately trying to annoy him.

Case b), but not case a), satisfies condition 3 since for the latter, S does not intend that H recognize S's intentions. There are further levels to which this example can be taken -- ones that require a reworking of Grice's definition. The interested reader is referred to [Schiffer 1972] for further details.

While many of Searle's constitutive conditions for a request are satisfied here, Searle would not consider case b) to be a request. First of all, the propositional content condition predicating a future act A of H, as opposed to singing "Moon over Miami", is not satisfied. Secondly, Searle employs the understanding of force as the intended response, rather than the perlocutionary responses of Grice. Communication, for Searle (and Austin), is said to take place when the hearer understands how the speaker intends for the utterance to be taken -- when he understands what the speaker was trying to do with the utterance -- not when the intended perlocutionary effect is recognized.

Finally, Searle's force condition requires that the hearer recognize S's intentions based upon the meaning of the utterance. In the example, however, it is not the meaning of the utterance that facilitates this recognition, but the poor quality of the singing. The Gricean program, as amended by Schiffer, attempts to account for communication that need not take place via the standard conventions of language. In fact, they attempt to account for the meaning of an utterance in terms of what the speaker means by the utterance. All that Schiffer claims is necessary, regarding the utterance, is that it should have a certain feature (e.g., awful singing) that will indicate the intended force. Searle would apparently like to rule out such behavior as constituting a speech act.

CHAPTER 3

ORGANIZATION AND REPRESENTATION OF KNOWLEDGE

Contents:

3.0 Introduction

Much of the short history of the field of Artificial Intelligence can be viewed as focussing upon aspects of representation. The solving of problems, it was learned, is greatly facilitated by the choice of appropriate representations for those problems. Not surprisingly, there has been no consensus on one representation that is "best" for most problems, since efficiency considerations have often dictated stringent constraints upon representation schemes. Hence, chess programs employ different representations from vision programs, that are again different from natural langauge programs.

Even within these areas, no agreement has been reached. For instance, a debate still exists over whether the representation of knowledge for natural language systems should be based on logical formulas or on semantic networks. The differences between the two representations, however, are largely illusory since recently, Hendrix [1976] has shown that suitably constructed semantic networks have the full expressive power of quantified first-order logic. Fikes and Hendrix [1977] are attempting to produce rules of inference for these networks that are complete and consistent. If that effort is successful, then the choice between these two representation tools will have to take place on grounds other than expressive and deductive power since they can then simply be regarded as notational variants.

The primary difference between the representations is that first-order logic has a well-defined semantics (in the Tarskian sense) while in the past, semantic networks have not. However, in chapter 7 a formal interpretation of the logical language that OSCAR maps into a network data structure will be defined, and thus OSCAR's particular semantic network has a Tarskian semantics. Hence, we must search for other arenas in which to match the competitors.

The purpose of this chapter is not to present a detailed account of OSCAR's representation of knowledge (which is nearly identical to Hendrix's [1976]) but rather to discuss its general organization. Those readers familiar with semantic networks, and Hendrix's work in particular, may safely skip to section 3.4.

3.1 Semantic Networks

Semantic networks are intended to serve as both a representation for knowledge of the world and as an encoding of the semantics of natural language sentences. A semantic network is simply a data structure -- a directed, labelled

graph. The essence of a network data structure is that all relationships involving a particular node can be found from that node.

The nodes of the graph are usually regarded as representing concepts, while the labelled arcs (or edges) are viewed as encoding binary relationships between concepts. N-ary relations are represented by creating a node for the relation and connecting it via arcs to the concepts participating in the relation. Typical examples of concepts that would be encoded by nodes are: the set of people, the person named 'John Smith', my dog, etc. Typical binary relationships are: set membership, subset, father, age, and spouse. The usual convention is to encode as arcs only those binary relationships that do not vary over time. Thus, in most systems, "age" and "spouse" would not be encoded as arcs but as n-ary relations, along with actions like hitting, throwing, and moving.

For most early semantic networks (e.g., [Rumelhart, Lindsay, and Norman 1972], [Norman and Rumelhart 1973], [Hendrix et. al. 1973], [Simmons 1973], and [Mylopoulos et. al. 1975]) the representation of natural language sentences was based on Fillmore's [1968] analysis of English verbs. Fillmore pointed out that verbs can be regarded as having syntactic <u>caseframes</u> that indicate the arguments (cases) that a verb can be expected to take. In a sentence, the arguments to a verb are usually specified as noun phrases that refer to concepts assumed to play particular roles in the action described by the verb.[1] For instance, the verb "move" might have as cases: an <u>agent</u> -- the animal doing the moving, an <u>object</u> -- the thing being moved (perhaps the agent himself), a <u>source</u> -- the place from which the object is moved, and a <u>destination</u>, the final location for the object. The names of the cases are supposed to indicate "who did what to whom".

Verbs in English are mapped to objects in the network representing instances of actions. Since this thesis does not deal with parsing, I shall refer only to caseframes for actions, rather than for verbs. The claim then is made that there are a small number of cases that describe how the arguments to an action relate to that action. It is presumed that cases that have the same name pick out objects that play analogous roles in the corresponding actions. This aspect of the "semantics" of these networks, however, is external to the representation. An extended discussion of such issues can be found in [Bruce 1976].

Most natural language parsers can map simple paraphrases of a sentence into a single case structure representation. Thus, "John gave Mary the hammer" would have the same case structure representation (see Figure 3a) as "Mary was given the hammer by John" or as "It was the hammer that John gave Mary".

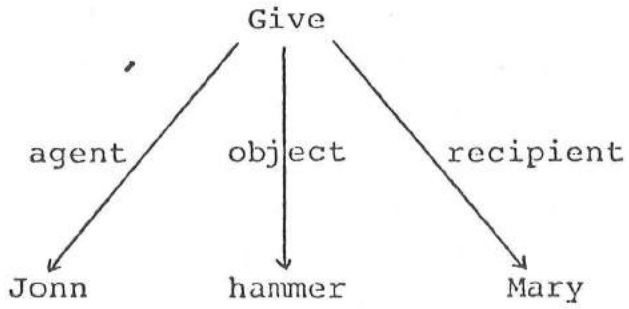

FIGURE 3a

EXAMPLE OF A CASE-STRUCTURE REPRESENTATION

The sentence is represented as an n-ary relation. The case arcs point to particular nodes (concepts) identified by the noun phrases in the sentence. The proper name "John"

enables the system to access the concept representing John. In a similar fashion, "Mary" or "the girl with red hair standing next to Bill" would pick out the concept standing for Mary. The object standing for the hammer is simply a particular instance of the class of hammers. Special procedures are required for discovering the referent of an arbitrary natural language description.

3.2 Some Difficulties with Early Semantic Networks

A number of problems with such schemes have been mentioned in the literature (see, for instance, [Nash-Webber and Reiter 1977], [Bruce 1975c], [Woods 1975], [Wilks 1975], and [Schank 1973]). A major criticism raised is that the "semantics" of such networks have often been somewhat mysterious. Precisely what did it mean to connect two nodes by an arc with a particular label? The labels were supposed to indicate the relationships between the concepts connected by those arcs, and they were supposed to be meaningful to the parsers that mapped sentences into these representations. However, the effects of labelling an arc one way or another were often kept secret in a global interpreter that manipulated the data structure. New arc labels were often invented when a new phenomenon was noticed, causing unforseen and unwanted interactions to develop. The modelling strategy presented here and the formal model of the system discussed in Chapter 7 hopefully enable OSCAR to avoid this charge by explicitly presenting the ways in which the various entities in the network are manipulated.

Many of the early versions of these networks were not representationally adequate -- they could not handle quantified statements, hypothetical worlds, and disjunctions. What was lacking was the equivalent of parentheses in predicate calculus -- a means for grouping entities to indicate the scopes of quantifiers. Hendrix

[1975] and [1976] has recently proposed solutions to these problems by developing partitioned semantic networks. Since these data structures are used extensively in OSCAR, they will be discussed in more detail.

3.3 Partitioned Semantic Networks

The new feature added to the above labelled, directed graph data structure is the space. Spaces are collections of other nodes, arcs, and spaces. Any space encodes an implicit conjunction of the relationships contained within it. In addition to having contents, spaces have all the properties of nodes.

3.3.1 Well-formed Partitioned Semantic Networks

Given a network with a set of nodes and spaces, we can characterize the legitimate data structures as follows:

N = {nodes} S = {spaces}, L = {labels}, OBJECT = N U S
where U is the set union operation.

All nodes (but not spaces) are contained in some space.

A (directed) edge is a 3-tuple <<O1,S1>,LL,<O2,S2>,S3> indicating that the edge in space S3, is labelled "LL" and relates object O1 in space S1 to object O2 in space S2. Notice that edges are not referenceable objects. S1, S2 and S3 could all be different, but usually aren't. The objects O1 and O2 could be identical, or could be any of spaces S1, S2, or S3.

Figures 3b, 3c, 3d, and 3e are examples of legitimate data structures. The conventions employed in these (and all) drawings of networks are:

- Boxes will indicate spaces.
- Labelled arrows will indicate arcs.
- Circles or capitalized words (or sometimes "<>") will indicate nodes.
- An arc will considered to be contained in (at least) the space in which its label is written.

- Nodes are contained in (at least) the space in which the circle or capitalized word representing the node is written.
- Spaces will be labelled by letters on the appropriate boxes.

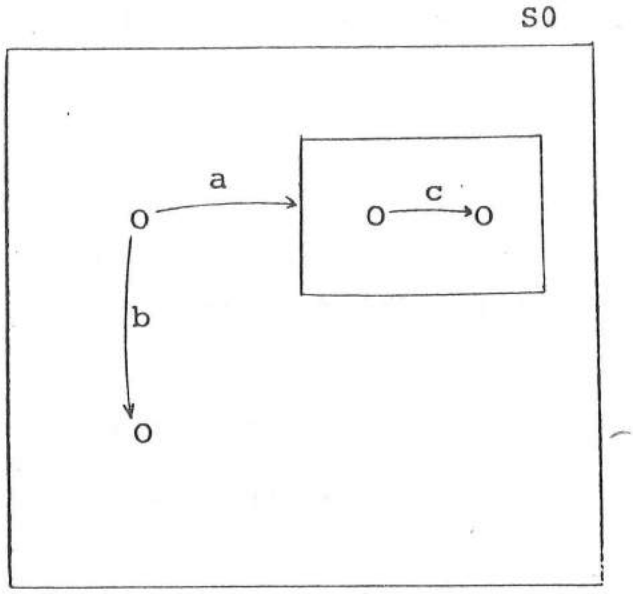

Figure 3b

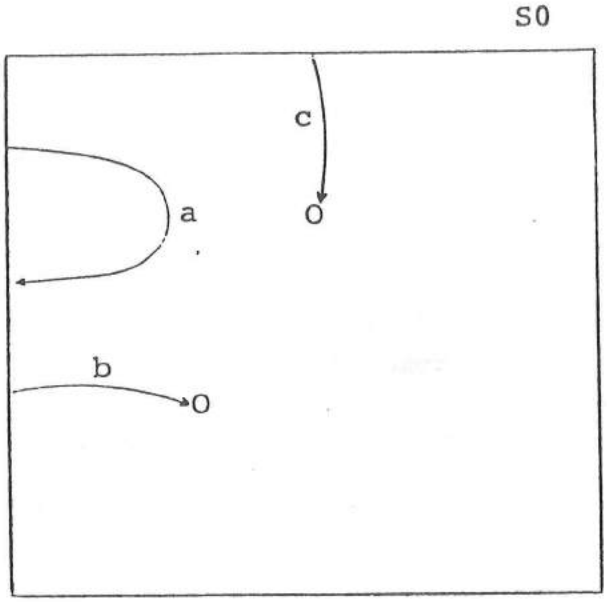

Figure 3d

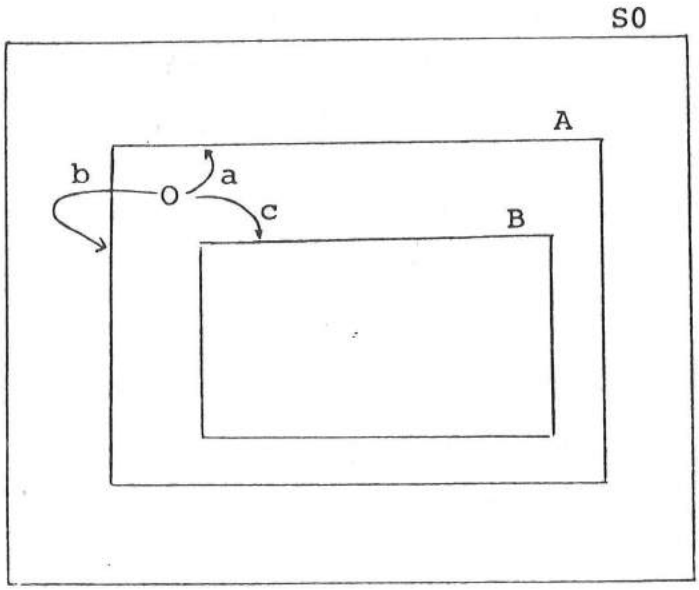

Figure 3c

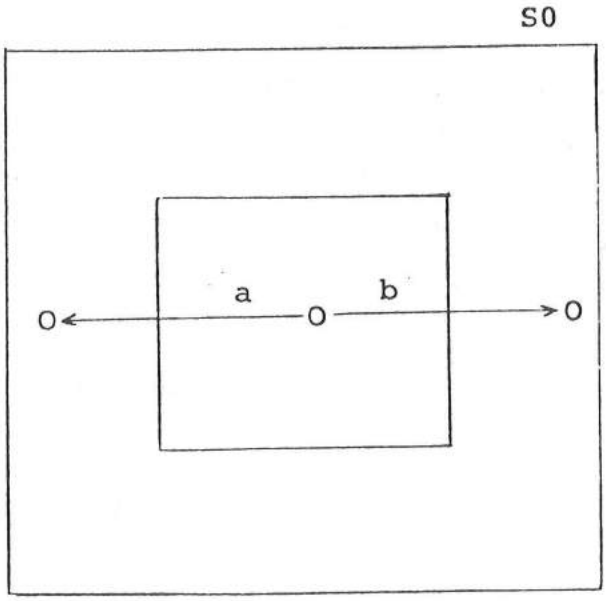

Figure 3e

Spaces may overlap -- objects in one space can also be contained in another. Hence, in Figure 3c, there could be a non-null intersection of spaces A and B. The intersections of spaces will be pointed out explicitly when they are of interest. However, not all of the above data structures will find uses in our repertoire of knowledge representations.

3.3.2 Typical Network Manipulations

The kinds of manipulations OSCAR performs on these data structures are: (The phrase "network entity" refers to any node, edge or space. "Object" means either a node or a space.)

- Creating nodes, edges, and spaces, subject to the above restrictions.
- Inserting a network entity that already exists in some space into a new space.
- Deleting a network entity from some space.
- Testing to see if an edge of a given label connects two objects, in a given space.
- Finding all objects at the ends of edges with some given label from some given object (in either edge direction).
- Testing if an object is contained in a given space.

The actual implementation of these data structures is not of general interest to the language problems at hand. The next section will present some samples of the encoding of predicate calculus statements into the network. I shall discuss the representation of: existentially quantified statements, negations, disjunctions, and universally quantified statements.

3.3.3 Existentially-Quantified Statements

The existentially-quantifed variables in a statement are encoded as network variables. The representation of $\exists x \in$ WOMAN s.t. (SPOUSE(JOHN) = x AND AGE(x) = 26) can be found in Figure 3f. The object marked as <X> is the network variable representing the woman. It has been stated to be

of TYPE WOMAN rather than an element of WOMAN since we maintain the convention that "E" (set membership) edges only point to distinct elements. Notice that the SPOUSE and AGE propositions refer to the same object.

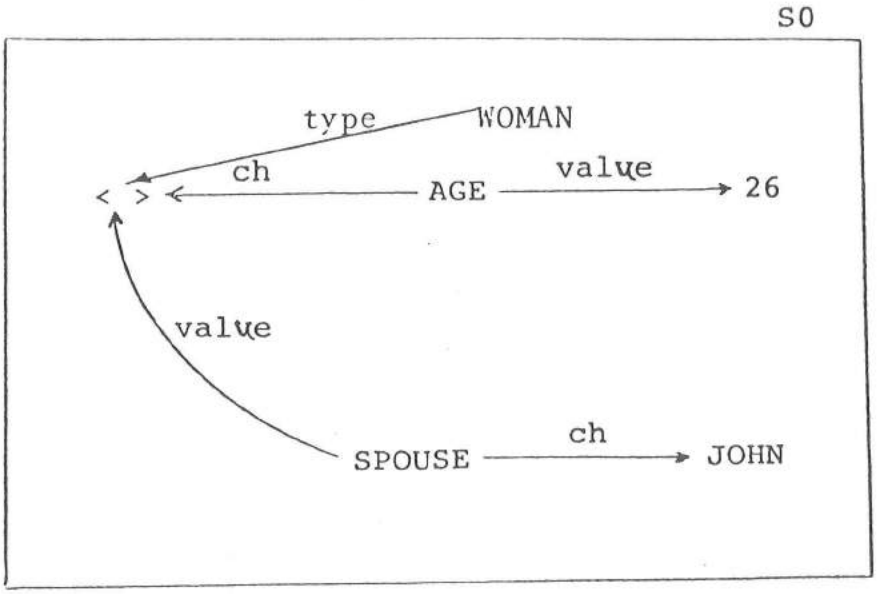

FIGURE 3f
REPRESENTATION OF "THE AGE OF JOHN'S SPOUSE IS 26"
("CH" stands for "characterizes")

A significant feature of all of the data structures pictured above is that all objects are contained either in a particular "top-level" space (S0), or in some space which is in the transitive closure of spaces embedded within S0. All these spaces carry implicit existential quantifiers. Hence any object contained within a space can be regarded as being existentially quantified with respect to that space.

3.3.4 Negation

Hendrix's suggestion for representing negation is to have a space that contains the conjunction of propositions that are not believed to hold in the enclosing space. Just as the outermost space is implicitly existentially

quantified, so too is a NOT space. Figure 3g shows the representation of NOT(AGE(JOHN) = 32), where the objects JOHN and 32 are not present in the NOT space. Notice that this is a legitimate memory configuration, as in Figure 3e. Since JOHN and 32 do exist in the space enclosing the NOT space, this representation states that it is not the case that there is an AGE relationship between JOHN and 32.

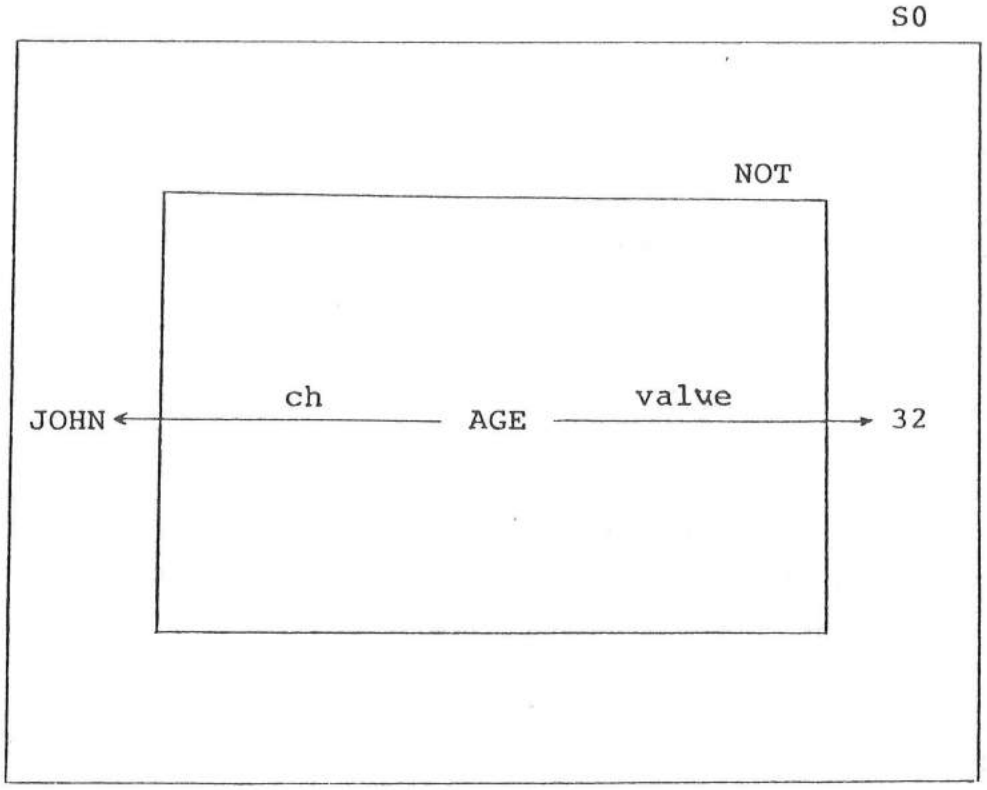

FIGURE 3g
USE OF A "NOT" SPACE

3.3.5 Disjunction

Spaces provide a valuable tool for encoding disjunctions since they enable one to separate the disjuncts from the enclosing space. This is desirable since we do not want to treat the disjuncts as true. The representation of "a or b" is that of Figure 3h, where a $\in$ A and b $\in$ B.

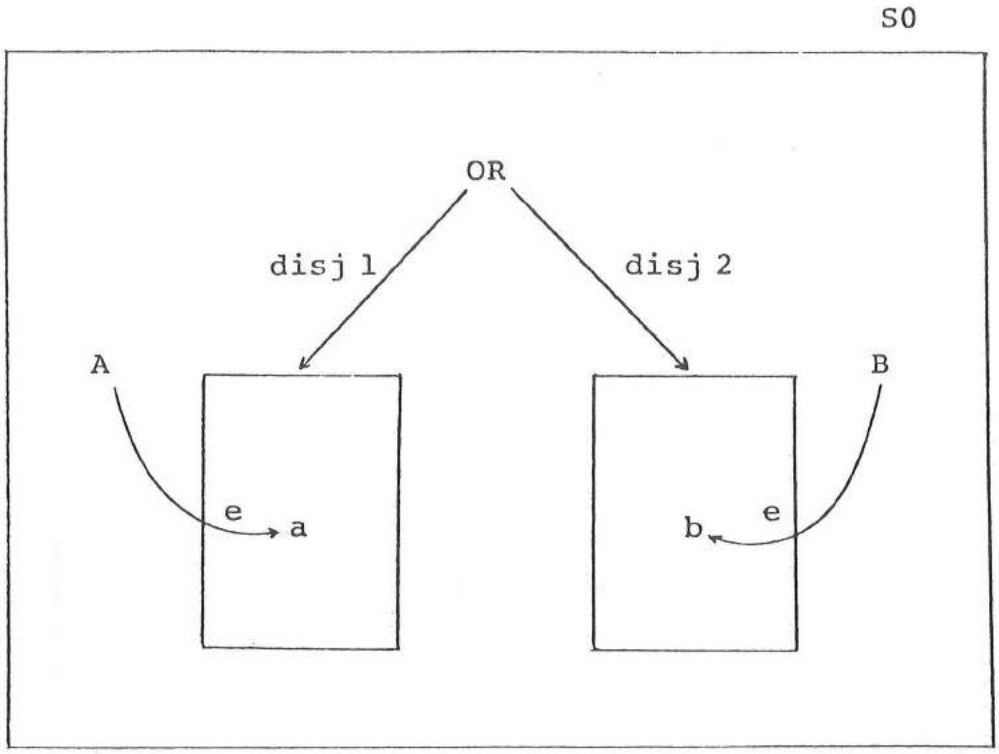

FIGURE 3h

REPRESENTATION OF "a or b"

Notice that the set membership arcs are contained in the disjunct spaces. Thus, as desired, when generating instances of A in the space S0, "a" will not be enumerated.

3.3.6 Universally Quantified Statements

Hendrix's [1976] representation solves the scoping of quantifiers' problem in a particularly nice way. His representation is based on the "implicit existential" property of spaces described above and on the observation that virtually every universally quantified statement can be placed in the form of an implication. (He has other, more complex representations that do not rely on this assumption.) Thus, ∀xP(x) can usually by written as ∀x(Q(x) => R(x)). For instance, the statement "Every employee of GM

has a relative who lives in Detroit", might then be represented in a logical form as:

$\forall x$ (Employ (GM, x) => $\exists y$ (Related-to (y, x) and live (y, Detroit)))

Within a partitioned semantic network, the above would become:

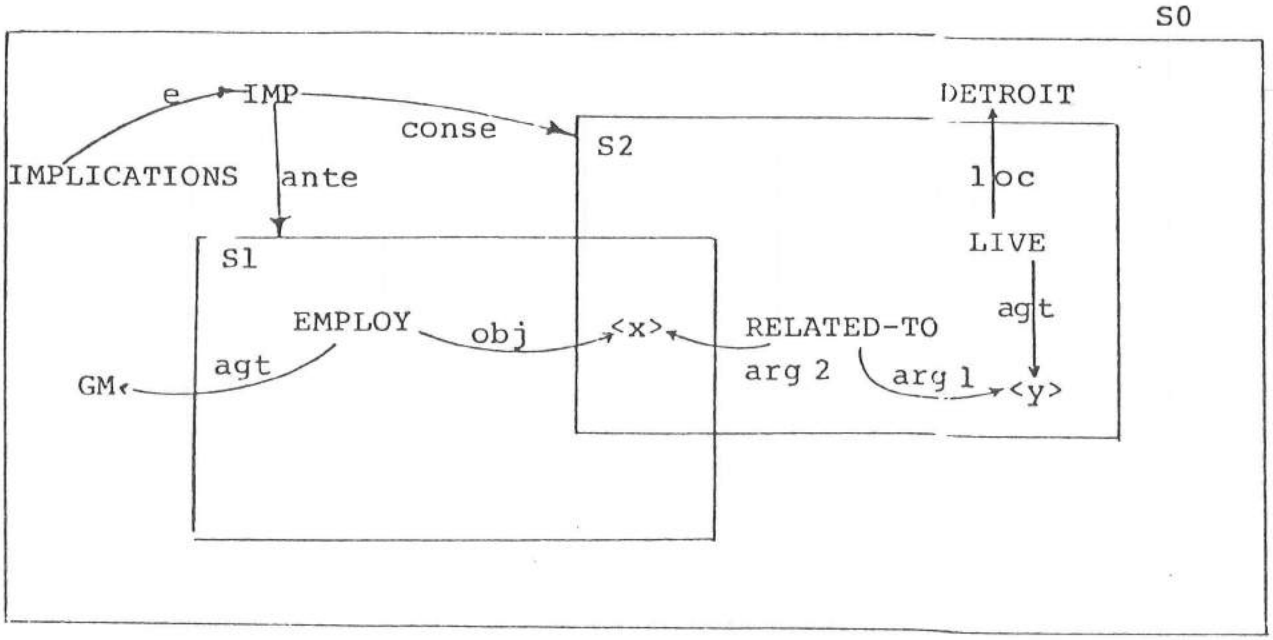

FIGURE 3i

EXAMPLE OF A UNIVERSALLY QUANTIFIED STATEMENT

This representation shows an element IMP of the class of IMPLICATIONS. IMP has ANTEcedents and CONSEquents encoded as spaces S1 and S2. The theorem states that if the configuration specified for the ANTE space holds in the enclosing space, the contents of CONSE are present in that space. The convention imposed is that objects within the overlap of ANTE and CONSE spaces are universally quantified. The remaining contents of ANTE and CONSE spaces are considered to be existentially quantified within the scope of the universally quantified ones. Notice that

implications are kept apart from the enclosing space (S0 here) and thus when searching for facts, one will not mistakenly discover theorems.

The reversal of the scopings above, as in
∃x(Employ(GM,x) and ∀y(Related-to(y,x) => Live(y,Detroit)))
would be represented in partitioned nets as:

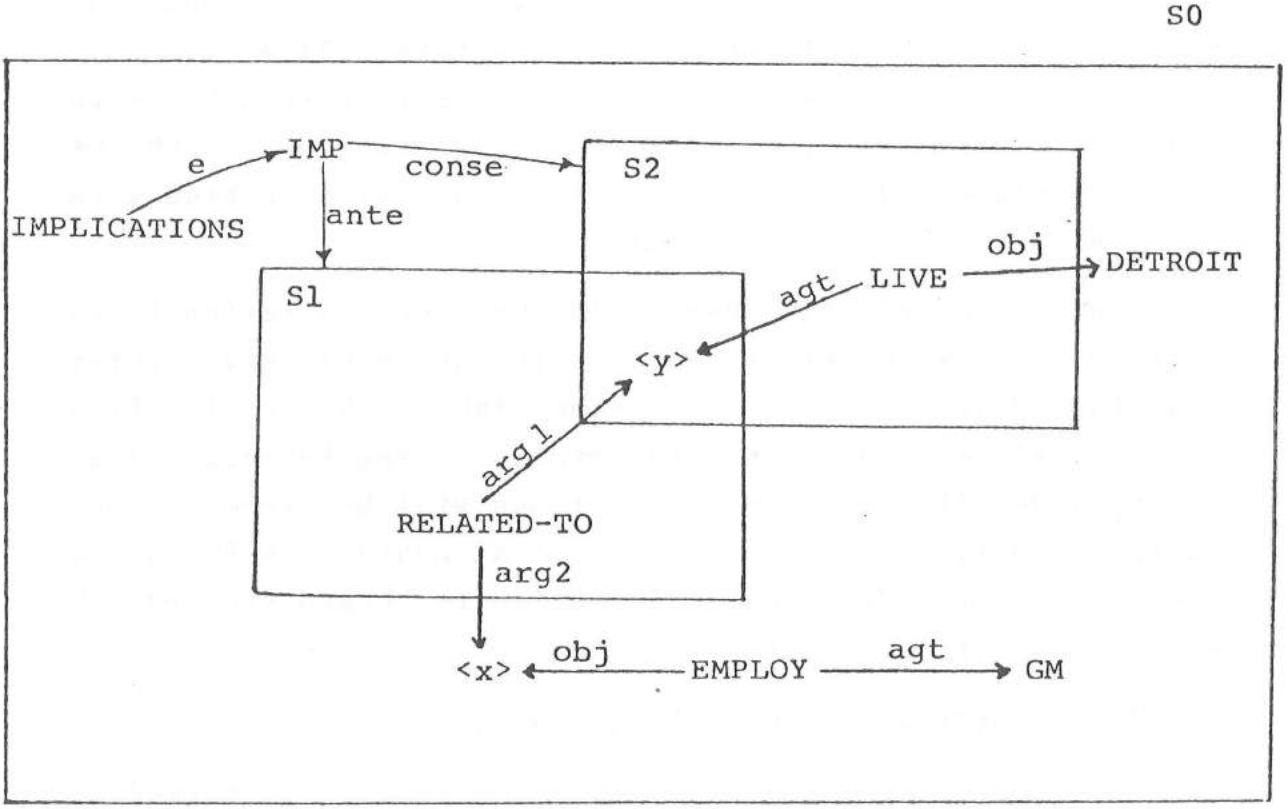

FIGURE 3j

AN EXISTENTIAL QUANTIFIER DOMINATING A UNIVERSAL

The existentially quantified variable here is placed in the space enclosing the statement since it is not dominated by a universally-quantified one.

3.4 The Modelling Scheme

The data structures described above can be viewed simply as a means of realizing more general implementation independent modelling schemes. The organization of knowledge described in this section is a simple version of Levesque's [1977] formalism (which was, in turn, adapted from [Abrial 1974]). The motivation for its structure is the desire to clarify the manipulations usually performed on semantic networks by global interpreters. The goal is to construct a communicable and modular knowledge representation that allows for rapid extensibility. To this end, an "object-centered" knowledge representation has been employed, where the operations to be performed on an object are associated (in a loose sense that will be made more precise) with the object itself.

The procedure for describing the representation is to introduce a new construct and give it an arbitrary syntax via the Model Definition Package (MDP). This will define the logical language that OSCAR encodes. The behavior of an entity under its associated operations will be presented and illustrated with MDP examples. To distinguish the formalism from the implementation, the former is termed the "model" and the latter "the network".

3.4.1 Objects, Classes, and Instances

Any referenceable entity in the model is termed an object. The model is viewed as a dynamic collection of objects, organized in certain ways that correspond to the model-builder's view of the domain of discourse. The model needs to be able to accomodate the addition and deletion of objects both as the user incrementally defines the domain and as the system operates. He can assign objects, via the MDP, arbitrary external names and thus have access to those objects during the construction of the model of some domain of discourse. The system, of course, manipulates the

objects themselves and thus knows nothing about any external names they might have.

A simple collection of objects offers no structure and thus cannot be considered to be much of a model of reality. To produce more heterogeneous models, additional means of structuring need to be introduced. First, we have the distinction between classes (e.g. PERSON, MAN) and instances, or elements, of classes. The operations associated with a class, fully specifying the behavior of its instances, are:

1. Testing if a given object is an instance of that class.

2. Adding an object as an instance of a class.

3. Deleting an object as an instance of the class.

4. Retrieving instances of the class.

This representation allows different classes to maintain their instances differently. For instance, some might be represented in a data base management system or perhaps on a different machine altogether. In practice, however, it has been found that the system, at the implementation level, is sensitive to the means of linking classes to instances. My feeling, though, is that we should strive to make such linkages transparent to the system's operation.

3.4.2 Defining New Entities

MDP gives the user the ability to create instances of classes by using the statement: [2]

(3.1) "X IS A NEW INSTANCE OF Y"

where "X" will be the external name of the newly created object and "Y" is the external name of the given class. The routine that is invoked by the interpreter when a statement of the form of (3.1) is entered is the program associated

with the class named by "Y", that asserts new instances of that class. (It is termed the "to add" program.)

We naturally want to extend our scheme to allow for subclasses. The MDP statement:

(3.2) "DEFINE X AS Y"

creates a subclass of the class whose external name is "Y" and assigns it the external name "X". Thus, any instance of "X" is also an instance of "Y". [3]

Once the model can contain arbitrary classes and subclasses (usually in, but not restricted to, the form of a tree), we must devise a way of determining if a given object is an instance of a given class. This is done by invoking the testing operation (called the "to test" operation) of the class.

3.4.3 Variables

The examples presented so far have dealt solely with objects that can be regarded as the equivalents of logical constants and predicates. The first step to take in order to approach logical adequacy is to represent variables. There are two different kinds of variables represented in OSCAR: "program-variables", that receive bindings, and "existential variables" that do not. Program-variables do not appear in the knowledge base of facts; they occur as formal parameters for various relations (and for programs written as network structures). As such, they will receive bindings when an operation on a relation is evaluated. Such variables differ from constants since the latter are their own bindings. Program-variables are given types that indicate the classes from which their values must be taken. The MDP statement:

(3.6) "X := A Y",

as illustrated by:

(3.6a) "SHOPPER := A PERSON",

creates a variable whose external name is "X" and whose value is to be chosen from the class "Y". Subsequent MDP statements can employ this variable by mentioning its external name (SHOPPER). At the time at which the network program is being defined, references to the external name of the variable refer to the variable itself. When the network program is executed, the binding of the variable is used.

Existential variables appear in the data base of facts (or goals). For instance, to say that "John is married", we would say:

```
THEREXISTS AN X SUCH THAT
    SPOUSE OF JOHN IS X
```

Such variables do not acquire bindings -- they are only used when one is enquiring about information in the data base, via a matching operation. (This will be explained later.) Hendrix's representation shows how we can avoid having to use explicit universally-quantified variables. Hence, all variables in the data base of facts are assumed to be existentially quantified with respect to the spaces in which they are contained.

3.4.4 Relations

The next structuring tool is the predicate. "Predicate" here is intended to mean an object that has a number of arguments. A proposition is a predicate with particular objects filling distinguished argument positions. A relation is a class of propositions, all of which share the same predicate. The following is an interaction with OSCAR in which three instances of classes are created, the last of which is a relation. (For the time being, only binary relations will be considered.)

(3.7) place is a new instance of class

[CLASS is the set of all classes.
PLACE is thus a class.]

(3.8) n.y.c. is a new instance of place

(3.9) loc is a new instance of relation

CASEFRAME:

(3.10) loc of (ch := a person) is (value := a place)

[LOC is an instance of the class RELATION, all of whose instances are themselves classes. MDP, realizing that since RELATION has arguments, LOC does as well, asks for a caseframe. This caseframe labels the program variables that are formal parameters to the LOC relation, as CH and VALUE. The order in which the arguments appear in this caseframe, and the words "of" and "is", indicate the syntax of LOC propositions.]

.
.
.

Finally, (3.11) LOC OF JOHN23 IS N.Y.C. is a proposition that is an instance of the relation LOC.

We must define the four associated operations in order to specify how instances of a class behave. However, since propositions have arguments, we cannot just use the standardly supplied mechanisms for manipulating instances of arbitrary classes.

3.4.5 Operations on Relations

The system provides standard operations designed to handle the most general cases of relations. When defining the operations for new relations, the user has the option to employ the standard operations, to employ operations of his own creation, or to construct composites. Thus, the behavior of the system can be as specialized as the user desires. However, he is not forced to specialize since the initial configuration of the system provides adequate, albeit simplistic, facilities.

This section will describe the kinds of operations available to the user. Only details necessary to understand

the examples of the system's operations presented in this thesis will be given. [4]

3.4.5.1 Testing

A typical operation that needs to be performed is to find out if a particular proposition is true. For instance, before planning to achieve any goal (say the proposition LOC OF USER IS INROOM), the system first tests whether it believes that proposition already. To do this, it employs the goal as a "conjecture", a proposition stating the form that matching propositions in the data base must take. It executes the program TEST on the conjecture, causing the testing operation associated with the type of the proposition being conjectured (the class LOC in this example) to be invoked by the interpreter.

The interpreter recursively evaluates a network program until it discovers a primitive encoded in the host programming language. The primitive is executed (possibly changing the data structure) and its results may become used in the rest of the computation. The interpreter is a version of the one described in [Norman and Rumelhart 1975] that has been modified and extended to deal with an object-centered representation and network spaces.

The standard testing operation for RELATION, called PROPTEST, is a program that searches for instances of the class to which the conjecture belongs and determines if any match the conjecture. Before dealing with matching, however, there are some preliminary topics to consider

3.4.5.1.1 The Dual Use of Spaces

At the implementation level, all network manipulations were defined to operate within some space -- a collection of network entities. When testing some conjecture, the interpreter needs to know to which space it should confine its search for matching propositions.

The space in which the searching is to take place is referred to as the world space (or sometimes as the world). Naturally, a conjecture should not be contained in the world space in which it is to be tested or it would be trivially true -- it would be its own answer. The conjecture must reside in some other context, that is termed the calling space. Typical calling spaces are those enclosing the definitions of relations. Each of the four class-defining operations must know both the calling space and the world space. In the text, when I omit the world space from a definition or invocation of an operation, the operation should be assumed to take place in the current world space.

In addition to serving in the role of restricting the objects that are visible to any manipulation operation, contexts, being classes, are objects in the model and can thus be part of propositions (see Figure 3b). The system is thus able to locate a space, via a connecting proposition, and then use it as a context to delimit searching. This dual function will prove crucial to the structure of the system.

3.4.5.1.2 Truth Values

A number of means for dealing with the incompleteness of the system's knowledge will be proposed. The first step to be taken is to base the system's inference procedures on a three-valued logic. The testing of any conjecture can return TRUE, FALSE, or UNK depending upon the contents of the world space. The object TRUE results from discovering a proposition matching the conjecture in the world space (or from inferences performed by the testing routine). UNK results when the system simply does not have enough information. The determination of a FALSE or UNK answer, however, depends upon the testing routine for the conjecture.

As an example, let us employ the definition of LOC above and assert the proposition that John is located at some particular place (call it INROOM) into the world space. When defining binary relations, the user is allowed to place cardinality constraints on the domain and range. Since LOC is defined as a functional relationship, the answer to TEST(LOC OF JOHN IS OUTROOM) is FALSE (where OUTROOM is an instance of PLACE and is different from INROOM). However, if we test LOC OF MARY IS OUTROOM, the truth-value returned will be UNK since MARY, an instance of PERSON, is known to have a location, but there is no proposition in the world space indicating what that location is believed to be.

SPOUSE is a different kind of relation than LOC and should be tested accordingly (perhaps one by municipal records and the other by spatial coordinates). Since PROPTEST could be employed as the testing routine for the more specialized relations SPOUSE and LOC, it is not capable of distinguishing between FALSE and UNK. Thus, if the conjecture is not believed to be true, PROPTEST returns the more general truth-value, UNK.

3.4.5.1.3 Three-valued Truth-tables

The truth tables for connectives in this many-valued logic are given below;

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

NOT	
T	F
F	T
U	U

Note that T OR U is T, F AND U is F, and NOT(U) is U. Not all tautologies for a two-valued logic hold when we have three or more truth-values. For instance, P OR NOT(P) is not a tautology since if P has the truth-value U, NOT(P) does as well. [5]

3.4.5.1.4 Outcome Propositions

The result of testing a conjecture is an outcome proposition, contained in a calling space, that relates the conjecture, its truth-value, and those propositions that led to the determination of truth. Outcome propositions are of three types: SUCCESS, FAILURE, and UNKNOWN. The caseframe for SUCCESS is:

```
success of (conjecture := a proposition)
           with (support := a proposition)
```

Thus, the value returned with a SUCCESS proposition is the proposition (perhaps a conjunction of them) that caused the testing program to judge the conjecture to be TRUE. Similarly the relation

```
failure of (conjecture := a proposition)
           with (obstacle := a proposition)
```

indicates those propositions that led to the conjecture's being judged FALSE. Finally, UNKNOWN simply has one argument -- the conjecture for which we lack the necessary information. These definitions are not exactly the ones used by the system (the actual ones also include a set of bindings to variables in the conjecture) but they are adequate for current expository purposes. [6]

3.4.5.1.5 Matching

Matching is a recursive process that compares a conjecture to a proposition. The matching algorithm presented below treats the existential and program variables of a conjecture in the same fashion.

Two propositions, a pattern and a datum, match if:

- when the pattern is a constant, the datum is the same constant.

- when the pattern is a variable with no binding, the datum is a constant or variable of the same type. (This case will create a temporary binding for the pattern variable.)

- when the pattern is a variable that has a binding, the datum is equal (EQ) to the object bound to that variable. In general, when two objects are compared for equality, unless they are identical, the result is unknown. For instance, in matching P(x,x) against the datum P(y,z) (where x, y and z are variables), x acquires y as a binding. The result of the match will thus be unknown since OSCAR cannot determine if y is equal to z. Of course, P(x,x) will match P(y,y).)

- the predicates, if any, are of the same type.

- for every argument of the conjecture, the corresponding arguments of the datum match.

This algorithm is not intended to be as sophisticated as those proposed by Moore and Newell [1973] and Bobrow and Winograd [1976] (although it will be extended later to capture some distinctions that they do not). Rather, the philoscphy of system design has been to demonstrate interesting behavior with simple tools.

3.4.5.1.6 Associating Operations to Relations: Inheritance

To accomodate flexibility in specifying the operations on classes, MDP is sensitive to the "defining properties" of classes. The determination of what are definitional versus incidental properties of an object is an age-old philosophical question to which I offer no solution. The design of OSCAR incorporates the view that operationally satisfactory results can be obtained by allowing such definitional properties. When we arrive at such properties

for a class, we can use them to determine if an object should be deemed a member. As we have discussed, this is the responsibility of the testing operation of the particular class.

To illustrate the user's capabilities to associate different operations to different relations, consider defining the class FUNCTION as a subclass of RELATION.

```
define function as relation
CASEFRAME:    function of (ch := an object) is
              (value := an object)
TO.TEST: inj.test of instance
```

The definition of RELATION includes a number of defining properties (or slots) such as CASEFRAME, TO.TEST, TO.ADD, TO.DELETE, TO.RETRIEVE, TO.GENERATE, etc. MDP examines the superclass' definition for slots and requests the user to specify new definitions for the corresponding ones in the subclass being defined. Consequently, the user is asked to supply a caseframe and the name of a testing program (among others). The definition of RELATION incorporates PROPTEST as the testing program. When the user is requested for his program to test FUNCTIONs, he has four options for a reply:

1. A reply of STD indicates that MDP is to use the "to test" of the superclass (RELATION) as the testing program of the current class.

2. A reply could indicate the new program to call (e.g. INJ.TEST)

3. A reply of NONE simply eliminates the defining property from the definition.

4. A reply of NULL leaves a dummy property in the definition. The only value in doing this is to enable future instances of this relation (themselves classes) to have such a property. When defining these instances, the

user will then be requested to supply specific values for the corresponding slots.

In the above example, MDP creates an INJ.TEST proposition inside the definition of FUNCTION. INSTANCE is a built-in parameter (a program-variable) that will become bound to any instance of the relation (e.g., the conjecture) that is then being tested, added, deleted, or retrieved.[7] By having an explicit parameter, we are able to use the STD mechanism to inherit defining properties from superclasses, which are parameterized by replacing the variables of the superclass' definition with the corresponding ones of the subclass.[8] Had the user responded with STD, he would have received the equivalent of PROPTEST OF INSTANCE where INSTANCE is the instance of FUNCTION and not of RELATION. We would now want to define LOC as an instance of FUNCTION in order to obtain the appropriate default operations.

The testing of propositions corresponds to classical logic in that testing discovers if a formula is an instance of an (non-logical) axiom or if the formula can be inferred from specialized rules. These rules, also axioms in classical logic, are encoded in the testing programs.

3.4.5.2 Asserting

At various times, new information is added into some context. Such occasions might typically be: creating an initial configuration of system and user beliefs, obtaining new information from a user's speech acts, adding new information into the user model as a result of the system's execution of a speech act. The maintenance of consistency in these contexts is the reponsibility of programs associated with the classes to which the asserted propositions belong. The "to add" program for LOC, for example, would accept a new location for some physical object and modify the previous location.

These programs are invoked by the interpreter when some other part of the program (or the user) executes a proposition of the form ADD PROP (in W), to assert the proposition PROP into world W. The propositions are already created before being asserted. Thus, the "to add" is not used to create instances, but rather to assert them.

An example of asserting new propositions is:

```
ADD ( AND ( (FASTENING OF DOOR IS LOCKED)
          (LOC OF USER IS INROOM) ) )
```

The interpreter, when presented with this proposition in the calling space, would evaluate ADD with its argument bound to the AND proposition, causing the "to add" program of AND to be invoked. This program evaluates the "to add"s of its conjuncts leading to the assertion of the FASTENING and LOC propositions, the asserting programs for which might have to change previous values in the world space.

The simplest "to add" program is PROPADD, an instance of which is attached to the class RELATION. It moves a proposition from the calling space to the world space without checking if there are previously asserted propositions of that class in the world space. Naturally, a large number of relations will employ more specific mechanisms for asserting instances rather than simply inheriting PROPADD. In particular, FUNCTION will use INJ.ADD to check the data base for previous values of the relation that should be discarded.

The definition of FUNCTION above, then, would continue with a prompt from MDP:
TO.ADD: inj.add of instance

Again, this proposition schema is now inheritable by instances of FUNCTION.

3.4.5.3 Retrieving

For a binary relation, retrieving involves evaluating, given the domain instance, the "to retrieve" program of the relation. Thus, if we have previously executed ADD (LOC OF JOHN IS N.Y.C.), RETRIEVE (LOC OF JOHN) would produce N.Y.C. as the value. The primitive RETRIEVE causes the "to retrieve" operation of its argument to take place. (For FUNCTION, the program is INJ.RETRIEVE.) For one-to-one functions, we can retrieve the domain instance as in RETRIEVE (LOC IS INROOM). Values returned from a retrieval can be constants, the special object nothing (e.g., when JOHN has no spouse) or a variable, when there ought to be a value but it isn't available. A retrieval of a relation like CHILDREN would cause a set of values (a space) to be returned. To obtain the elements of the set, one must then evaluate the "to retrieve" of the set (a standard, system-supplied routine).

The above pertains to values that are stored in the world space. Clearly, we may not want to store all values (such as the values of the factorial function) but would prefer to compute them as they are needed. To illustrate some of the possible combinations of operations that are available to the user, let us define the relation FACT.

3.4.5.3.1 Definition of FACTorial in MDP

The definition assumes a number of primitives, in particular OR, IF, MINUS, and TIMES. If the testing of OR's first disjunct is a SUCCESS, then OR returns without evaluating the second argument. Otherwise, OR forms an outcome proposition based on its truth-table and the outcomes of testing the disjuncts. The syntax of the primitive IF is:

```
IF   condition
THEN expression    /* if condition was a SUCCESS */
ELSE expression    /*  "     "      "  " FAILURE */
UNK  expression    /*  "     "      "  " UNKNOWN */
```

IF returns the expression appropriate to the outcome proposition resulting from the testing of the condition. MINUS and TIMES are primitives that do the obvious.

```
fact is a new instance of function
CASEFRAME:  fact of (ch := an integer) is
                    (value := an integer)
TO.TEST:        std or
                    (value eq (retrieve of instance))
TO.ADD:  std
TO.RETRIEVE:  /* retrieve explicit value */
              std or
                   /* compute value */
                   (if (ch eq 0)
                    then 1
                    else (ch times (fact of (ch minus 1)))
                    unk value)
```

In the definition of the testing routine for FACT, the second disjunct says to evaluate RETRIEVE OF INSTANCE. (Of course, INSTANCE is bound to the original FACT conjecture.) The retrieval program of FACT first checks the data base (via the STD retrieval program of FUNCTION, INJ.RETRIEVE) and if successful, returns the value. Otherwise, the IF

expression that recursively defines FACT is evaluated. When FACT is invoked recursively, the interpreter evaluates its argument and recursively executes the "to retrieve". This occurs because the recursive call of FACT is part of retrieving the value of the argument to TIMES.

3.4.5.4 Deleting

The primitive DELETE, when evaluated on some proposition in some space, causes the "to delete" operation of that proposition to be invoked. The simplest deletion program is the primitive PROPDELETE, used by the class RELATION, which erases all traces of its argument. Other classes are free to have more specific rules for deletion. (For instance, deleting the fact that some living person is male should cause an assertion that that same person is female.) Numerous issues regarding consistency arise and I do not pretend to have solved them. A major issue is whether we want to use a deletion operation that does not allow us to recover old states of the data base. One can propose ways of keeping earlier states, using what Hendrix [1976] terms a "visibility lattice" of spaces. However, OSCAR takes the simplest approach and uses PROPDELETE.

3.5 Summary

The top-level concepts of the model developed so far are summarized in Figure 3k. (Dashed arrows are subset links while solid arrows are "element-of" links.)

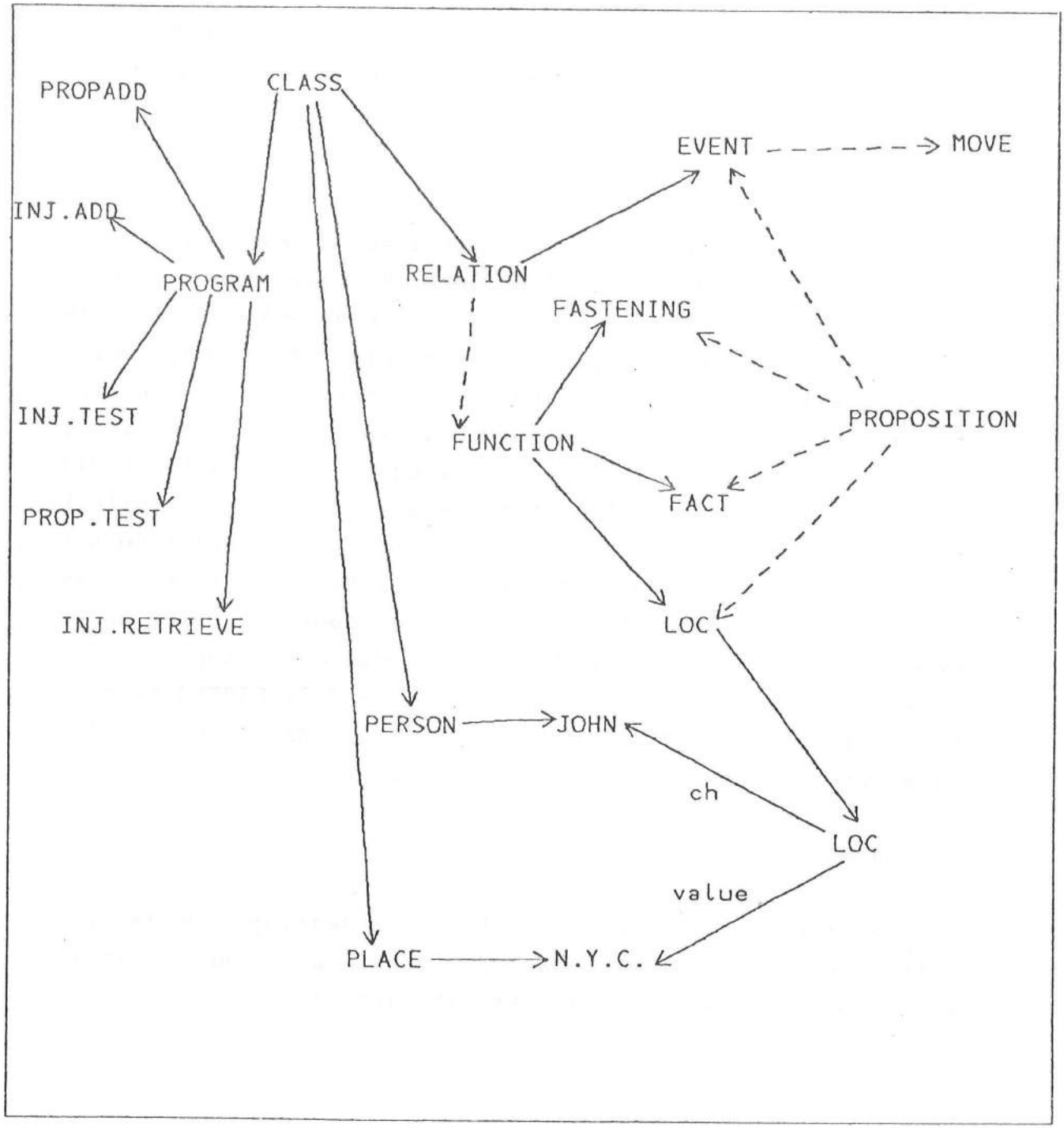

FIGURE 3k

SOME OF OSCAR'S TOP-LEVEL CONCEPTS .

RELATION, PROGRAM, and FUNCTION are (meta) classes whose instances are themselves classes. FUNCTION happens to be a subclass of RELATION indicating that instances of FUNCTION, e.g., the classes FACT and LOC, are instances of RELATION. However, LOC is a subclass of the class of PROPOSITIONs (as is FACT) since any instance of LOC is a proposition. This is a case where a class is an instance of one superclass and a subset of another.

The classes, PROPTEST, PROPADD, PROPDELETE, INJ.TEST, and INJ.ADD are instances of the (meta) class PROGRAM. Instances of the program INJ.ADD are found as defining operations for LOC and FACT since they were inherited from the superclass FUNCTION. The class EVENT (defined in Chapter 5), being an instance of the class RELATION, inherits the testing and adding programs of RELATION. This allows the class MOVE, a subclass of EVENT, to use these same programs. Again, EVENT is a subclass of PROPOSITION. Finally, the proposition LOC OF JOHN IS N.Y.C., is an instance of the class LOC where JOHN and N.Y.C. are elements of the classes PERSON and PLACE respectively.

The modularity and clarity of the approach derive from the ability to use default, and hopefully secure, programs to manipulate various classes. For instance, in the model proposed above, the user need not directly concern himself with LOC or FASTENING. He can simply choose to model them as injective functions with certain specific domains and ranges. He is thus able to create models of domains of discourse that ignore uninteresting details. Subsequent changes to the manipulation routines for a class are performed in a disciplined fashion and will be transparent to the rest of the system. This allows OSCAR to maintain a certain degree of data independence.

With a small number of manipulation routines doing the bulk of the work, we have a chance of establishing that they each leave the data base in a consistent state. Future

research should be addressed to this question, perhaps following the methodology of research in abstract data types (see, for instance, [Guttag 1976]).

The next chapter will describe the means for segmenting the classes and their instances into various contexts by using the primitives BELIEVE and WANT.

3.6 Notes

1. The viewpoint that words refer to mental concepts and not directly to objects in the real world is implicit in most network representations in the literature. My assumptions will be discussed in Chapters 4 and 9.

2. Anything the user types to OSCAR will be printed in lower case while OSCAR's responses will be in upper case. MDP statements in the text, like the above, will be printed in upper-case.

3. To avoid having to continually qualify all references by "...whose external name is Q", I will adopt the convention of using upper-case external names to refer to the respective objects in the model.

4. When displaying propositions in their network form, edges will be labelled with the name of the appropriate formal argument of the relation. However, these names (cases) do not impact the rest of the system. Only the manipulation programs of that class need to know what the edge labels are.

5. For an excellent survey of many-valued logic, see [Rescher 1969], where this logic (originally due to Kleene) is described.

6. A similar concept is that of the "residue" as discussed in Srinivasan's [1976] MDS system.

7. What is created here is not exactly a proposition. It uses the program variable INSTANCE and it is not contained in the data base of facts, but rather in the definition of a class. In the future, this kind of "proposition" will be termed a <u>proposition schema</u>.

8. This inheritance takes place at definition time in the current system. This is different from Levesque's formulation where he allows for inheritance at run-time. [Levesque 1978] and [Levesque 1977] give a more careful treatment of the issue of inheritance, in the context of definitional and assertional properties.

CHAPTER 4

USER MODEL AND MEMORY ORGANIZATION

Contents:

4.0 Introduction

A goal of this research is to show how what we say in conversation is influenced by our knowledge of our conversants. This chapter examines the system's means for representing such knowledge.

There were two independent principles used in designing this system, structural similarity and solipsism. Structural similarity is the belief that the organization of the mind is essentially the same throughout our species. Accordingly, OSCAR's models of other people have the same structure, but certainly not the same content, as OSCAR's model of itself. This principle is employed at all levels, and hence the model that the user is believed to have about OSCAR has the same structure as OSCAR's model of the user. However, OSCAR's model of itself need not be the entirety of OSCAR--for instance, its model of its "mental" processes is different from the set of actual processes that comprise it.

With this in mind, the system was designed from an "egocentric" standpoint -- from OSCAR's view of the world. This approach is appropriate when one's goal is to construct a process model that can be activated to perform in real situations. Such a modelling strategy is motivated by a long-range desire to have a psychological theory that could be used to explain how people operate based upon their perceptions. The opposite approach, one I attribute to logic, is to describe people from the "outside". In this vein, we see some researchers in Artificial Intelligence, following philosophers of logic, attempt to characterize "knowledge" by incorporating both concepts in the mind and objects in the real world into their formalisms. My point is that it is not necessary that our models directly include such characterizations. Rather, the cognitive processing models we produce ought to be such that they can be described by logical means to the same extent as those means characterize human behavior. Logical descriptions would then serve as standards rather than as mechanisms.

4.1 Belief, Knowledge, and Solipsism

The distinction that many philosophers make between belief and knowledge is based upon there being a notion of objective truth, i.e., of there being a reality. This is evident in their definition of "knowledge" as belief that is true and justified (i.e., true for the right reasons). The concept of "know" is then of a comparative nature: to determine if someone knows something, one compares that person's beliefs against what is true in the world. What troubles me is the concept of "truth". Our process models, and perhaps people, cannot make direct comparisons with reality, only to beliefs about reality. There may in fact be a reality, but following the above, our models can never "know" anything about it. Hence, I claim "belief" is the concept upon which we want to build models of cognitive processing.

This is not meant to be idle metaphysical speculation, but rather this "solipsistic principle", that would stipulate that one operates solely on the basis of mental models, is essential to the system's structure. In OSCAR's case, this principle would say that OSCAR can only manipulate symbols, i.e., mental objects, some of which correspond to real world entities. This does not deny the possibility of gaining information about reality via sensory apparatus. The point here is that no matter how good the sensory apparatus is, the system must still rely on beliefs.

Memory is thus structured via "beliefs", propositions contained in some appropriately defined space, and "wants", what the system believes its goals to be. I will show how these are handled using the data structuring tools of Chapter 3. In addition, in order to answer some criticisms of precursors to this approach, I present a representation for negated and disjunctive beliefs. Discussions of quantified and mutual beliefs will be deferred to later chapters.

4.2 Representing Belief

Beliefs are created by asserting BELIEVE propositions of the form:

(agt := a person) believe (obj := a proposition)

When instances of BELIEVE are asserted, the proposition bound to the variable OBJ will be placed into an appropriate location in memory. The tool used to structure the memory to facilitate this is the "belief space".

The facts that the system believes are contained, as an implicit conjunction of propositions, within its "belief space". This space circumscribes the totality of OSCAR's memory, thus enforcing the solipsistic principle. In that

sense, the system's belief space, labelled "SB", also contains all the objects related by propositions within SB. For instance, JOE, 32, and AGE OF JOE IS 32 are objects that would be contained in SB. Absence of a particular proposition from a belief space does not commit the system, one way or another, to the truth value of that proposition.

In addition, a number of spaces are contained within SB. Such spaces correspond to the system's goals, theorems, negations, and hypothetical worlds such as those created by novels and plays. It is important to realize, however, that there can be objects contained within those embedded spaces that are not contained in SB. Those objects would be interpreted differently based upon the interpretation of the containing space.

4.2.1 Identifying a Belief Space

A space is a "belief" space if it is an argument to a BELIEVE predicate. Hence, the semantics of a space are dependent upon the connections that space has with other entities. Figure 4.2a shows a belief space for the object S. Any proposition inside the belief space is thus interpreted, by routines to be described later, as a belief. For instance, asserting the proposition AGE OF U IS 26 would result in the representation pictured in the figure.

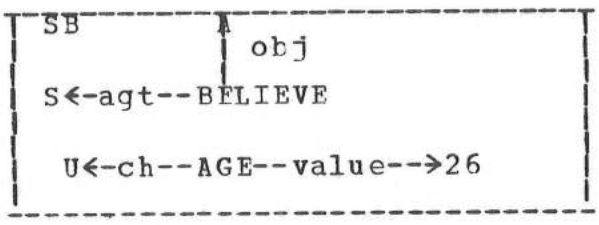

FIGURE 4.2a

S's BELIEF SPACE CONTAINING A PROPOSITION REPRESENTING "U's AGE IS 26."

4.2.2 OSCAR Believes that it Believes

One of the implications of saying the system's model of reality is solely via beliefs about mental objects is that there is nothing within the system that is "outside" the system's belief space. This fact and the fact that we need a proposition to identify the space as a belief space, jointly lead us to represent the identifying proposition for "system believes" inside the belief space itself, as in Figure 4.2a. The object S is the system's conceptual entity representing itself. Such a proposition referencing the space in which it is contained gives us a representation for "believing that one believes". Essentially, this representation states that "If S believes P, then S believes that he believes P."

4.2.3 Memory Organization

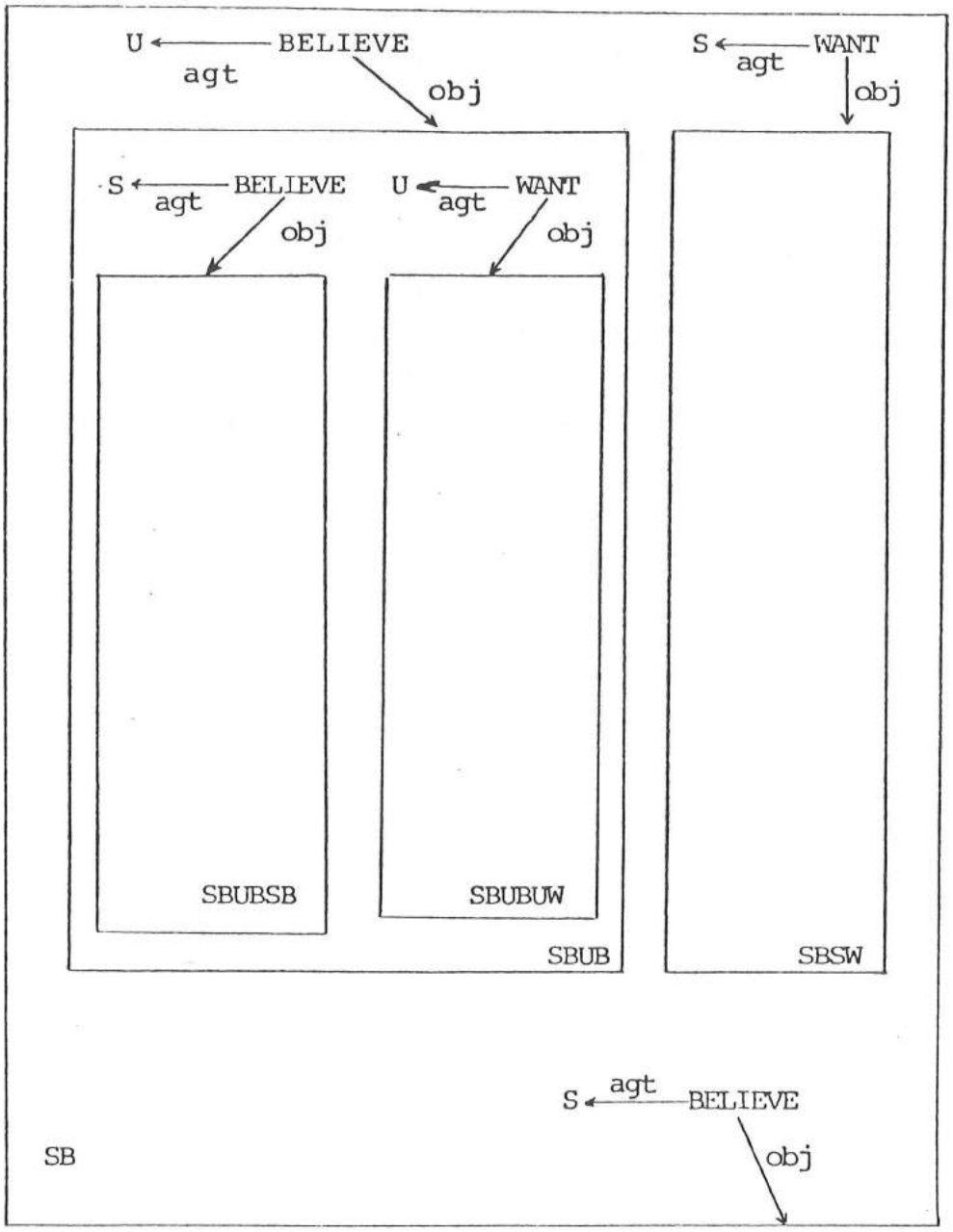

FIGURE 4.2b

SCHEMATIC OF OSCAR'S MEMORY STRUCTURE

Figure 4.2b is a schematic of the organization of memory, the top level of which is the space of system

beliefs, SB. SB contains a proposition, identifying the system's want space. Hence this space represents what the system believes it wants, and is labelled "SBSW". WANT and BELIEVE propositions, and associated spaces, can be created in SBSW. SB also contains propositions identifying a space, labelled "SBUB", of what the system believes the user believes, where the user is represented by the object U.

4.2.4 Embedded Belief Spaces and Their Intersections

The solipsistic and structural similarity principles indicate that the arguments to any belief proposition, for some person, are contained within that person's belief space. In further examining the structure of SBUB, it should be noted that, again by the structural similarity principle, space SBUB contains a proposition that references space SBUB. Consequently, not only does the system believe that it believes, but it also believes the same about its user. Thus SB(p) => SBSB(p), SBUB(p) => SBUBUB(p), and so on.

From the discussion in Chapter 3, it should be clear that belief spaces can overlap and thus share information. In Figure 4.2b, all references to the object S are to the same object, but that object is contained within both SB and SBUB. In fact, all constants that the system believes the user knows about (i.e., constants like JOHN, 32, and N.Y.C., in SBUB, and in all other embedded belief spaces) must also be resident in SB since, of course, SBUB is the system's model of the user.

Since searching routines are confined to particular spaces, there is no danger that a proposition solely contained in an embedded space will be seen when OSCAR is operating with respect to another. Thus OSCAR can believe that the user and it attribute different properties to the same object. OSCAR can see the user's viewpoint of some object by operating with respect to the space SBUB.

If we create SBUBSB, we can obtain the set intersections of the spaces in Figure 4.2b as indicated in Figure 4.2c.

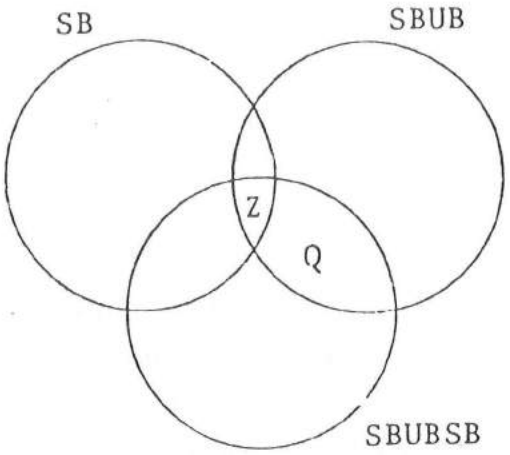

FIGURE 4.2c

INTERSECTIONS OF BELIEF SPACES IN FIGURE 4.2b

Proposition Z is maximally shared knowledge, while proposition Q is what U believes, according to S, to be shared knowledge (as would be the case if S intentionally lied to U).

Notice that we have represented, in the memory structure of Figure 4.2b, three levels of belief embedding. We could, in principle, continue to represent more levels, but how many do we really need? By defining the semantics of BELIEVE appropriately, we can show that any number of levels can be easily handled. What we will do is to create an adaptive representation that will supply new memory levels when needed.

4.2.5 Operations on BELIEVE

The operations to be defined here include: testing to see if some proposition is a belief, adding a new belief, and retrieving beliefs. The deleting of beliefs is analogous to adding them and thus will not be discussed separately.

4.2.5.1 Testing Beliefs

In general, executing "TEST of P" causes the "to test" operation associated with P's generic class to be invoked. If P is a BELIEVE proposition, then it is of the form "AGT BELIEVE PROP" where AGT is a variable bound to the person whose beliefs we are checking, and PROP is a variable bound to the rest of the conjecture in which we are interested. All operations are performed with respect to some world space W, usually a belief space. BELIEVE's testing algorithm then is:

```
BELIEVE.TOTEST:

    Find the belief space B of AGT in W
    If none, then return an UNKNOWN proposition, with
         PROP as the conjecture.
    Else If TV, the outcome proposition resulting from
              TEST OF PROP in B, is a SUCCESS proposition,
         Then Return TV
         Else, If TV is a FAILURE proposition
                Then return TV.
          Otherwise, /* TV was an UNKNOWN proposition. */
               Return the reversal of the truth-value pro-
               position returned from TEST OF NOT(PROP) in B.
```

This algorithm simply retrieves the AGT's belief space and uses it as the new space in which to continue future testing. Of course, if PROP is another BELIEVE proposition, the above algorithm is called recursively. Clearly OSCAR, given a proposition whose nesting complexity is higher than

that of the current memory, will eventually run out of memory levels and return UNK.

The above testing procedure also checks NOT spaces inside the agent's belief space to see if the agent believes NOT(PROP). If so, the outcome proposition returned from this test must be reversed (FAILURE --> SUCCESS, and vice-versa) since what we were originally interested in was the truth of P, rather than NOT(P). Belief spaces within NOT spaces can be employed as well. For instance, we may test if SBUB(P) and discover that SB(NOT(UB(P))), and hence return a FAILURE proposition.

Relations (e.g., LOC) are free to perform whatever specialized inferences are necessary. Nothing prevents OSCAR from performing different inferences in different belief spaces for the same class. Its model of what the user believes about LOC could be completely different from its own LOC and hence different testing programs could be called.

The testing routine can easily be adapted to deal with beliefs tied to roles (e.g., plumber, carpenter, shopper) in a script. (See [Schank and Abelson 1975] for a discussion of scripts). In addition to looking in the AGT's belief space, the testing routine would check belief spaces associated with the roles to which the agent is currently bound. Thus, when the user is known to be a shopper, typical beliefs ascribed to shoppers (including a shopper's beliefs about stockboys) are available.

Fikes and Hendrix's [1977] theorem-prover, SNIFFER, employing natural deduction and case analysis as rules of inference, could be used to augment OSCAR's deductive capabilities. However, they are only concerned with first-order logic and hence, SNIFFER could not be directly used to prove theorems "across" belief (or want) spaces. However, within one belief space, it appears as though SNIFFER is

applicable. The "to test" of the class BELIEVE might then be an evaluation of the standard means for testing beliefs, conditionally followed by the more general deductive machinery.

4.2.5.2 Adding New Beliefs

At any point, our memory structure has a certain, hopefully small, number of levels. In Chapter 8, I will argue that the most deeply embedded level may be more complex. However, for simplicity, all levels of the belief nesting will be treated as identical. Since each relation is in charge of maintaining consistency for its instances, all that BELIEVE need do is to create and locate appropriate spaces.

The way to maintain an adaptive representation when adding a belief for an AGT into a space that does not exist, is to create a new belief space. The algorithm goes as follows:

```
BELIEVE.TOADD:

    Find the belief space B of AGT in W.
    If none,
    Then Create a space B in W.
         Create the proposition AGT BELIEVE B in W.
         Put that proposition in B as well
         (for AGT's believing that he believes B.)

    ADD PROP INTO B
```

The "to add" program of the proposition bound to PROP then handles the rest of the additions into belief space B. Multiple levels of belief will be asserted, by a recursive application of the above algorithm, when PROP is another BELIEVE proposition. The routine for deleting beliefs proceeds by finding AGT's belief space and calling the "to delete" operation on PROP in that space.

4.2.5.3 Retrieving Beliefs

Two cases need to be handled here. If nothing is specified for PROP, then the AGT's belief space is retrieved from the current world and returned. Otherwise, we evaluate the retrieval program of the proposition bound to PROP in the AGT's belief space, and pass the result back. An example of this would be
RETRIEVE (USER BELIEVE (LOC OF SYSTEM))
PROP would be bound to LOC OF SYSTEM, and LOC's retrieval program would be invoked, returning the appropriate object.

4.3 Representing Want

Since OSCAR is intended to be a purposeful system, it needs some way of representing its goals. Again, by our structural similarity principle, any representation mechanism for the system's goals should be extendable to those of the user. I will present a mechanism that, combined with the belief space approach above, satisfies these requirements.

The major problem in representing "want" is that of scoping. Most other representations of "want" in natural language systems, e.g., [Meehan 1977], have avoided the issues that arise in developing a representation scheme that handles both goals and beliefs. Meehan simply represented "want" outside of the knowledge representation altogether by employing a special "goal stack". Hendrix's use of spaces for "wants", on the other hand, allows one to use the standard representation machinery to segregate goals into appropriate contexts, thus preventing searching routines from mistakenly treating goals as beliefs.

A proposition is identified as being a goal by its presence in a "want space". All want spaces are enclosed in belief spaces. For instance, the system's want space (SBSW) is enclosed in the belief space SB. Similarly, within the

user's belief space, SBUB, we have a want space for the user, SBUBUW.[1]

The presence of a proposition in a want space for some agent is meant to capture the agent's wanting that such a proposition exist in the enclosing belief space. The arguments to propositions inside a want space that are constants are not themselves contained in the want space, but are already contained in the enclosing belief space. For instance, in Figure 4.3a, S's goal of block36 being red is represented as a proposition in S's want space (SBSW). Neither the object "block36" nor "red" is present in SBSW, implying that what is wanted is that there exist a COLOR relationship between them. Extensive examples of goals will be demonstrated in the next few chapters.

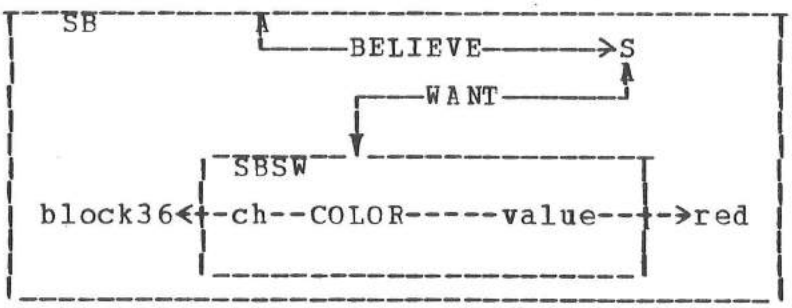

FIGURE 4.3a

REPRESENTATION OF "SYSTEM WANTS THE COLOR OF BLOCK36 TO BE RED."

The scoping capability of want spaces, as pointed out in [Hendrix 1975], enables us to distinguish between the specific and non-specific readings of certain types of sentences. Thus, "John wants to marry a doctor" has two representations in Figures 4.3b and 4.3c corresponding to there being a specific doctor, say Joan, he wants to marry or the reading corresponding to whomever he wants to marry (represented by a variable) must be a doctor. When a variable is present in the want space, what is wanted is that there be some object in the enclosing belief space satisfying the desired expression that defines that

variable. The difference in readings is representable via a difference in whether the object representing the doctor is inside the want space (the non-specific reading) or outside it in SB (the specific doctor).

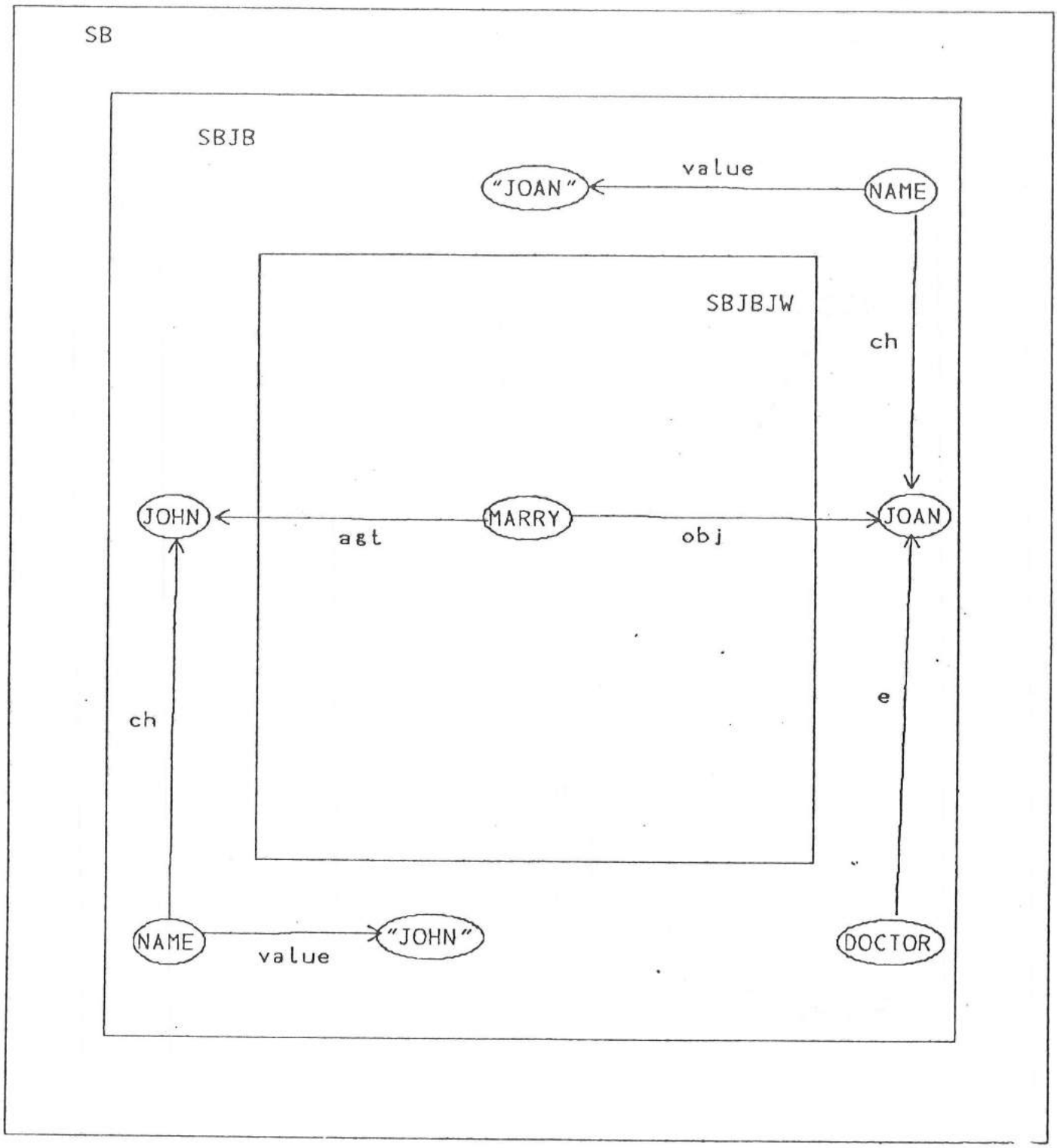

FIGURE 4.3b

REPRESENTATION OF "JOHN WANTS TO MARRY JOAN, A DOCTOR."

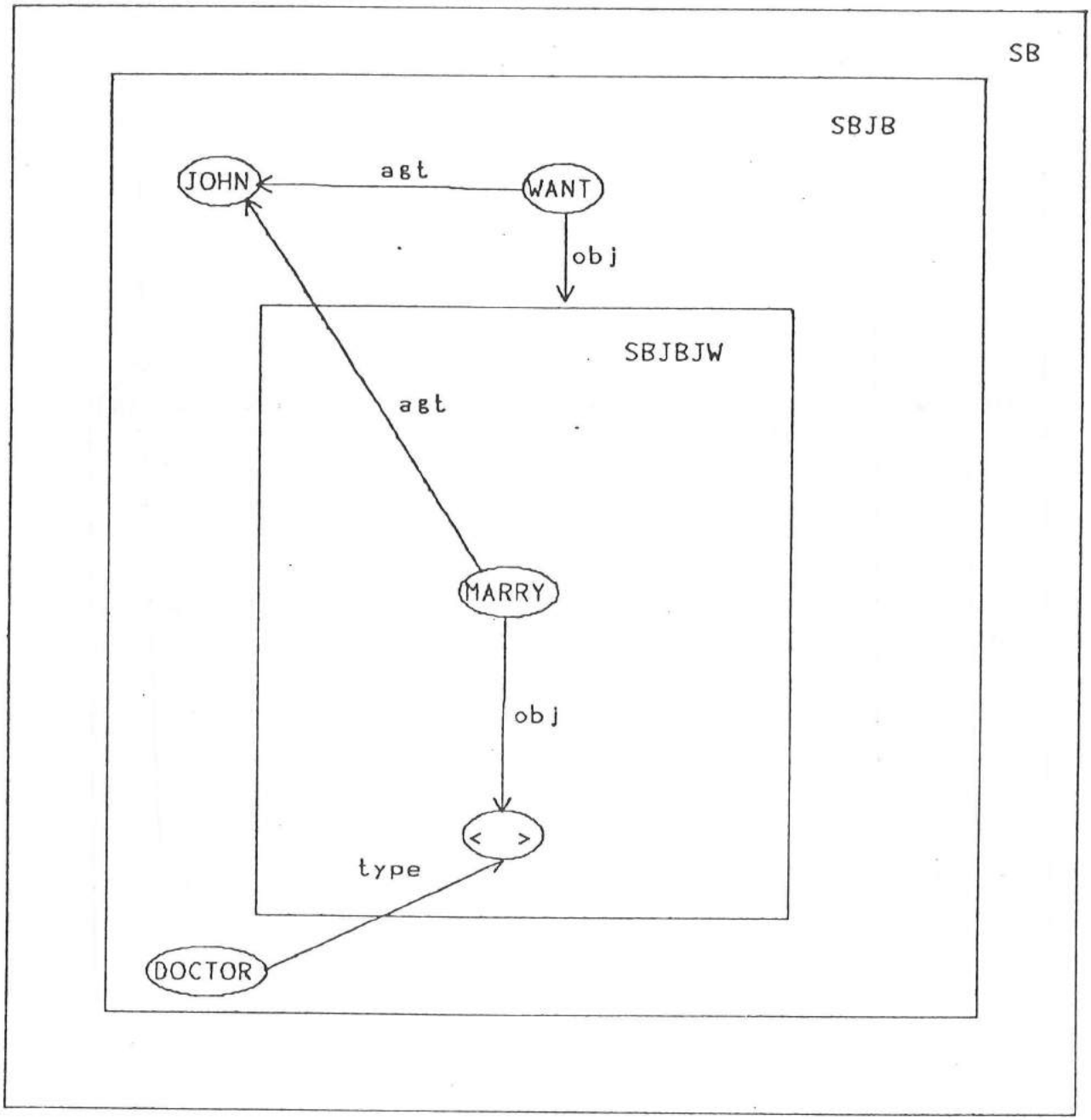

FIGURE 4.3c

REPRESENTATION OF "JOHN WANTS THAT WHOEVER HE MARRIES BE A DOCTOR."

While this representation mechanism can handle the differences between the two readings, some sophisticated linguistic component must decide upon the preferred one. A more complete treatment of this problem, dealing with such statements as: "John wants to marry a [specific] doctor, but I don't know who she is", can be found in chapter 7.

In order to specify the program's semantics of "want", I need to discuss how WANT propositions are manipulated using this memory structure.

4.3.1 Testing WANT

Again, we must specify routines to determine if some proposition is a goal. The caseframe for WANT is
(agt := a person) want (prop := a proposition)
The TEST algorithm, parallel to BELIEVE.TOTEST, is as follows:

WANT.TOTEST:

```
Find AGT's want space, WS, in world W.
IF V, the proposition returned from testing
     PROPTEST OF PROP IN WS, is a SUCCESS
Then return V
Otherwise, /* it was an UNKNOWN proposition since */
           /* PROPTEST cannot return a FAILURE */
     Find NOT space, N, in W.
     Call WANT.TOTEST recursively on PROP in N.
     If the proposition returned was a SUCCESS,
     Then return a FAILURE proposition.
     If the value was an UNKNOWN proposition, then
     return it.
```

Inferences about goals are even more difficult than those about beliefs. For instance, suppose we know that John wants some proposition P to be true. Suppose, in addition, that we believe that John believes that P => Q. Can we use this implication in the form of "modus ponens" if

we want to know if John wants Q? The difficulty, of course, is that P could "imply" any number of propositions, any one of which may have been the goal that John had in mind. Perhaps he had none of them in mind at all, and simply started his planning at P. John might even believe Q is already true. For instance, we could say "if something is a cat, then it is an animal." If we know John wishes his birthday present, a dog, were a cat, are we then to say that John wishes his birthday present were an animal?

Even more tenuous "inferences" would occur when P and Q are the preconditions and effects of some action. We might justify the use of "want" inferences provided that the chain of goals ended in a goal we believe John already wants. This is a possible basis for a plan recognition algorithm.

These issues are avoided entirely by not allowing any such "inferences" to take place in want spaces. This is enforced by only calling PROPTEST to see if there is a proposition identically matching PROP in the want space. Had the standard testing program of PROP been invoked then unconstrained inferencing would have taken place. In addition, when the routine above searches for the AGT's wanting PROP <u>not</u> to be true, the "to test" of NOT is not invoked since it would then call the "to test" of PROP and perform various inferences. The algorithm above simply checks for an identical match in NOT spaces.

It should also be noted that WANT does not check for propositions of the form NOT(JOHN WANT P). Since want spaces are enclosed in belief spaces, we are nearly always testing propositions of the form AGT BELIEVE AGT WANT P. Testing NOT(JOHN WANT P) would be done by the testing program of BELIEVE.

4.3.2 Adding WANT

The algorithm for asserting new goals is:

Find the AGT's want space, WS, in W
If none, create one, WS, in W
PROPADD OF PROP into WS.

PROPADD, as we have seen, simply transfers the proposition bound to PROP into the want space. If there is no such want space, one is created.

By calling the program PROPADD directly, we bypass any special heuristics that some classes may employ when adding their instances. This is done since, just as we did not want inferences to take place when testing "want", we do not want the standard update processing (analogous to antecedent theorems in PLANNER) that would occur in a belief space to occur in a want space. What would appear to such routines to be "contradictory" information could be legally contained in want spaces since those propositions would not necessarily be simultaneously true in the enclosing belief space. For instance, actions may change the value of someone's location from one place to another, but the person's starting and ending positions may both appear as goals in the want space. The reason for this is that the plans in a want space are sequences of modifications to belief spaces.

4.3.3 Retrieving WANT Propositions

The means for retrieving WANTs are identical to those of BELIEVE. We return either the AGT's want space or the "to retrieve" of PROP in the AGT's want space.

4.4 Some Problematic Constructs

Moore [1977] claimed that an explicit "data base" approach (i.e. having a separate data base for a person's beliefs) could not represent any of the statements: "John

doesn't believe P", "John believes P or John believes Q", or "John knows Mary's location". (In the case of the disjunction of beliefs, Moore claimed that the only representation he could devise within the "data base" approach would entail an exponential growth in the number of alternative data bases of John's beliefs.) These inadequacies, he claimed, partly lead him to search for a different approach to representing "know", and hence "believe". Unless Moore is referring solely to CONNIVER-style (see [Sussman and McDermott 1972]) databases, (which in recent years have not been used as representation schemes in natural language systems), this criticism of representational adequacy is inappropriate.

4.4.1 Negation of Beliefs

We can represent the belief that propositions are false using the techniques of Chapter 3, by placing them in a "NOT space", itself inside a belief space. The extension to representing "it's not the case that the user believes P", shown in Figure 4.4a, should now be obvious. Hence, part of what OSCAR believes to be false includes beliefs about the beliefs of others.

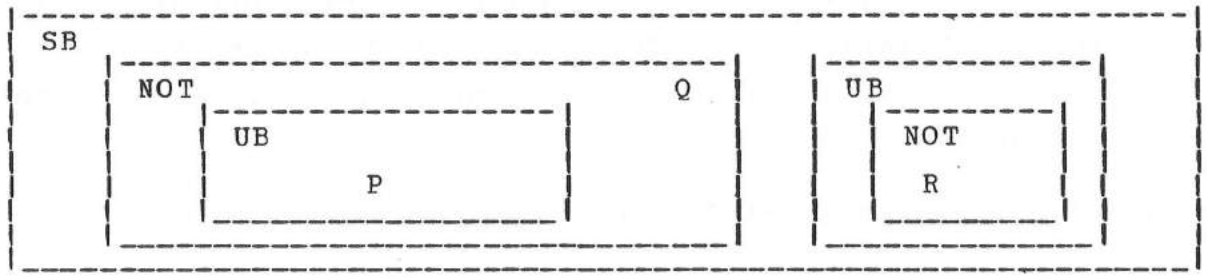

FIGURE 4.4a
REPRESENTATION OF "SYSTEM BELIEVES THAT IT'S NOT THE CASE THAT THE USER BELIEVES P" AND "THE USER BELIEVES R IS FALSE."

4.4.2 Disjunction of Beliefs

Hendrix's representation for disjunction gives us a convenient way to represent such statements as "John believes P or John believes Q". The representation for this statement is that of Figure 4.4b. It should be noticed that this disjunction is not contained in SBJB. Contrary to Moore's claims, there is no exponential growth in the number of data bases necessary to represent John's beliefs since there is no need for each belief space in a disjunction to be a complete specification of everything John believes. The representation proposed here simply maps directly onto a set of propositions.

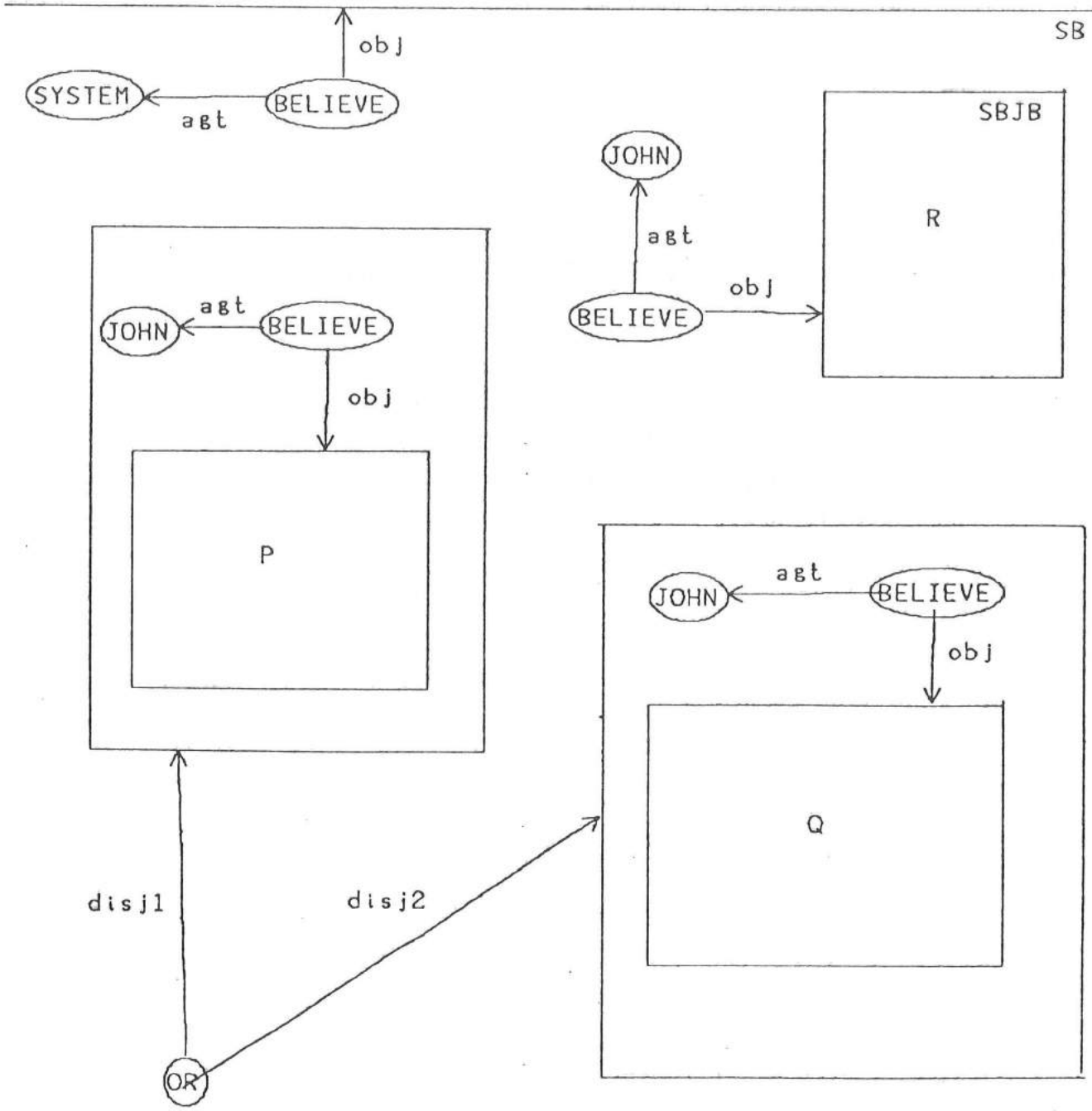

FIGURE 4.4b

"JOHN BELIEVES P OR JOHN BELIEVES Q"

Notice also that this is not the same as saying "John believes P or Q", which is represented in 4.4c.

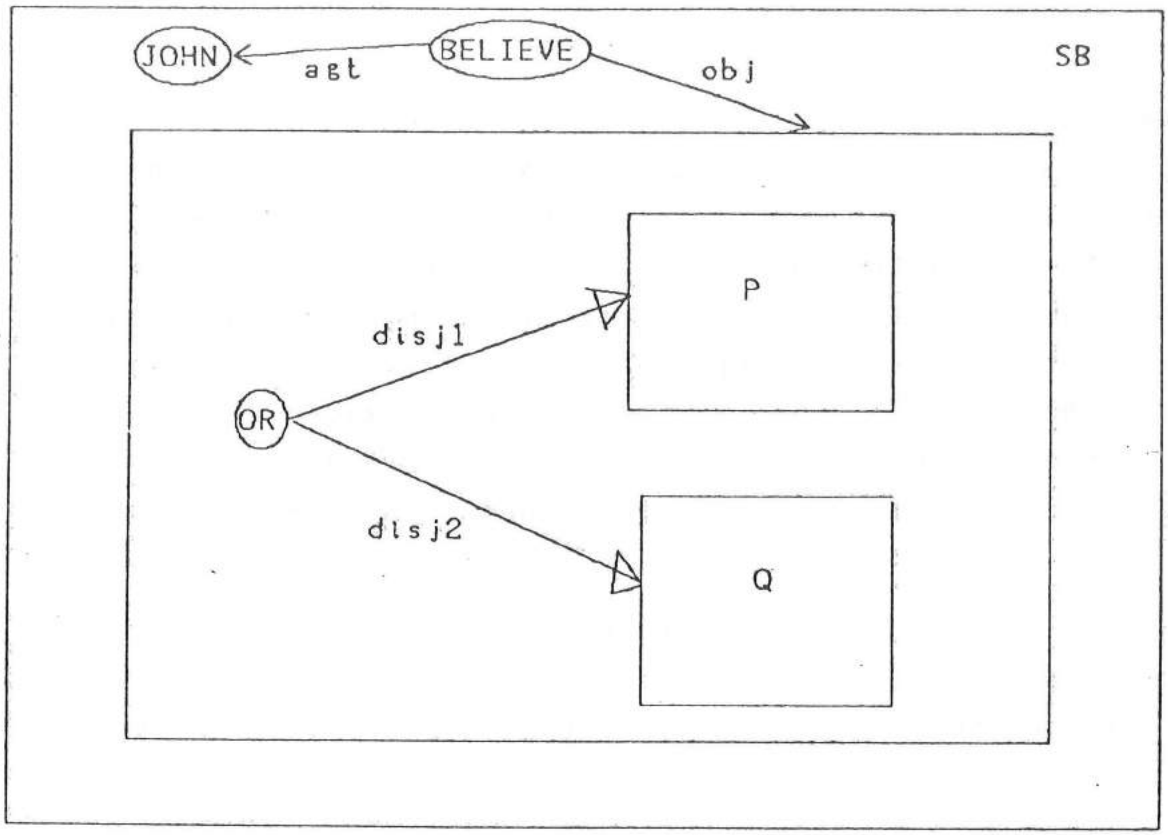

FIGURE 4.4c

"JOHN BELIEVES P OR Q"

4.5 Digression -- Belief Spaces, Existence, and Agnosticism

Those readers uninterested in a discussion of how the representation handles the problems of existence in reality should ignore this section. Nothing new will be presented here. Rather, I will be concerned with justifying why I have chosen the particular representational primitives that I have.

The major ontological question to be answered is "do the objects in a belief space denote objects in the real world?" Representational adequacy dictates that we represent mental objects that are intended to stand for real world (but possibly non-physical) objects, mental objects that are believed not to have counterparts in reality, and mental objects about which the system is not commited to there being corresponding objects in the real world. I will propose a representation whose consequences require further study.

Mental objects in a belief space are to be interpreted as representing real world entities (even abstract ones). A belief space thus means "belief that there exists in the real world." This does not violate the solipsistic principle since the system cannot mentally manipulate real world objects. With this interpretation, objects standing for concepts not believed to exist in reality, e.g., Santa Claus, would be in a NOT space in SB. In addition, OSCAR's memory could include spaces for various hypothetical worlds (e.g. "Christmas myths") within which the object representing Santa Claus could be located. The system could be uncommitted to Santa's existence by not placing Santa in either SB or its NOT space, but leaving it solely in the hypothetical world. The presence of such a space in SB would mean the system believes such a myth exists, while the contents of that space would be what is believed true (and what exists) in the myth, not in reality. For instance, the proposition representing the statement "Santa Claus lives at

the North Pole" would need to be in such a myth space since it is a true statement about Santa. We would not want to place such statements in a NOT space since if we did, we would not be able to handle differences between "Santa lives at the North Pole" and "Santa lives at the South Pole".

Given these assumptions, we would have means for representing objects that are believed to exist, or not, in reality. Abstract sets and instantaneous events are then believed to "exist" in the real world. I am not sure if such ontological commitments would be problematic. However, I have chosen this approach to belief and existence over its competitor (explained in the footnote) since common situations (beliefs about existence) are represented simply. I do believe, though, that logical systems would have difficulty being agnostic about the existence of concepts.

A future goal is to show how one can generate (or plan) referring expressions without employing reality as a source of denotations through which one must go in order to refer. Hence, while belief and existence would be tied, the reference mechanism might not care. The reference machinery might then treat Richard Nixon in the same fashion as Scrooge or Winnie the Pooh, independently of their supposed existence. If such an approach were successful, it would convince me of the value of OSCAR's ontology.

4.6 Notes

1. People may in fact have goals of which they are unaware. However, once they use them in a deliberate planning process, one might then claim that they have become aware of them. The above only offers a representation of "conscious" (in some very loose sense) goals, about which OSCAR could conceivably converse.

2. There is another approach that makes no commitment to the existence of the system's mental objects. Mental objects that do denote real world entities would be specially flagged by an EXISTS-IN-REALITY proposition (or, alternatively, by being in a special space as Hendrix employs and later abbreviates). A belief space would then contain all the system's mental objects; the propositions in SB would be acceded to by the system as

"believed true". Such concepts as "God", where one might want the system to remain uncommitted, would then reside in SB. Objects like "Santa Claus", whose counterparts are not believed to exist, would be in the belief space SB, the EXISTS-IN-REALITY proposition would be in the NOT space in SB, and the proposition stating that Santa inhabits the North Pole would be in some "Christmas myth" space. The interpretation of propositions in NOT spaces, of course, would be relative to the interpretation of the space containing the NOT space.

The status of the existential quantifier would then become problematic. Would we want to quantify over mental objects, or possible mental objects? Would we need another quantifier to distinguish existence in reality from mental existence? Since I cannot answer these questions, I have not adopted this approach despite intuitive feelings that it may be superior in the end.

CHAPTER 5

PLANNING SPEECH ACTS

Contents:

5.1 Introduction

The thrust of the argument so far is that if one treats language from the point of view of action, one can begin to arrive at an understanding of how we know what to say. The demonstration of this point is the purpose of the next two chapters. The representation mechanisms will be seen to create speech act definitions in the same form as those of other actions with the exception that speech acts are intended to explicitly manipulate the beliefs and goals of speakers and hearers.

Within this context, I will show that even a simplistic planning algorithm suffices to produce interesting speech act behavior. In particular, OSCAR plans requests when trying to influence the user's goals, and plans inform speech acts when trying to influence his beliefs. Not only does OSCAR discover what kinds of speech acts to issue, but it also determines their propositional content in the process.

Prototypical plans are displayed that serve to show the belief and goal configurations that OSCAR develops in planning a REQUEST or INFORM. To show more variety, a new domain of discourse, an extended version of one proposed by Power [1974], is formally defined. Plans incorporating speech acts are then developed from different initial belief configurations. For instance, these often involve planning more than one speech act (say a REQUEST and an INFORM) where the INFORM is planned to establish beliefs of the user that then enable the REQUEST to be planned. Decisions made during the normal course of planning will be shown to influence the propositional content of speech acts. All of the above will be shown to depend, in a natural way, upon OSCAR's model of its user. However, before proceeding to the planning of speech acts, we must first discuss the structure of operators and plans that OSCAR employs.

5.2 The Structure of Events

Operators, or events, are the system's means for changing its model of the world. They do not, however, manipulate or influence the physical world. Events are used to model such physical operations in the domain of discourse as moving, pushing, and painting, and lead to changes of properties of entities, such as location or color. Each includes an agent -- a person who is intentionally performing the corresponding action. This presumed intentionality will play a significant role in the planning of speech acts.

In order to see the structure of operators employed throughout this thesis, consider the definition of the class EVENT in Figure 5.2a.

```
event is a new instance of relation
CASEFRAME:     (agt := a person) event
TO.TEST: std
TO.ADD:  std
TO.GENERATE:  std
ANY OTHER DEFINING PROPERTIES?
cando.pr
PLEASE DEFINE CANDO.PR: null
ANY OTHERS?
want.pr
PLEASE DEFINE WANT.PR:  agt believe (agt want instance)
ANY OTHERS?
effects
PLEASE DEFINE EFFECTS:  null
ANY OTHERS?
body
PLEASE DEFINE BODY:     add (retrieve of (effects of self))
ANY OTHERS? no.
```

FIGURE 5.2a

INTERACTION WITH MDP TO DEFINE THE CLASS EVENT

EVENT is defined to be an instance of the meta-class RELATION. The caseframe defines one case, AGT, that states that all events are performed by some person. The testing program is the usual one for RELATIONs, namely PROPTEST, that simply examines the world space for exact matches of the conjecture. For an n-ary relation, conjectures are simply instances of the relation whose presence in a particular space is in question. The standard asserting program, PROPADD, which simply moves the conjecture into the world space, is also used. The printing of an INSTANCE of

EVENT is controlled by PROPGENERATE, the standard printing program for relations.

Operators change the contents of the world space via their effects, which are written as proposition schemas. Evaluating the body of the operator asserts an instance of the effect schema into the current belief space; the evaluation simulates the doing of a real world action. Unlike Levesque's [1977] definition of programs, the EFFECT of an operator is a statement of the propositions that are to be true after the operator is completed. The statement of these propositions, then, is divorced from the means for making them true. Such additions may necessitate making appropriate deletions to the world (such as the initial location of the agent). However, the deletions need not be mentioned explicitly in the definition since they have been made the province of the "to add" routines for the relations in question.

The next two defining properties of EVENT are the "cando" and "want" preconditions. The ramifications of splitting up preconditions into two groups will be discussed in section 5.7.5.2. An event is applicable in any configuration of the world space where its preconditions are true. In general, the "cando" preconditions (or CANDO.PRs) state those applicability conditions that need to be true in the belief model. All preconditions in this system are also stated as proposition schemas. Testing to see if a particular precondition is true involves evaluating the "to test" procedures of the appropriate relations. Notice that the testing of the preconditions is not represented explicitly but is invoked by a process external to the act -- the planning algorithm or the interpreter.

The second type of precondition, the "want" precondition (or simply the WANT.PR) is used in the modelling of intentional behavior. It states that someone will do some action only if they want to do it. An agent may believe

that all the CANDO.PRs of an action are true, but he will not do the action unless he wants to. The WANT.PR in Figure 5.2b (AGT BELIEVE AGT WANT INSTANCE) is applicable to any EVENT and hence is inheritable via the STD mechanism of Chapter 3. We shall see later how this WANT.PR becomes crucial to the planning of speech acts.

The relation MOVE is defined as a subclass of EVENT and thus receives EVENT's defining properties (see Figure 5.2b). MOVE takes three arguments: a human agent, an initial location for that agent, and his final location. This MOVE relation states that before someone can move from one place tc another, he must first be at that original place. The WANT.PR states that the agent must want to move before doing so. The EFFECT clearly establishes a new location for the AGT, while the BODY of the action is simply the adding of the effects into the world space.

```
define move as event
CASEFRAME:  (agt := a person) move from (source := a place)
              to (dest := a place)
CANDO.PR:     loc of agt is source
WANT.PR:      std      [ agt believe (agt want instance) ]
EFFECTS:      loc of agt is dest
BODY:    std
TO.TEST: std
TO.ADD:  std
TO.GENERATE:  std
```

FIGURE 5.2b
DEFINITION OF THE EVENT MOVE IN MDP.

5.3 Plans

People do not simply execute actions at random. They have reasons for actions; they intend for them to achieve certain effects, for instance, establishing conditions for

future actions. In trying to model rational behavior, we must model how such "connected" causal sequences of actions, plans, are created. The objective of a planning algorithm is to discover a sequence of physical actions that will change the real world to make a desired condition true. Internally, the algorithm models actions by operators, as previously discussed. However, OSCAR cannot affect the real world and hence it maintains no "action routines", only operators that change its beliefs. (From here on, the terms "action", "event", and "operator", in technical contexts, should be regarded as synonymous.)

The planning algorithm attempts to compose operators so that at any step in a plan, the current operator is applicable in the current state of the world model. There are two obvious strategies for creating such a plan achieving a given goal. A pure backward-chaining planner would find operators to achieve specific goals, and develop subgoals from the preconditions of those operators that were not satisfied in the world model. A pure "forward-chaining" planner would: start with the current model of the world, discover an applicable operator that could transform the world into a new one where the operator's effects held, and repeat until the original goal was achieved. Current planning algorithms (e.g., Fikes and Nilsson's STRIPS [1971], and its successors) use both of these strategies, allowing them to insert operators anywhere in a plan, rather than being forced to add operators only at one end, as the pure strategies would. In order to work "middle-out", they associate to each (sub) goal the model of the world in which the goal is to be achieved. Thus, all the information needed by an operator would be found with the goal. Of course, the problem is to guide this process in the direction of the ultimate objective.

Perhaps the best existing planning system is Sacerdoti's [1975] NOAH program. I discuss it here since its methodology may provide a model for extending OSCAR.

5.3.2 The NOAH Planning System

This problem-solver simultaneously addressed two issues: planning by refinement, and non-linear plan development. NOAH tries to avoid premature commitments to the temporal ordering of operators in a plan. In contrast, linear planners, such as those characterized above, when achieving a conjunctive subgoal (say, A AND B), would attempt to achieve one conjunct and then the other. The plan to achieve A, however, might interfere with the plan to achieve B. For instance, if A is the goal "paint the ceiling" and B is "paint the ladder" (as in [Sacerdoti 1975]), the algorithm should discover that painting the ladder first would interfere with the painting of the ceiling. To do this, NOAH develops plans to achieve A and B separately. It then criticizes the whole plan, discovering conflicts between the operators in parallel branches, and attempts to reorder the offending ones.

Operators in NOAH are described at multiple levels of detail. Planning is first done at one level. The resulting plan is criticized and then is expanded to a lower-level of detail by replacing each operator with the set of operators that defines it. Planning continues until all operators have been expanded as far as possible, and all leaf subgoals are achievable. This methodology allows NOAH to avoid consideration of unimportant details in the early stages of plan development.

5.4 OSCAR's Planning Algorithm

OSCAR employs a pure backward-chaining planning algorithm that, compared to NOAH, is quite primitive. However at our current state of investigation it is

sufficient to produce interesting speech act behavior. Its limitations will be discussed after the algorithm has been described.

The action PLAN, a primitive, is invoked by an MDP statement whose form has been defined to be AGT PLAN TO ACHIEVE GOAL, where AGT and GOAL are variables ranging over the classes PERSON and PROPOSITION, respectively.

Essentially, the algorithm works as follows:

Given a goal GOAL, a world space W, and a want space WS,

Until there are no new goals, or ordered to give up:

1. Find the generic action(s) having as effect a proposition of the same type as the GOAL. This will result in an ordered list of acts that could possibly produce the GOAL.

2. MATCH GOAL against the effects of the first action on the above list to obtain a set of bindings to the arguments of the action. If the match fails, get the next producer on the list and repeat step 2. If the list becomes exhausted, planning fails.

3. Create an instance, in WS, of the producing action (Call the new instance PROD.) This creation process replaces instances of variables in the producer by any bindings they might have in the list of bindings obtained from step 2. An EFFECT proposition should be created in WS to relate the GOAL to PROD.

4. Test the preconditions of the producer, in world W, using bindings where possible. Link PROD to the supporting proposition(s) of any preconditions already believed to be TRUE. Create and link as new goals, any preconditions that are FALSE or UNK.[2] These new goals should be placed on the agenda of unsolved subgoals.

5. Decide upon the next GOAL. (The user makes this decision.)

5.4.1 Applying the Algorithm to an Example

Perhaps the best way to explain the properties, decision points, and limitations of the algorithm is to examine each of the steps in more detail when applied to an example. The simplest possible example is one in which there is one action, say MOVE, and one state affected by it, LOC. The system will plan to achieve LOC OF SYSTEM IS INROOM, where INROOM is a PLACE. OSCAR is doing the planning so the world space is SB and the want space is SBSW. Since this algorithm is parameterized by the world and want spaces, OSCAR is also able to simulate its user's planning by performing the algorithm in the user's spaces. This will be a useful maneuver, as section 5.6 will show.

Step1 essentially produces an ordered list of actions that are known to have effects of the appropriate type. Discovering the potential producing actions of a state is a simple matter since MDP inserts special links for states created in an EFFECTS schema in the definition of any action. Thus, if the goal is a LOC proposition, OSCAR would discover that MOVE is one appropriate action.

On the other hand, deciding upon the best ways of achieving a goal, i.e. the ordering of the list, is a much more difficult problem. The way that most current planning systems (cf. [Sacerdoti 1975], [Hayes 1975], etc.) deal with this problem is to associate a procedure to each state (e.g., LOC) that decides how goals of that form should be achieved. Such procedures often incorporate domain-dependent heuristics. I have adopted this methodology and use such procedures, termed "dfunctions" after Hayes' versions, whenever a state can be produced by more than one action.

In Step 2, the EFFECT of MOVE matches the goal and establishes bindings for the AGT and DEST parameters of MOVE. Step 3 creates a new instance of MOVE in the want space and incorporates it into the plan. However, the matching of effects to goal in Step 2 has only determined some of the bindings of some of the arguments of the action. In particular, SOURCE still has not been bound. When creating an instance of MOVE in the plan a new variable would be created for the SOURCE position. When the testing of preconditions does not produce bindings for such new variables, OSCAR will eventually ask its "oracle" (the user) for bindings of some of the remaining variables in the plan (cf. the example of Chapter 1).

Finally, the WANT.PR of MOVE must be tested. Planning takes place in the want space of the AGT of the PIAN action (the PLANNER). Hence, all propositions and objects created in the plan are automatically wanted by the PLANNER. The precondition of the MOVE relation states that AGT BELIEVE AGT WANT INSTANCE. AGT is bound to SYSTEM and INSTANCE has become bound to the new instance of MOVE (call it MOVE1) that was created by STEP 3, and is already part of the plan. Therefore, the WANT.PR of MOVE1 is that the SYSTEM has to want MOVE1! But, of course, this condition is trivially true since MOVE1 is already in the system's want space. Therefore, no new subgoals need to be added to the plan, which is now complete and is pictured in Figure 5.4a.

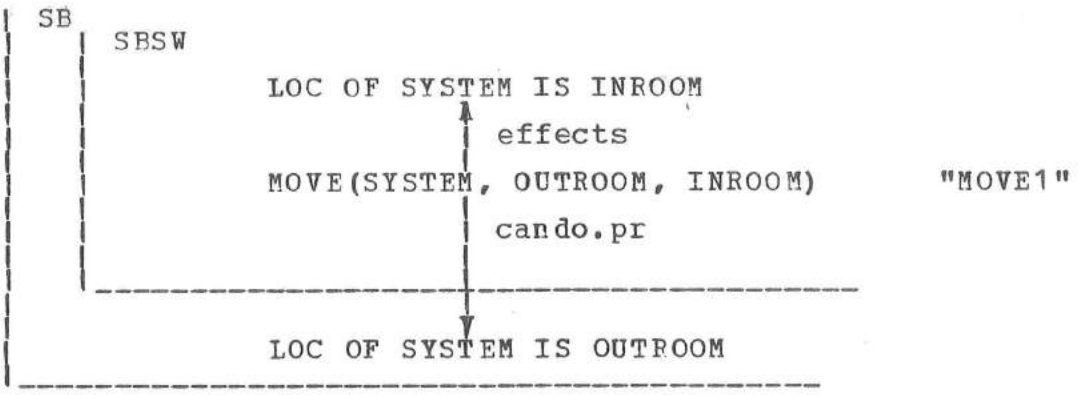

FIGURE 5.4a
A TRIVIAL PLAN TO MOVE

5.4.2 Limitations of the Algorithm

A comparison of this algorithm with NOAH and other problem-solving algorithms clearly points up the simplistic nature of the above method. Some of the simplifications inherent in the algorithm are:

- Bindings:

  When OSCAR cannot determine any bindings for a variable in an action that has just been added to a plan, it "asks" the user. Currently, this interaction is simply a "canned" way of alleviating the difficulty. Chapter 6 will mention how we can get the system to plan a question when it is lacking such binding information.

  When there is more than one possible binding for a variable, the first one OSCAR encounters is chosen. NOAH, however, attempts to delay such binding choices until there is an obvious best choice.

- Backtracking:

  There is no backtracking in the algorithm. Once an action has been chosen and its EFFECT has been successfully matched to the goal, no other operators are tried. This could severely limit performance, and correctness, in complex task domains but no major

performance improvements in our simple domains are forseen, given the goal of demonstrating the capacity for producing speech acts.

- Multiple models of the world:

  Since OSCAR is strictly a backward-chaining planner, there was no need to attach different models of the world to different goals in the plan. As a later example shows, this simplification causes OSCAR to produce a number of superflous speech acts.

- What to do next:

  When the preconditions to an operator in the plan require that more than one subgoal be achieved, we need to decide which to work on next. This "focus of attention" strategy is a major factor in controlling the combinatorics of problem-solving processes. Much attention has been devoted to it by researchers in speech understanding (cf. [Walker et. al. 1976]). However, again our objectives seem to have been better served by simply having OSCAR let the user make such decisions, thereby giving him direct access to the critical decision making aspects of planning.

There was a deliberate attempt not to incorporate many of the sophisticated techniques of current problem-solvers. Instead, I have tried to pursue the application of planning to language by implementing a "lowest common denominator" of the lot. I expect that more sophisticated planning algorithms (especially NOAH-like ones) will soon be necessary to progress beyond the stage reported here. With that in mind, the representation of knowledge was designed to have a flavor similar to what one would need if one were to use NOAH. Since the definitions of actions can be stated in terms of a sequence of constituent actions, NOAH-like expansions of levels of detail can be obtained.

5.5 Planning a Request

The plan of the previous example involved only one agent and thus was artificially simple. When one attempts to influence the behavior of others, communication usually occurs. This section will show how OSCAR can attempt to get the user to perform some action by planning speech acts of the directive class. Speech acts in this class, other than REQUEST (e.g., SUGGEST and COMMAND), would have more specific preconditions but would be planned in the same fashion. While employing the same domain of discourse as the previous example, this time the goal will be LOC OF USER IS OUTROOM.

5.5.1 Definition of REQUEST

In order to plan requests using the current algorithm, we need to define an action REQUEST in the form specified in section 5.2.1. The flavor of Searle's definition of a request (see section 2.2) can be captured by the following (annotated) action definition in MDP:[3]

```
define request as event
CASEFRAME:    (agt := a person) request that (recip := a person)
                   do (obj := an event)
CANDO.PR:     agt believe
                   recip cando obj
              [I can't reasonably request you to do something
              that I don't believe you can do.]
                   and
         agt believe
              recip believe
                   recip cando obj
              [If I don't believe you believe you can
              do the action being requested of you, I
              cannot perform solely a request but may
              also have to provide you with information
              that I believe will enable you to perform
              the action.]
WANT.PR: std
EFFECTS: recip believe
              agt believe
                   agt want obj
              [This indicates that the speaker's goal
              is to communicate what the speaker wants.
              This allows the hearer to refuse to
              perform action OBJ without invalidating
              the speaker's utterance as a request.]
BODY:    generate instance
         std
TO.TEST: std
TO.ADD:  std
TO.GENERATE:  std
```

FIGURE 5.5a

THE DEFINITION OF REQUEST IN MDP

As usual, testing, adding, and printing routines are inherited from RELATION through EVENT. In particular, OSCAR's printing of a REQUEST will be of the form of a performative -- "I request that..." The BODY of REQUEST prints the REQUEST (because this is a speech act) and then asserts the EFFECTS into the world space. The propositional

content of a REQUEST can be any instance of the class EVENT. In particular, it can be another speech act.

As with other events, the definition of REQUEST is an operator schema that describes the form of its instances. In the usual way, the preconditions and effects are proposition schemas. The term "speech act" will often by used to refer to the definition of the act (the operator schema) in addition to its use in referring to instances of that schema. The context of the usage of the term should make matters clear.

Most of Searle's formulation has been incorporated into this definition. Neither his "non-obviousness" condition nor his "sincerity" condition are present since, as will be shown, they are incorporated into the planning process.

5.5.2 CAUSE_TO_WANT

The speaker's goal in issuing a request is to get the hearer to know what the speaker wants him to do. The effect of a REQUEST indicates that a separate step is necessary for the hearer to decide to do that action. In a previous section, I argued that the planning process requires some controlling strategy that determines what the system ought to do next. One might argue that this ought to be the overall controller of the entire system. In other words, OSCAR would be in a "decide what to do next and do it" loop, where the "doing" may create new goals. While OSCAR could assume such a strategy of its users (and everyone else), its version of this mechanism could be much more complicated than its model of the corresponding strategy for others. It is sufficiently complex that I have decided to trivialize it, as mentioned previously, by incorporating an oracle that decides which of OSCAR's goals deserves immediate attention.

From this perspective, the point of a REQUEST is to influence the hearer's "what's next" process. Before the hearer can decide to perform some action, he must want to do

it. Hence, we need a way of getting a person to want someone else's goal as his own -- a way of influencing someone's "to test" procedure for WANT. The act CAUSE_TO_WANT is intended to model what is necessary for someone to affect someone else's procedure:

```
define cause_to_want as event
CASEFRAME:     (agt1 := a person) cause
               (agt := a person) to want to do (obj := an event)
CANDO.PR:      agt believe
                    agt1 believe
                         agt1 want obj
WANT.PR:  none
EFFECTS:       agt believe
                    agt want obj
BODY:     std
TO.TEST:  std
TO.ADD:   std
TO.GENERATE:  std
```

FIGURE 5.5b
THE DEFINITION OF CAUSE_TO_WANT *

Thus, one attempts to get a person to want some goal by getting them to know that you want it. Since we are assuming the above decision loop, a WANT.PR is not necessary -- everyone is assumed to be continually deciding what they want to do next. This simplistic CANDO.PR might be extended to require that AGT believes that a friendship or authority relationship holds between the two participants. Such an extension is easily stated in MDP and presents no difficulties for OSCAR, as we shall see. Other extensions are possible (for instance, see Schank [1975]).

5.5.3 The Planning of a Simple REQUEST

REQUEST supplies the necessary precondition for CAUSE_TO_WANT (as will other act combinations) and thus, when the WANT.PR of some planned action to be performed by someone else is not believed to be true, OSCAR plans a REQUEST. The following are the user's interactions in getting OSCAR to plan to achieve LOC OF USER IS OUTROOM.

```
add (loc of user is inroom)
OK
add (user believe (loc of user is inroom))
OK
system plan to achieve (loc of user is outroom)
DECIDE ON AGENT OF CAUSE_TO_WANT PLEASE
system
OK
generate (planof system to achieve (loc of user is outroom))

SYSTEM REQUEST THAT USER DO
    USER MOVE FROM INROOM TO OUTROOM

ACHIEVING:

USER BELIEVE
    SYSTEM BELIEVE
        SYSTEM WANT
            USER MOVE FROM INROOM TO OUTROOM

THEN:

SYSTEM CAUSE USER TO WANT TO DO
    USER MOVE FROM INROOM TO OUTROOM

ACHIEVING:

USER BELIEVE
    USER WANT
        USER MOVE FROM INROOM TO OUTROOM                    ++

THEN:

USER MOVE FROM INROOM TO OUTROOM                            +

ACHIEVING:

LOC OF USER IS OUTROOM
```

FIGURE 5.5c

OSCAR'S PLAN TO GET THE USER TO MOVE

What is printed here, then, is a temporal ordering of actions followed by a statement of their effects. The final steps of the development of this plan require that OSCAR achieve the preconditions of the REQUEST that was just created. The WANT.PR of the REQUEST (i.e., that SYSTEM BELIEVE SYSTEM WANT SYSTEM REQUEST...) is trivially true

since the agent of the REQUEST is the planner, and hence the REQUEST is in space SBSW. What remains are the CANDO.PR's of REQUEST (see Figure 5.5a) involving the relation CANDO.

5.5.4 CANDO

The CANDO.PR's of REQUEST refer to the applicability conditions on the hearer's doing the action being requested. The relation CANDO, intended to model someone's being able to do some action, is defined as a primitive with the caseframe:

(agt := a person) CANDO (obj := an event)

It thus takes two arguments: the person whose ability is in question and the action to be performed. In general, a person AGT CANDO some action OBJ if: the CANDO.PR's of OBJ are true with the agent of OBJ bound to AGT, or if AGT can plan to make those preconditions true. One way of determining if AGT can make the CANDO.PR'S of OBJ true is to simulate AGT's planning within the appropriate belief and want spaces for AGT. If we assert that a person AGT CANDO some action OBJ, then we are in fact asserting the CANDO.PR's of OBJ with AGT bound to the agent slot.

The algorithms for testing and adding CANDO are:

Testing AGT CANDO OBJ:

If the evaluation of the TEST of the CANDO.PR's of OBJ yield a SUCCESS proposition

Then return that proposition.

Else, If the AGT = PLANNER (a variable indicating who is doing the planning, if planning is in fact taking place),

Then return the outcome proposition from the above testing. This is necessary to rule out a superflous recursive call to the PLAN routine. The

PLAN algorithm will eventually work on those CANDO.PR's if it is not already doing so.

Else find AGT's wantspace in the current world space. Invoke the PLAN algorithm on the CONJECTURE of the above outcome proposition. This simulates the AGT's planning provided the PLANNER variable is set to be the AGT.

When the planning algorithm returns, it either returns a plan for AGT, achieving the above conjecture, or it returns the goal(s) upon which it was working when it either failed or was forced to give up. CANDO then forms an appropriate outcome proposition and passes the result back.

Adding AGT CANDO OBJ:

Evaluate the "to add" of the CANDO.PR's of OBJ with AGT bound to the agent role of OBJ. This will add the CANDO.PR's into the world space.

This algorithm does not quite guarantee that the agent can in fact execute the action. In addition to checking preconditions, to guarantee that execution can start, one must check that there exist values for the arguments of the action. OSCAR can criticize the plan at each step and can create new goals to supply values to slots of planned actions. However, a discussion of representing that the user knows the values of the variables must await our discussion of questions in Chapter 6.

All that remains to complete the plan of Figure 5.5c is to check the CANDO.PR's of REQUEST. These preconditions refer, via CANDO, to the CANDO.PR's of the action being requested -- MOVE. OSCAR then determines if it believes LOC OF USER IS INROOM and if USER BELIEVE (LOC OF USER IS INROOM), both of which it does since they were initially asserted. Planning is thus successfully terminated.

5.5.6 Searle's Conditions for a REQUEST

An argument I have given for studying the processing that underlies language is that one may find that we can generalize the descriptions of various phenomena by characterizing the processes that produce them. I claim this may be the case in describing speech acts. Searle [1969] attempted to set forth necessary and sufficent conditions for making successful requests (among other speech acts). Those conditions included a "non-obviousness" condition that, in the case of a request, stated that it should not be obvious to the speaker that the hearer is about to do, independently of the request, the action being requested. If that were obvious to the speaker, the request would be pointless. A second condition in his list, the "sincerity" condition, stated that the speaker had to want the requested act to be performed.

However, the non-obviousness condition applies more generally to rational, intentional behavior rather than just to speech acts. The non-obviousness condition is the WANT.PR of the act being requested (goal "++" in Figure 5.5c). If the system believed the WANT.PR were already true, then the plan would proceed no further; no REQUEST would take place. However, OSCAR might make a REQUEST for the purpose of communicating that it wanted the user to perform the act even though it believed the user already wanted to do it. But in that case, OSCAR would not have the usual goal from which REQUESTS are planned but would only want that the user believe that the system wants him to do it. Some process other than CAUSE_TO_WANT would have to state such a goal, for purposes which we can only surmise.

The sincerity condition in the example of Figure 5.5c is the goal labelled "+". The system's wanting the user to move must have been a goal in order for this plan to have taken place at all! Searle's sincerity condition is not part of the definition of REQUEST but is the reason for

planning REQUEST speech acts. The conditions are properties derived from the algorithm rather than from the objects upon which the algorithm is working. By discovering such conditions, we derive the advantage of not having to repeat them for each action definition.

5.5.6 Summary of the Planning of REQUESTS

It should now be clear that OSCAR can plan a REQUEST whenever its goal is that someone do some action. That goal would lead, via the WANT.PR of the action, to getting that person to believe that the system wants him to do it. Not only is OSCAR deciding to issue a REQUEST, but it is also determining the propositional content of the speech act via the planning process. Before requesting that a person do some action, OSCAR ensures that he can (and believes he can) perform it, i.e., that the appropriate preconditions of that action are true. As will be seen in section 5.6, new speech acts may need to be planned in order to get him to believe they are.

5.6 Planning INFORM

The other class of speech acts that is of immediate interest is the class represented by INFORM. This section will describe how OSCAR generates INFORM speech acts on the basis of believing that the user needs to know some proposition. Again from a non-linguistic goal, OSCAR will plan a speech act and also determine its associated propositional content. I will give simple examples of OSCAR's planning and defer more extensive discussion to section 5.7 and Chapter 6.

5.6.1 Definition of Inform

An INFORM speech act consists of a speaker's stating a proposition to some hearer for the purpose of getting the hearer to be aware that the speaker believes that proposition to be true. The formal MDP version of this definition is:

```
define inform as event
CASEFRAME:    (agt := a person) inform (recip := a person)
              that (obj := a proposition)
CANDO.PR:     agt believe obj
WANT.PR:      std
EFFECTS:      recip believe
                   agt believe obj
BODY:    generate obj
         std
TO.TEST: std
TO.ADD:  std
TO.GENERATE:  std
```

FIGURE 5.6a

DEFINITION OF THE SPEECH ACT INFORM IN MDP.

The CANDO.PR simply states that the only prior condition to informing someone that proposition P is true is that the speaker believes P. This condition will be rewritten when we consider the planning of questions. (A CANDO.PR of the speech act LIE is that the speaker believes NOT(P), while that of DENY is that the speaker believes the hearer believes NOT(P).)

The effect of an INFORM is to communicate the speaker's belief. This allows for the hearer to refuse to believe the proposition without invalidating the speaker's action as an INFORM. Therefore an intermediate step, termed CONVINCE, is necessary to get the hearer to believe the proposition.

5.6.2 CONVINCE

The act that produces someone's believing some proposition, CONVINCE, will represent what it takes to affect a person's testing procedure for BELIEVE. The simplest possible way of defining this is to say that all that is necessary to get someone to believe some proposition is to get him to believe that someone else believes it. Since I feel that people are always evaluating the truth of their perceptions, there is no need to try to get them to want to do such an evaluation. Hence there is no need for a WANT.PR on CONVINCE.

The MDP definition of CONVINCE is:

```
define convince as event
CASEFRAME:     (agt := a person) convince
               (recip := a person) that (obj := a proposition)
CANDO.PR:      recip believe
                    agt believe obj
EFFECTS:       recip believe obj
WANT.PR: none
BODY:    std
TO.TEST: std
TO.ADD:  std
TO.GENERATE:  std
```

FIGURE 5.6b

DEFINITION OF CONVINCE

Notice that while OSCAR models what is necessary to get someone else to believe something, and may even assume everyone employs a similar model of everyone else (including OSCAR), there is no requirement that OSCAR actually use such a simple-minded testing procedure for BELIEVE. As with CAUSE_TO_WANT, the modelling that enables one to plan to influence such a procedure need not be as detailed as the procedure itself.

For a more sophisticated precondition of CONVINCE, one might state that not only does the hearer need to know the speaker's beliefs but if he also knew the speaker's reasons for holding such beliefs, he would be more inclined to believe the speaker. In other words, to CONVINCE someone of proposition P, one may have to CONVINCE them of the reasons for believing P, which requires that they know (or be CONVINCEd of) the reasons for believing those reasons, etc. Such a chain of reasons for believing might be terminated by common knowledge that all people in certain categories believe, (and believe everyone else does too) or by a belief the speaker believes the hearer already has. While OSCAR employs a version of CONVINCE that uses the simple CANDO.PR, the above seems to be an interesting extension.

5.6.3 Planning INFORM Speech Acts

The planning of INFORM speech acts now becomes a simple matter. For any proposition P, the plan to achieve the goal USER BELIEVE P would be that of Figure 5.6c. Notice that we do not need to state, as a precondition to INFORM, that it is not the case that the hearer already believes P. This is another instance of a "non-obviousness" condition that is obviated by viewing speech acts in a planning context.

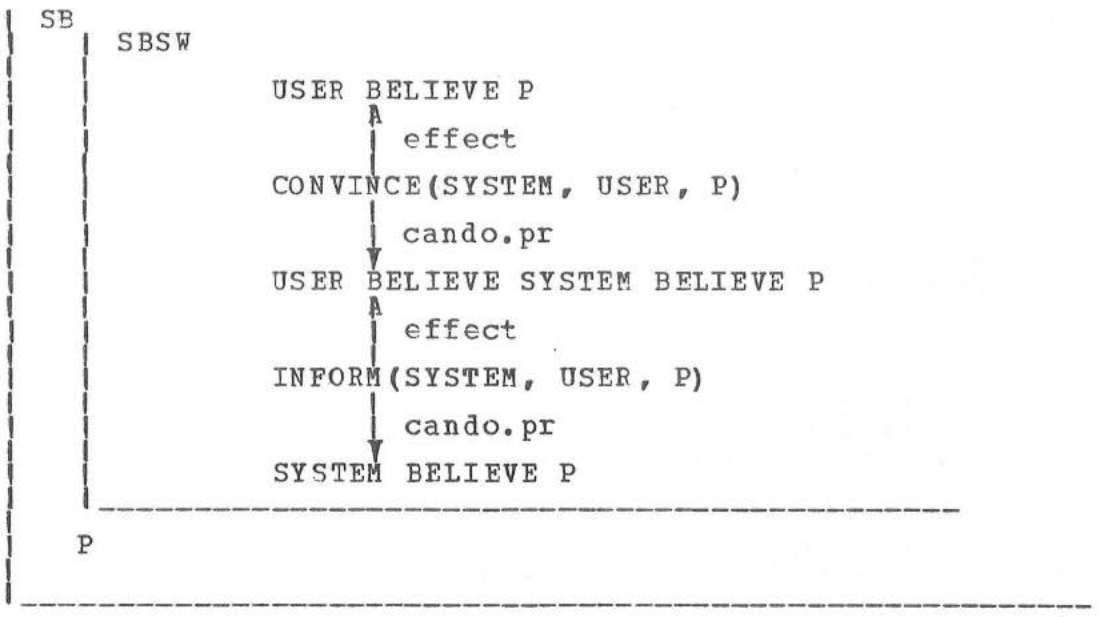

FIGURE 5.6c

PROTOTYPICAL PLAN OF AN INFORM.

What would be Searle's sincerity condition for the INFORM above (SYSTEM BELIEVE P) turns out to be a precondition for the speech act rather than a perlocutionary effect of the act, as we had for a REQUEST. One may then question whether Searle's sincerity condition is a consistent naming of distinctive features of various kinds of speech acts. If we were to use REQUEST as a model, the sincerity condition for an INFORM would be SYSTEM BELIEVE SYSTEM WANT USER BELIEVE P. In other words, for a speaker to sincerely inform someone of P, he must want that person to believe P. It is not clear, however, whether these naming problems really make any substantive differences in the behavior of a system that would use speech acts.

To continue with our earlier example, another way to achieve the goal of LOC OF USER IS OUTROOM is as follows:

```
(assume the same initial configuration of beliefs
as in Figure 5.5c)

system plan to achieve (loc of user is outroom)
DECIDE ON AGT OF CAUSE_TO_WANT PLEASE
system
OK
generate planof system to achieve (loc of user is outroom)

SYSTEM INFORM USER THAT
    SYSTEM WANT
        USER MOVE FROM INROOM TO OUTROOM

ACHIEVING:

USER BELIEVE
    SYSTEM BELIEVE
        SYSTEM WANT
            USER MOVE FROM INROOM TO OUTROOM

THEN:

SYSTEM CAUSE USER TO WANT TO DO
    USER MOVE FROM INROOM TO OUTROOM

ACHIEVING:

USER WANT
    USER MOVE FROM INROOM TO OUTROOM

THEN:

USER MOVE FROM INROOM TO OUTROOM

ACHIEVING:

LOC OF USER IS OUTROOM
```

FIGURE 5.6d

PLANNING AN INFORM OF A WANT

The initial stages of this plan are identical to Figure 5.5c up to the CANDO.PR of CAUSE_TO_WANT. OSCAR is again trying to achieve the WANT.PR of MOVE, namely that USER WANT USER MOVE FROM INROOM TO OUTROOM. Eventually, the propositional content of the INFORM becomes SYSTEM WANT USER MOVE FROM INROOM TO OUTROOM. In this instance, OSCAR does not need to proceed through CONVINCE since an INFORM of a WANT produces the necessary effects. Testing the CANDO.PR of INFORM determines if the system believes this proposition. Clearly, it does since the MOVE by the USER is in the SYSTEM's want space, SBSW. The WANT.PR of INFORM is trivially true, as before, and thus the plan is complete.

Whenever OSCAR's goal is to get the user to believe some proposition P, and OSCAR believes P, it can plan an INFORM

of P speech act. The following two sections point out how such planning can lead to other utterances.

5.6.4 Stating a CANDO

One of the preconditions of REQUEST is that the RECIP BELIEVE RECIP CANDO OBJ. One way to achieve this condition, when requesting the user to do something, is to plan SYSTEM INFORM USER THAT USER CANDO OBJ (with variables replaced by their bindings). This can only be done provided that SYSTEM BELIEVE USER CANDO OBJ is true. Hence, OSCAR would plan a speech act underlying the sentence "You can open the door."

5.6.5 A New Precondition for CAUSE_TO_WANT

Another variation on the same theme is to redefine CAUSE_TO_WANT to check for a friendship relationship between the two participants. The new definition is:

```
define cause_to_want as event
CASEFRAME:    (agt1 := a person) cause (agt := a person)
              to want to do (obj := a proposition)
CANDO.PR:     agt believe
                   agt1 believe
                        agt1 want obj
              and
              agt believe
                   friend of agt is agt1
.
.
.
```

FIGURE 5.6e

A NEW DEFINITION OF CAUSE_TO_WANT
INVOLVING THE RELATION FRIEND.

where FRIEND is an appropriately defined binary relation and everything else is the same as the previous definition of CAUSE_TO_WANT in Figure 5.5b. (To see how to define FRIEND as a symmetric relation, see [Levesque 1977].)

To achieve the goal of LOC OF USER IS OUTROOM, OSCAR would proceed as before and plan either a REQUEST that the user move or an INFORM that the system wants the user to move. However, in a parallel plan branching from the CAUSE_TO_WANT proposition (because of its conjunctive CANDO.PR) we would find a plan to get the user to believe that the system was its FRIEND. There is no need to expand that plan here since it follows exactly from Figure 5.6c where the proposition P is FRIEND OF USER IS SYSTEM. Thus, OSCAR would be generating the speech act equivalents of:

"I'm your friend. Can you come outside?"

"I'm your friend. I want you to come outside."

5.6.6 Summary of Planning INFORMs

OSCAR will plan to INFORM the user of some proposition P whenever its goal is to get the user to believe P is true, i.e. whenever OSCAR believes the user needs to know P. Such plans can arise when trying to achieve either the CANDO.PR's or WANT.PR's of other actions, as the above examples show. More than one speech act can be generated from a single non-linguistic goal because of conjunctive preconditions to previously planned actions. But what has not been shown is how OSCAR would decide to issue a REQUEST rather than an INFORM of a WANT when attempting to influence someone's behavior. The similarities and differences between these two ways of achieving the same goal will be discussed later in this chapter.

5.7 An Extended Example

In order to exercise our methodology more fully, OSCAR's planning capabilities will be demonstrated in an extended version of the domain of discourse proposed by Power [1974]. I will show how OSCAR can plan a variety of speech acts, and thus will demonstrate what conversational competence OSCAR has. Planning decisions that OSCAR (with occasional help from its oracle) makes will be shown to influence the speech acts (their kind and content) that are planned. The plans that OSCAR can produce incorporate REQUESTs and INFORMs in the fashion detailed in sections 5.5 and 5.6. Planning for other speech acts may employ a simulation of the user's planning.

5.7.1 Power's Domain of Discourse

This domain is the one described in Chapter 1. It consists of a room with a swinging door. The door has a lock that can only be unlocked from the inside. Hence, OSCAR's simple model of this world contains three new operator schemata: MOVETHRUDOOR, PUSHDOOR, and UNLOCKDOOR. (The earlier action MOVE is now supplanted by MOVETHRUDOOR.) The first is what an agent does to change LOC from INROOM to OUTROOM (or vice-versa) when the door is OPEN. PUSHDOOR changes the DOORPOSition of the door from (or to) OPEN to (or from) CLOSED. In order to do this, the door must be UNLOCKED. UNLOCKED is the value of the relation FASTENING that obtains when an agent, who is located at INROOM, performs the action UNLOCKDOOR on the LOCKED door. The rest of the actions and states (e.g. speech acts, CONVINCE, CAUSE_TO_WANT, etc.) are as they were defined in 5.5 and 5.6.

5.7.1.1 Initial Configuration

In the initial configuration, OSCAR believes that it shares knowledge with the user that it's location is OUTROOM, the user's is INROOM, the door is CLOSED and LOCKED, and that it and the user are FRIENDs. OSCAR also believes that the user is aware of the same action definitions as the ones it is using. The creation of this configuration of beliefs via MDP into OSCAR is shown in Figure 5.7a. This creation process is longer than that of Chapter 1, Figure 1, since the MUTUAL_BELIEF relation that will create a strong version of shared knowledge is not defined until Chapter 8.

```
add ( (loc of system is outroom)
              and
        ( (loc of user is inroom)
                and
            (doorpos of door1 is closed) )  )
OK
add ( (fastening of door1 is locked)
              and
        ( friend of user is system) )
OK
add ( user believe ( (loc of system is outroom)
                          and
                    ( (loc of user is inroom)
                            and
                        ( doorpos of door1 is closed))))
OK
add ( user believe (  (fastening of door1 is locked)
                            and
                      (friend of user is system) ) )
OK
```

FIGURE 5.7a
THE INITIAL CONFIGURATION OF BELIEFS

5.7.2 A Sample of OSCAR'S Range

The utterances listed here include speech acts that are planned from this initial configuration (and other slightly different ones) where the goal to be achieved is getting into the room. The current starting state is one of maximally shared knowledge about the world. Such states require less to be said than ones where the speaker knows little about the hearer's beliefs. Therefore, the list that follows includes optional utterances that are realizations of speech acts OSCAR can plan given a different state of knowledge.

One aspect of evaluating OSCAR's conversational competence is seeing if a person would issue the same speech acts as OSCAR when given the same circumstances. Rather than engage in extensive psychological testing, I simply point out when one of OSCAR's speech acts seems completely unlikely. OSCAR also does not issue some speech acts that a human might. In both of these cases, ways in which OSCAR's behavior can be improved are discussed.

5.7.2.1 What OSCAR Can Say to Get Into the Room

The following sentences are realizations of speech acts that OSCAR can plan in order to get into the room.

1. "Can you unlock the door?"

   Alternatively:

   a. "I'm your friend. Can you unlock the door?"
   b. "The door is locked. Can ycu unlock it?"

2. "I want you to unlock the door."

   Alternatively:

   a. "I'm your friend. I want you to unlock the door."

3. "Can you open the door?"

   Alternatively:

   a. "I'm your friend. Can you open the door?"
   b. "The door is closed. Can you open it?"
   c. "The door is locked. Can you open it?"

4. "I want you to unlock and open the door."

Alternatively:

a. "I'm your friend. I want you to unlock and open door."

OSCAR should be able to say:

5. "I want to come in."

   Alternatively:

   a. "I want to come in but the door is locked."

6. "Can you help me get in? The door is locked."

   a. "I want you to help me get in, the door is locked."

7. "I'm Joe's friend. Can you open the door?"

8. "I'm a police officer. Open the door."

5.7.3 Planning Sentences 1 and 2.

The plans for sentences 1 and 2, in Figures 5.7b and 5.7c, are straightforward adaptations of our prototypical examples of planning REQUESTs and INFORMs. The plans for sentences 1 and 2 are identical until OSCAR attempts to achieve the precondition of CAUSE_TO_WANT. At this point it can complete the plan via either of the methods discussed in sections 5.5 and 5.6.

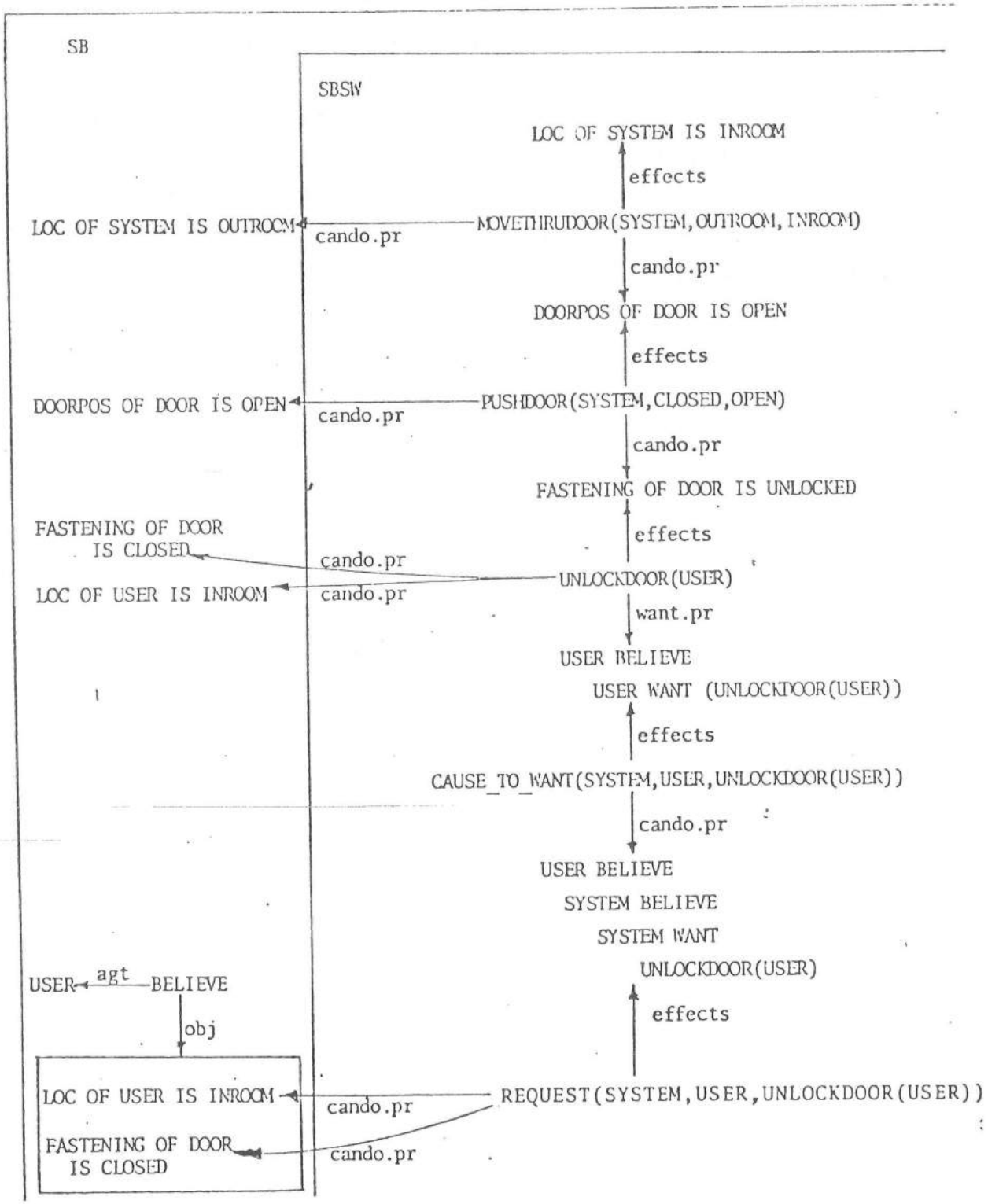

FIGURE 5.7b

PLAN FOR "CAN YOU UNLOCK THE DOOR?"

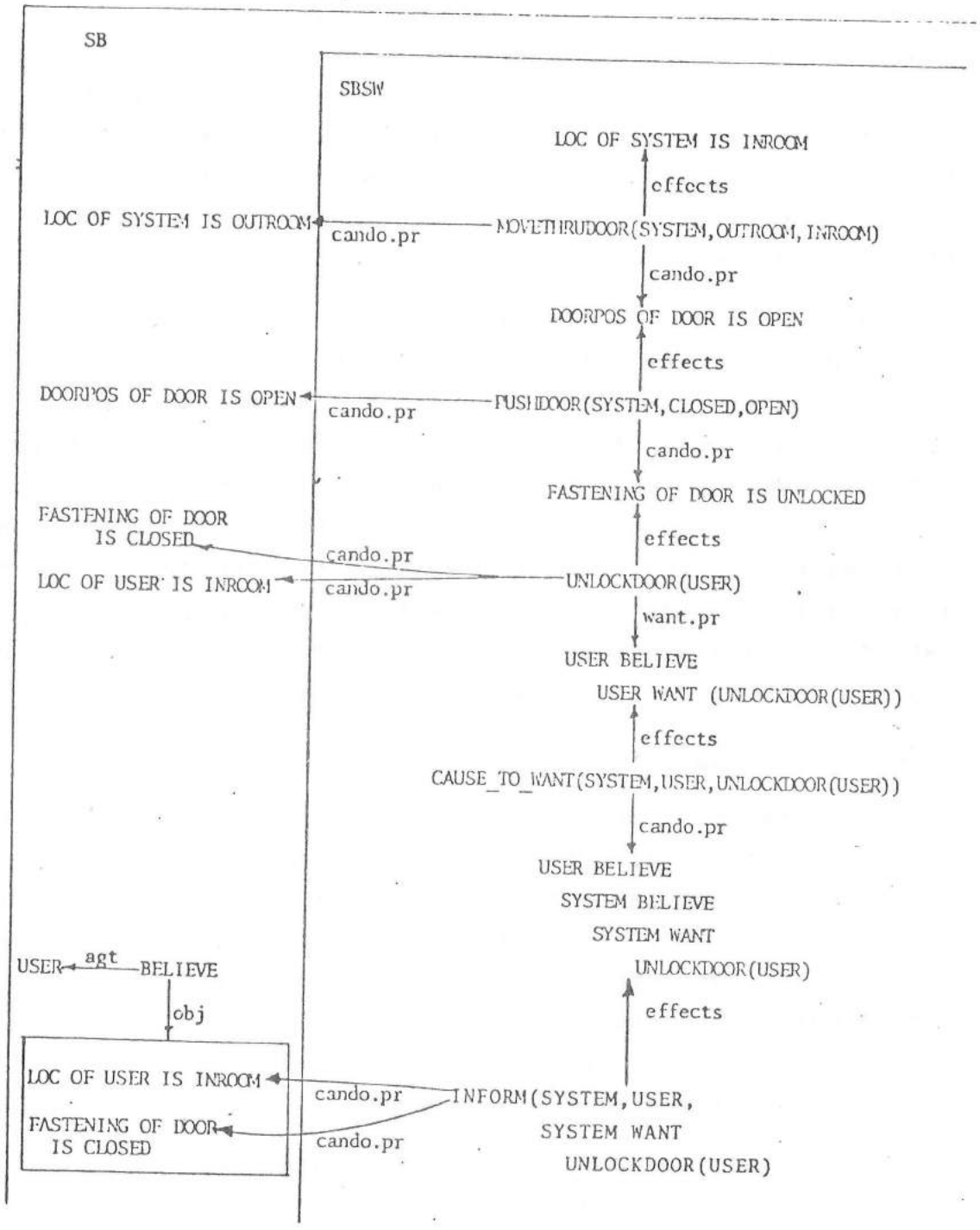

FIGURE 5.7c

PLAN FOR "I WANT YOU TO UNLOCK THE DOOR."

Essentially, OSCAR realizes that in order to be INROOM, it must MOVETHRUDOOR. In order to get the door to be OPEN, enabling the MOVETHRUDOOR, OSCAR plans a PUSHDOOR act. However, the matching of the effects of PUSHDOOR to the goal of the door's being OPEN does not determine who will do the pushing. No new bindings are available from testing the CANDO.PR's so OSCAR invokes its oracle and obtains SYSTEM as the binding. Notice that, automatically, the WANT.PR of PUSHDOOR is now true.

What remains is for the door to be UNLOCKED. Again, the matching of the effects of UNLOCKDOOR leaves the agent of this action unbound. However, unlike the previous case, the testing of the CANDO.PR's determines that the condition LOC OF AGT IS INROOM is true when AGT is bound to USER, since OSCAR believes LOC OF USER IS INROOM. At this point, OSCAR must achieve the user's wanting to UNLOCKDOOR. In the usual fashion, this leads to either the REQUEST or the INFORM of a WANT provided that SYSTEM is chosen as the one causing USER to decide to want to do the act.

5.7.4 Planning Sentences 3 and 4

These two sentences differ from sentences 1 and 2 solely by the propositional content of the speech acts (and by an extra speech act in sentence 4). OSCAR will try to get the user to do PUSHDOOR (the act realized by "open the door") rather than UNLOCKDOOR after it has chosen USER to be the AGT of PUSHDOOR. Then OSCAR must get the USER to want to PUSHDOOR, yielding, in nearly the standard way, the REQUEST or INFORM of WANT speech acts underlying our two sentences.

In this instance, the CANDO.PR's of the REQUEST, referencing the preconditions of PUSHDOOR, are not believed to be true -- the door is locked. Hence, OSCAR must check if it believes the user can unlock the door, and if he believes he can. OSCAR now has a number of goals to be

achieved. Given this particular planning algorithm, one choice for the next goal eliminates superfluous processing.

We would not want OSCAR to have to request that the user unlock the door in addition to pushing it. Intuitively, what we would like to have happen is for OSCAR to believe that once it requests that the user push the door, the user will want to perform all the actions necessary to enable the opening of the door. OSCAR avoids making a second request based on its simulation of the user's planning. The goal that leads to such planning is the one of determining if the user believes he can push the door open. This requires that OSCAR believe the user thinks the door is unlocked. However, no such proposition exists in SBUB.

Recalling the definition of CANDO in 5.5.4, if CANDO is tested in a belief space other than that of the PLANNER, it will recursively call the planning routine to achieve the appropriate condition. In the case of testing FASTENING OF DOOR IS UNLOCKED in SBUB with the USER as the PLANNER, CANDO will simulate the user's planning to achieve the door's being UNLOCKED. Fortunately, the configuration of user beliefs is such that the preconditions of UNLOCKDOOR are true; CANDO thus returns a SUCCESS outcome. This embedded plan is shown in Figure 5.7d.

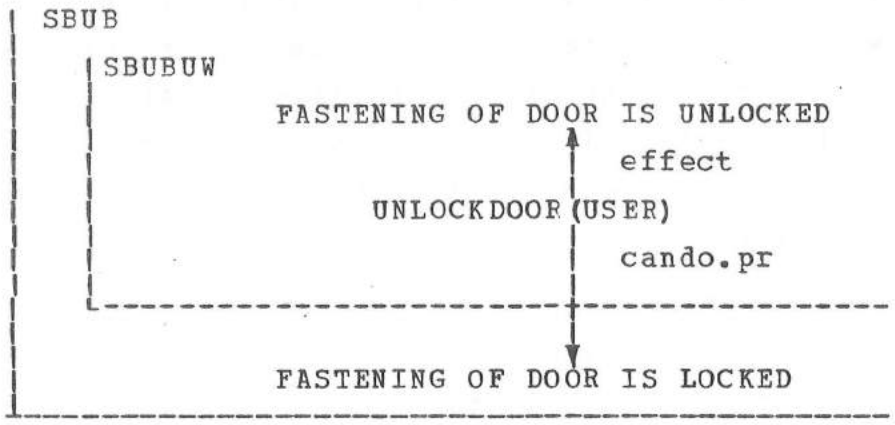

FIGURE 5.7d

A PLAN EMPLOYING THE USER'S SPACES

OSCAR's planning to achieve the other CANDO.PR of PUSHDOOR adds a goal that the user unlock the door. But, of course, the user must want to unlock it. The WANT.PR of UNLOCKDOOR states:

USER BELIEVE
    USER WANT
        USER UNLOCKDOOR

This precondition is true since USER UNLOCKDOOR appears in space SBUBUW from the earlier embedded planning! Hence, OSCAR would request that the user open the door but not unlock it.

Bear in mind that there is a distinction to be made between planning for the user to do something versus simulating the user's planning to do that same thing. The first plan is in SBSW and can lead to a REQUEST by the system. The latter plan takes place in SBUBUW.

5.7.4.1 Limitations of Simulating the User's Planning

The above technique of employing a simulated plan in ruling out a WANT.PR, is of limited utility since a number of circumstances must arise before it is applicable. First, the CANDO.PR of the act being requested (e.g., PUSHDOOR) must not be initially true in SBUB. If OSCAR believed that the user (erroneously) thought the door were already unlocked, OSCAR would not develop a plan for him to unlock it, and hence would request that he both open and unlock the door. Using just the concept of BELIEVE, OSCAR has no reason to check the user's beliefs against its own. The relation KNOW could be defined for this purpose. Of course, OSCAR itself can never KNOW anything; it can only BELIEVE.

Secondly, the application of this technique requires that the WANT.PR of an event (e.g., PUSHDOOR) be established (leading to a REQUEST) before the other preconditions. Then, in checking preconditions via the CANDO.PR's of a REQUEST, OSCAR first would need to achieve the user's

believing he could do the requested action. In planning the PUSHDOOR act, if OSCAR attempted to get the door unlocked before attempting to get the user to want to open the door, there would be no embedded plan to use at a later stage. Part of the reason for incorporating an oracle into the planning process was to discover such restrictions on the choices of what to do next. Hence, I see nothing wrong with sequencing restrictions per se. What is open to question, however, is the generality of this particular one.

Finally, while the embedded planning technique eliminates the second request of UNLOCKDOOR, it only does so when the means chosen to achieve the user's wanting to open the door was a REQUEST. If OSCAR chose to first issue an INFORM of a WANT, it would have to plan to get the user to want to unlock the door -- i.e., it would plan "I want you to unlock and open the door" (utterance 4). The cause of this difficulty will be discussed in section 5.8.2.

5.7.5 Optional Utterances

When we attributed beliefs to OSCAR and its model of the user, we stated that OSCAR believed it was shared knowledge that it and the user were friends. This need not have been the case. If OSCAR believed it and the user were friends, but did not believe the user had such knowledge, then OSCAR would have an extra goal to be achieved -- USER BELIEVE FRIEND OF USER IS SYSTEM -- as a prior condition to CAUSE_TO_WANT. Achieving this goal via an INFORM, as discussed in section 5.6.5, accounts for the "I am your friend" for utterances in groups 1 through 4.

5.7.5.1 Utterances by Embedded Planning

OSCAR employs its knowledge of its user in deciding to issue INFORM speech acts when it believes the user needs to know a certain proposition. This is the reason behind the INFORMs planned in utterances 1b, 3b, and 3c. One way of

determining that the user needs to know something (e.g., that the door is closed) is to simulate the user's planning in space SBUBUW and discover where it is missing information. Such simulations are triggered by CANDO when a direct search in SBUB for the preconditions of the act in question yields UNK.

No embedded planning need be done on REQUEST's precondition USER BELIEVE USER CANDO UNLOCKDOOR to produce utterance 1b. CANDO need simply return USER BELIEVE FASTENING OF DOOR IS LOCKED as the new goal. In our usual way, this leads to an INFORM speech act.

The planning algorithm could spend a long time deciding if the user is able to do some action. I have not worked on strategies for deciding when to give up. Hence, when asked which goal to work on next, I may signal the embedded planning to halt and return a conjunction of goals, for which no methods of achievement have been found, to CANDO. At this point, CANDO would know what the obstacles were and pass them on to be added to the main plan. Thus OSCAR would construct a specific precondition for REQUEST -- namely exactly what it was that the user ought to know. In testing the CANDO.PR's of a REQUEST that the user PUSHDOOR, OSCAR would test the precondition ∃xUSER BELIEVE (DOORPOS OF DOOR IS x). The variable occurs here since there is no argument filling the SOURCE position for PUSHDOOR. (Such quantified beliefs will be discussed extensively in Chapters 6 and 7.) This conjecture is returned to CANDO, perhaps in conjunction with the form underlying utterance 3c -- USER BELIEVE (FASTENING OF DOOR IS LOCKED). In the former case, the matching of variable x against the data base yields CLOSED as a value. Utterance 3b then follows by the standard INFORM-CONVINCE route (given the extensions to be discussed in Chapter 6).

OSCAR could have been designed to give up after expending a certain amount of resources on an embedded plan.

Since this is just an arbitrary heuristic, there was no point in implementing it, given the goal of demonstrating a range of speech act behavior.

In simulating the user's planning to achieve his believing some proposition P (say, the door's position, as above), it will turn out that OSCAR could plan for the user to ask OSCAR (or someone else) a question about P. Since OSCAR is prepared to tell the user what he needs to know, its simulation of the user's planning such a question is superflous, albeit legitimate. However, I know of no principled way of preventing the planning of such questions.

A human might not state "The door is locked. Can you unlock it?" since he would probably reason that by requesting someone to unlock the door, he is communicating that he believes the preconditions of the act being requested are true. The reasons why OSCAR is incapable of such planning will be discussed in the next chapter.

A final point is that if we did not separate the two kinds of preconditions on the definitions of acts, we could encounter an infinite recursion. Let us assume the preconditions were undifferentiated. Testing the preconditions of REQUEST would cause a test of USER BELIEVE USER CANDO ACT, that would succeed only if the user believes he wants to do ACT. If that failed, OSCAR would determine if the user could plan to achieve its wanting to do that action. This simulation might then create a request by the user that the system request him to do the act, in order to achieve the user's knowing the system wants him to do it. There would be no way of preventing an infinite regress of the parties' each trying to get the other to get themselves to want to do the act. By splitting up the preconditions of every act into two classes, we can prevent such problems by only testing CANDO.PR's.

5.7.6 Sentences 5 and 6

Currently, OSCAR cannot plan sentences 5 and 6. The reason for the difficulty with sentence 6 is that there is no act HELP to request. As previously mentioned, helping is being treated as an act whose body is a collection of strategies that the helper performs once he knows the goal of the person whom he is supposed to help. The stating of this goal is what leads to the INFORM "I want to be in" in sentence 5. The goal is represented as a state -- viz. LOC OF SYSTEM IS INROOM. Previous statements of goals have been statements of desired acts. The difference between statements 3 and 5 is, I believe, one of OSCAR's knowing what act to try in statement 3, but not knowing so in statement 5. The definition of HELP should be formulated so that it is applicable whenever OSCAR is trying to achieve any goal state.

Utterance 6 differs from 5 by its containing both an explicit request for HELP to achieve a particular goal and a statement of the obstacle. The explicit request would indicate that the speaker does not believe the hearer wanted to help. In circumstances where people are expected to be helpful, such a request would probably not be made -- only a statement of the goal and, perhaps, of the obstacle would be needed. The formulation of a definition for the act HELP, the beginnings of which can be found in Chapter 9, would enable OSCAR to plan sentences 5 and 6 in the usual fashion.

5.8 Some Problems

A few remarks might be useful here about some of the problems that were glossed over in an attempt to simplify matters for expository purposes. These points will cover the problem of planning for someone to make an inference and differences between REQUEST and INFORM of WANT.

5.8.1 Planning for Someone to Infer

Instead of saying to the user "I am your friend, could you open the door?", we might want OSCAR to be able to say "I'm Joe's friend..." How might it go about planning such a speech act? It would have to believe that stating a friendship relation between itself and someone else would cause the user to believe that it and the user were friends. In the same fashion, we might want OSCAR to plan for its user to infer that it is in a position of authority over the user. Such an inference might be involved in planning such utterances as "I'm a police officer, open the door." In general, OSCAR needs to plan for the hearer to perform an inference. Hence we might want OSCAR to have an INFER act in its repertoire, describing what it takes to get someone to make an inference. Of course, the preconditions of such an act would be problematic.

A possible way to discover that the FRIEND or AUTHORITY precondition to CAUSE_TO_WANT is true is to have the relations themselves indicate what inferences would satisfy them. Such propositions would be returned as the CONJECTURE parts of UNKNOWN outcome propositions. For instance, testing USER BELIEVE (SYSTEM HAS-AUTHORITY-OVER USER) might return as a new conjecture, USER BELIEVE (OCCUPATION OF SYSTEM IS POLICE-OFFICER). Such a facility is easy to implement given appropriate definitions of the testing routines for relations like HAS-AUTHORITY-OVER.

We are assuming that the user will make the inference we have intended. The stipulation of chapter 4 was that we could not expect the user to make all possible inferences. For this reason, the class of inferences of which we could hold such expectations was restricted to those driven by the specialist routines. However, perhaps there is a more general principle operating here -- a kind of conversational maxim (cf. [Grice 1967]). Since OSCAR would be stating an extra proposition, the user is entitled to think that it has

something to do with the situation and thus he would be more motivated to discover why OSCAR stated it. Hence, he might then be lead to infer what he ordinarily would not. I have not investigated this topic and thus I leave the above speculations as material for future research.

5.8.2 REQUEST vs. INFORM of WANT

The astute reader may have noticed that my formulation of Searle's versions of REQUEST and INFORM does not maintain equivalence, in terms of conditions tested, between a REQUEST and an INFORM of a WANT. The difference lies in that nowhere in the planning of the INFORM did OSCAR test if the hearer believed he could do the action. Thus no simulation of the user's planning would occur via an INFORM of a WANT. The other precondition to REQUEST (i.e., RECIP CANDO OBJ) is tested by the planning algorithm in the process of planning any action and hence is completely redundant.

As mentioned previously, one can utter "I want..." and not intend that the hearer do anything once he knows about your goal. A hearer, of course, must determine what the speaker's intentions are vis-a-vis the goal in order to be sure of the "request" interpretation. However, if one does intend to get a hearer to do some action, it appears that the conditions under which REQUESTs and INFORMs of WANTs are planned ought to be the same. An objective of the research so far has been to show that they can be planned to achieve exactly the same effects and hence could be considered identical.

In order to force an equivalence between a REQUEST and an INFORM of a WANT, we could define various actions differently. We could remove both of the CANDO.PR's from REQUEST and add a new CANDO.PR to CAUSE_TO_WANT that states AGT BELIEVE AGT CANDO ACT. In other words, the new definition would say that a person will decide to want some

action if he believes someone else wants him to do it and if he believes he can do it. With these changes, both ways of getting someone to want to do some action would involve his believing he is able to do it.

After doing this, where would that leave our definition of REQUEST? It would have no preconditions (save a WANT.PR that is applicable to most acts). Should we have an action REQUEST at all? Would INFORM of a WANT suffice?

If we drop REQUEST as a speech act, how would such utterances as "I request that you open the door" be generated? Recall that the act REQUEST was supposed to be the prototypical member of the directive class of speech acts. Other members of that class would have additional preconditions (e.g., SUGGEST) that would differentiate them from a simple INFORM of a WANT. Since REQUEST was chosen as the prototype of a class of acts formed around the similarity of their effects, it is not surprising that REQUEST reduces to a way of achieving those effects.

The more general issue is that need to determine criteria that can tell us when we should be satisfied (or unsatisfied) with our act definitions. The only criteria I can supply at this time are ones of input-output behavior; the system should account for a specified class of utterances. The reason for changing to this new representation is that the old one could not handle a number of utterances. (Similar reasons will lead us, in Chapter 6, to refine the definition of INFORM.) For instance, "I want ycu to open the door. It's locked" is not plannable via the old definitions of acts since OSCAR needs to be able to access the user's version of the CANDO.PR for PUSHDOOR, as is done with REQUEST, to realize that he needs to know about the lock.

While the issue of how to define speech acts is far from settled, I have found that the methodology of incorporating

speech acts into a planning system to be beneficial to detecting unforseen problems with what would otherwise seem to be reasonable act definitions.

5.9 Notes

1. The effects could become true via a sequence of other actions encoded as the body of MOVE. While designing the system with such decompositions in mind, I have not pursued this any further. See [Sacerdoti 1975] for discussion of the uses of decomposition in planning systems, and [Allen forthcoming] for a discussion of how decomposition relates to plan recognition.

2. A more sophisticated planning algorithm might distinguish between these two kinds of answers. When given a FALSE answer, the algorithm might back up and try another alternative, while an answer of unknown might lead to its trying to find out.

3. Schmidt [1975] and Bruce and Schmidt [1974] have proposed similar definitions.

4. There is nothing in this definition that prevents the variables AGT and AGT1 from being bound to the same person. A more sophisticated system would define the set from which AGT is taken as "everyone but AGT1". Such definitions have not been attempted here.

CHAPTER 6

PLANNING QUESTIONS AND THREE-PARTY SPEECH ACTS

Contents:

6.0 Introduction

The purpose of this chapter is to show how a specific need for information can result in the asking of a question. Questions are interesting for a number of reasons. First and foremost is that in order to plan a question one must believe that the hearer of the question knows the answer. However, the speaker should not know what the hearer believes the answer to be since if that were the case, he would not ask. The representation of such knowledge has been problematic for philosophers (cf. Quine [1956], Kaplan [1969], Hintikka [1963], and others) and has never been addressed by any AI natural language system to date. OSCAR, however, has a direct representation of such beliefs.

Much of this and the next chapter will be devoted to a discussion of their representation, semantics, and use.

Secondly, questions and three-party speech acts are interesting since they are compositions of REQUEST and INFORM. OSCAR will construct questions from their parts when it wants to know something. Hence the types of questions considered here (e.g., WH, polar, existential, teacher/student) provide a strong test of conversational competence -- a test of representation and of planning of speech acts. Again, the methodology is to show how OSCAR generates a variety of questions.

Finally, the ability to plan and ask questions gives OSCAR the basis for engaging in mixed initiative and clarification dialogues. Other natural language systems (cf. SCHOLAR [Carbonnell and Collins 1973] and SOPHIE [Brown and Burton 1975]) engage in mixed initiative dialogues but cannot be extended to three-party conversations since they would have no basis for choosing to ask a question of one person over another. The reason for this is that none of these systems maintains an explicit model of the user. OSCAR, however, is ideally suited for making such decisions since it can believe that person A knows the answer but that person B does not, and thus it can choose to formulate a question to A.

This chapter will begin by discussing what questions are and why they cannot be planned with the current machinery. These problems will lead to the major representational distinction of this chapter -- knowing that vs. knowing what. Once the representation questions have been cleared up, the means by which OSCAR plans each of the question types identified above, in addition to a number of three-party speech acts, can then be described.

6.1 Questions and Three-Party Speech Acts

Numerous people (e.g., Gordon and Lakoff [1971]) have suggested that questions can be regarded as requests for information. In our terminology, a question is a REQUEST by a speaker S to a hearer H that H INFORM S of some proposition P. Other combinations of REQUEST and INFORM may be plannable, but not as questions. A REQUEST by S to H that H INFORM someone else, say T, of something is not a question since person T is not the speaker of the REQUEST. REQUEST(S,H,INFORM(T,H,P)) is an example of a combination of REQUEST and INFORM that is not plannable since OSCAR would ask T to do the INFORM, not H.

The propositional content of the INFORM is what differentiates one kind of question from another. I am not simply referring to the differences in content between WH and polar questions that are explicitly marked at the surface level, but instead to a classification of the types of propositions involved.

Lehnert [1977] has proposed a number of categories such as:

- Causal Antecedent -- "Why did John go to New York?"
- Instrumental/procedural -- "How did John go to New York?"
- Expectational -- "Why didn't John go to New York?"

In her categorization, lexical classifications of questions (WH, Y/N, etc.) may map onto numerous "conceptual" categories that involve questioning relationships between objects and propositions. We will be primarily interested in grosser classifications of the propositional content of questions. For instance, the point of a WH-question is to discover missing information in some proposition, while that of a polar question is to know the truth-value of a proposition. Existential questions are those that can be paraphrased by "does there exist a relationship of a certain type between x and y". Finally, teacher/student questions

are of a different genre in that their generation is not necessarily dependent upon the teacher's wanting to know the answer.

Three-party speech acts depend upon all the mechanisms that are needed to plan questions. An example of this class is the following request to Bill: "Ask Tom to tell me where Mary is."

Other examples are:

- "Ask Bill to tell you where Mary is."
- "Ask Bill to tell me (you) whether Mary is at home."
- "Bill knows where Mary is. Ask him to tell you where she is and then tell me."

Nothing about these speech acts is peculiar to three people; any number could, in principle, be involved.

6.2 Problems with the Propositional Content of a Question

What first needs explanation is the origin of questions. Questions seem to arise because the questioner "wants to know" something. Why he wants to know it -- to complete a plan, to find out if the hearer knows it, etc. -- is a different, and interesting, issue that will be discussed later. "Wanting to know" has a direct representation in the system, namely, SYSTEM WANT (SYSTEM BELIEVE P). In developing a plan, then, there would be a SYSTEM BELIEVE P proposition in space SBSW. In fact, OSCAR always wants to believe something in that its representation of any goal P is "SYSTEM WANT SYSTEM BELIEVE P". Hence, all plans in Chapter 5 are a bit simplistic in that they do not mention the preceding SYSTEM BELIEVE on each goal.[1] This causes no problems since, in general, they are ignored. However, in planning questions, we cannot engage in this luxury.

The goal SYSTEM BELIEVE (LOC OF USER IS OUTROOM) is somewhat ambiguous as it stands since there is no mention of time in the proposition. There is no distinction between wanting to know that P is now the case vs. wanting to make P

true in the future and thus believing that P then holds. For instance, when wanting to know whether the train to Montreal leaves at 3 P.M., we do not want to plan to change its schedule in order to say yes. Instead, we want to form a question about its departure time. Hence, OSCAR needs the ability, when confronted with a proposition of the form SYSTEM BELIEVE P, to either plan to achieve the entire proposition, which leads to a question, or simply plan to achieve P. The criteria for deciding on the goal to be achieved would thus be based on the time specified for the belief.

Since OSCAR does not currently deal with temporal information, it employs a heuristic, dependent upon whether or not SYSTEM BELIEVE was prefixed separately, to determine whether or not to achieve the full goal.

Every goal being planned now has a BELIEVE in front and thus deciding how to achieve one will always invoke the dfunction for BELIEVE. This function uses some simplistic heuristics, based on the number and kinds of embedded propositions in the goal, to determine the producing action.

6.3 A First Attempt at Planning a Question

Let us see what happens if OSCAR tries to plan a question with the above structures (including the previous definitions of INFORM, CONVINCE, etc.) The goal is SYSTEM BELIEVE (LOC OF MARY IS INROOM).[2] Figure 6.3a illustrates the situation.

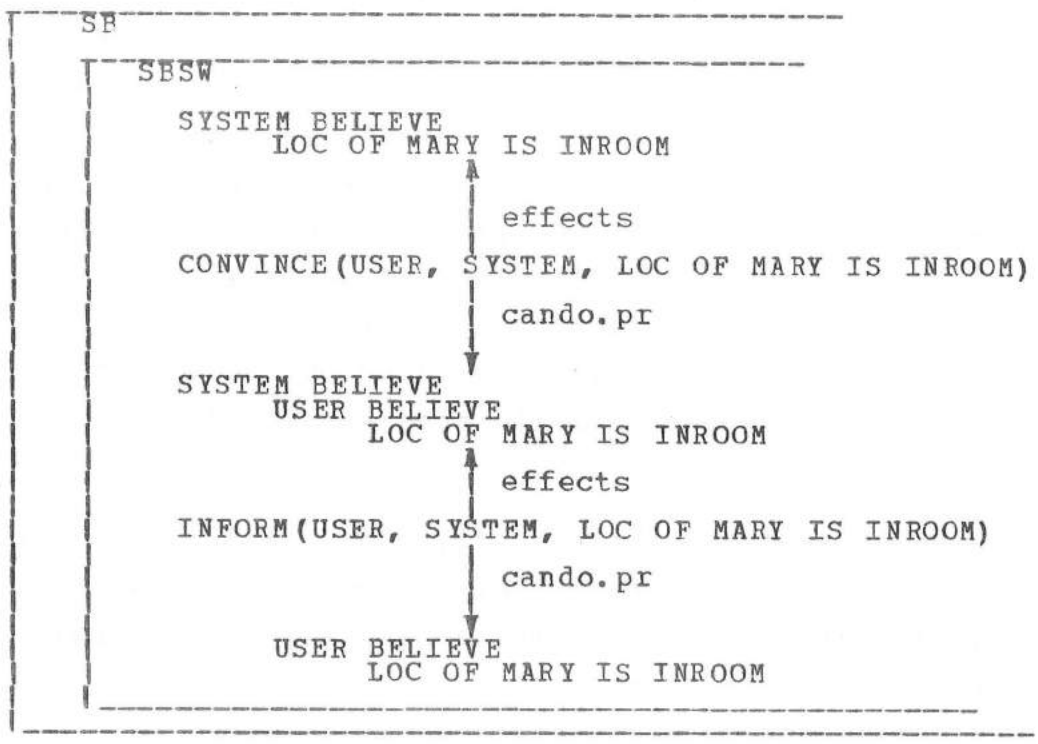

FIGURE 6.3a

PROBLEMS WITH PLANNING QUESTIONS

Employing the BELIEVE.DFUNCTION, OSCAR chooses CONVINCE to achieve this goal, and it matches. After invoking the oracle to see whose beliefs should be convincing, the precondition that is tested is SYSTEM BELIEVE (USER BELIEVE (LOC OF MARY IS INROOM)). If this test succeeds, planning can stop since OSCAR can then conclude that the user is in the room. Hence, we must assume the precondition fails in order to reach a question.

Now, the above goal is achievable via INFORM(USER,SYSTEM,LOC OF MARY IS INROOM). However, the CANDO.PR of INFORM that will be tested is USER BELIEVE (LOC OF MARY IS INROOM). But, because of the prefixed SYSTEM BELIEVE and the fact that the world space is SB, this CANDO.PR of INFORM is the same as that of CONVINCE! In fact, it always will be in such plans. There is no reason why it should fail for CONVINCE and succeed for INFORM.

6.4 Intuitions behind the Difficulty

The preconditions worked fine when planning an INFORM with SYSTEM as AGT. Why should they go awry here? To see why, let us assume that OSCAR wants to know where Mary is and decides to ask the user a question. For CONVINCE, it needs to know what the user believes, while for INFORM it needs to know simply that the user can do the INFORM -- i.e., that he knows where she is. OSCAR would have nothing to be convinced about with respect to Mary's location if Tom has told it that he has just visited Mary (and OSCAR believes him). It would thus believe that Tom knows where she is but could not conclude anything about her specific location from that fact.

Our first problem, then, is to devise a way for OSCAR to represent someone else's knowing the value of some expression without OSCAR's knowing it. In addition, in differentiating between the preconditions for INFORM and CONVINCE, we need to be able to state which of the two possibilities (knowing that vs. what) are wanted for which preconditions. Hence, we want first to make a distinction in representing values of expressions and secondly to make a distinction in the preconditions based upon the belief spaces in which we want to know the value. To this end, I define Known constants.

6.5 Known Constants

What does it mean to say

1. "Bill knows who Mary's husband is."?

Can I say Bill can pick him out in a crowd or he could draw his picture, or Bill knows where he lives? Such questions have troubled philosophers since the ancient Greeks. I have no intention of proposing criteria for knowing who, where, what, something is. On the contrary, what I will do is to

presuppose such criteria and hence make them primitive and unanalyzable. What this means is that I will supply a way of representing that the value of an expression is "known" and will not discuss how one decides when the value is "known".

There are a number of other concepts from which the new primitive needs to be distinguished. We can say that

2. "Bill believes (or knows) Mary's spouse is Tom."

In this case, we know which value Bill believes the expression SPOUSE OF MARY has, namely the constant TOM. We can also say that

3. "Bill believes Mary has a spouse."

In other words, Bill believes there exists a value for the expression. Our new concept of a "known value" falls between the two on a scale of "specificity". Case 2 is fully-specified since TOM is a constant. In other words there exists a particular object for "Tom", in our model of Bill's belief space, distinct from all other such constants. Case 3 is non-specific since it contains an existential (in a logical form, it would be BILL BELIEVE ($\exists$x [SPOUSE OF MARY IS x])). We cannot say that Bill knows who that spouse is, only that he thinks she is married. (He may only have seen a ring.) Case 1 is "stronger" than case 3 since we believe Bill knows more than just the fact that Mary is married -- he knows who her husband is. But case 1 is weaker than case 2 since for the former, we do not know which value Bill believes it to be while in the latter instance, we do.

Case 1 does not make sense with respect to the system's top-level belief space. If OSCAR believes only that Mary is married, then it would have an existentially quantified variable as the value of SPOUSE OF MARY. If it believes her spouse is Tom, then the object it is using to stand for Tom is the value of the expression. But OSCAR cannot say it

knows who Mary's spouse is without in fact having a constant as the value of SPOUSE OF MARY -- OSCAR either knows who he is or it doesn't. But one's knowledge of everyone else's beliefs is certainly weaker than one's knowledge of oneself, and hence it makes sense to have these objects only in embedded belief spaces.

6.5.1 Representing "Knowing That"

The distinction we are after involves the representation of objects, in an embedded belief space, that are neither constants (case 2) nor variables (case 3). OSCAR's constants are those objects "x" created with the MDP statement "x is a new instance of y". It is important to realize that any constants that OSCAR believes the user knows about must be constants for OSCAR as well.

As Figure 6.5a indicates, the representation of expressions is distinct from the representation of their values. In the figure, the expression SPOUSE OF MARY is connected to its value, a variable (indicated by "<>") with a VALUE edge. This expression's being in some belief space, say SBUB, states that the system believes the user believes Mary has a spouse. Notice that the object representing Mary (i.e. MARY) is a constant that is present in both SB and SBUB.

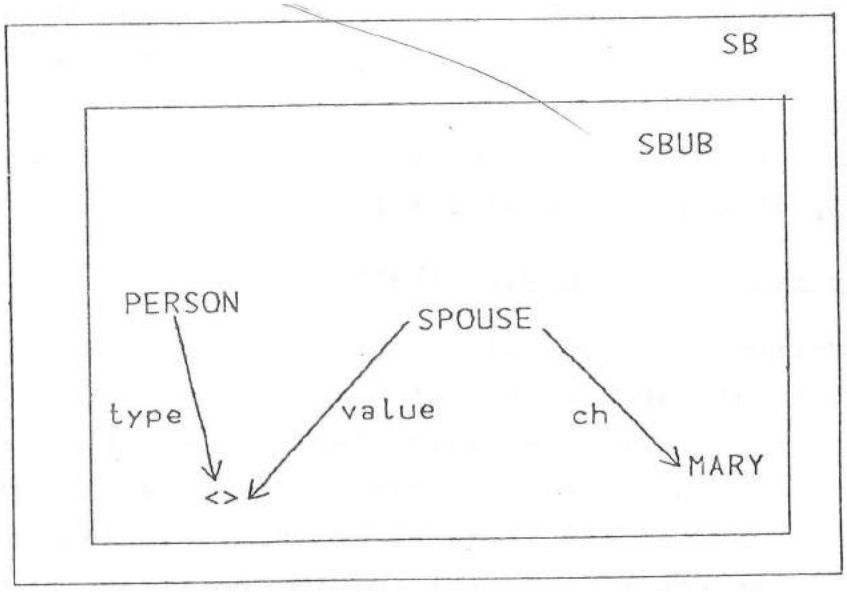

FIGURE 6.5a

"THE USER BELIEVES MARY HAS A SPOUSE."

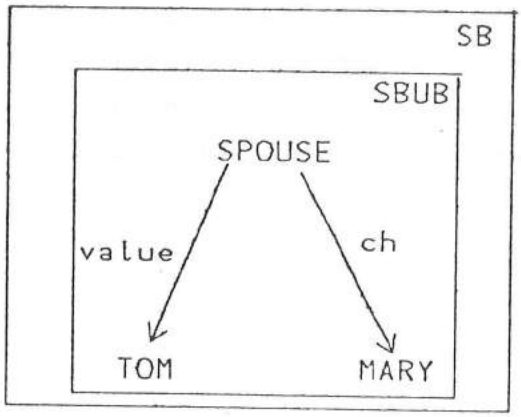

FIGURE 6.5b

"THE USER BELIEVES
MARY'S SPOUSE IS TOM."

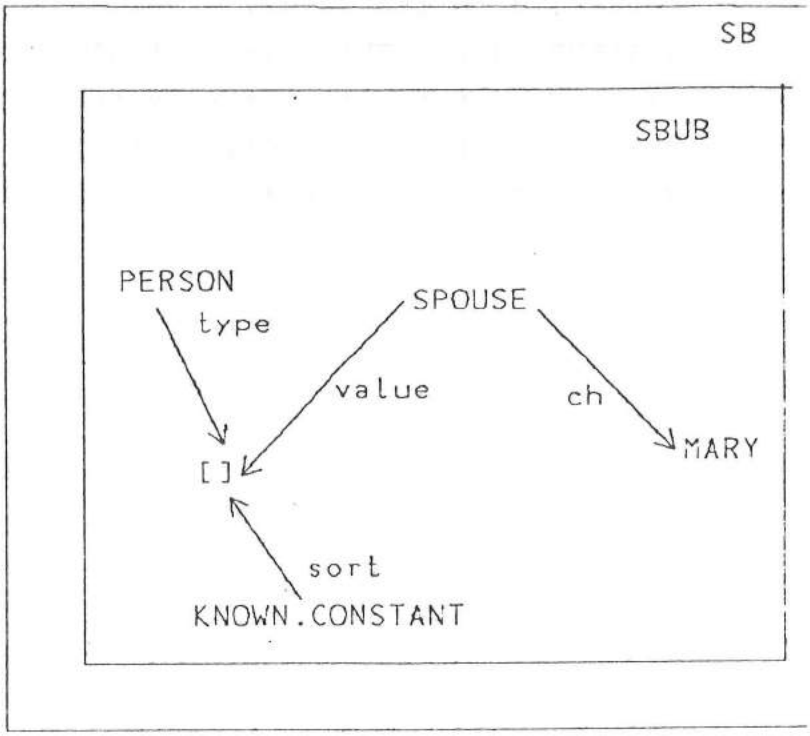

FIGURE 6.5c

"THE USER KNOWS WHO MARY'S SPOUSE IS"

(version 1)

Objects are classified by linking them by SORT propositions to one (and we shall see, perhaps more than one) of three classes: CONSTANT, VARIABLE, and (for our new distinction) KNOWN.CONSTANT. (In diagrams, the SORT propositions will be shown only where they are not obvious.) These propositions are in addition to "type" information that relates an object to the class(es) of which it is a member (e.g. PERSON). Figure 6.5b shows the representation (case 2 above) of a propcsition in SBUB with two constants, MARY and TOM, such that SPOUSE OF MARY IS TOM. Figure 6.5c shows the representation of "the user knows who Mary's spouse is." The object indicated by "[ ]" is a KNOWN.CONSTANT (or "k.constant") taken from the class PERSON.

I may believe that you know the value of some expression, but I do not. Since I believe you know what it is, I must believe it exists. However that object cannot be a constant for me or else I would know what it is, and I don't. Hence, the k.constant for you (in SBUB) is a variable for me (in SB)! What we now have is an object whose SORT depends upon the belief space in which it is viewed. Figure 6.5d indicates this situation.

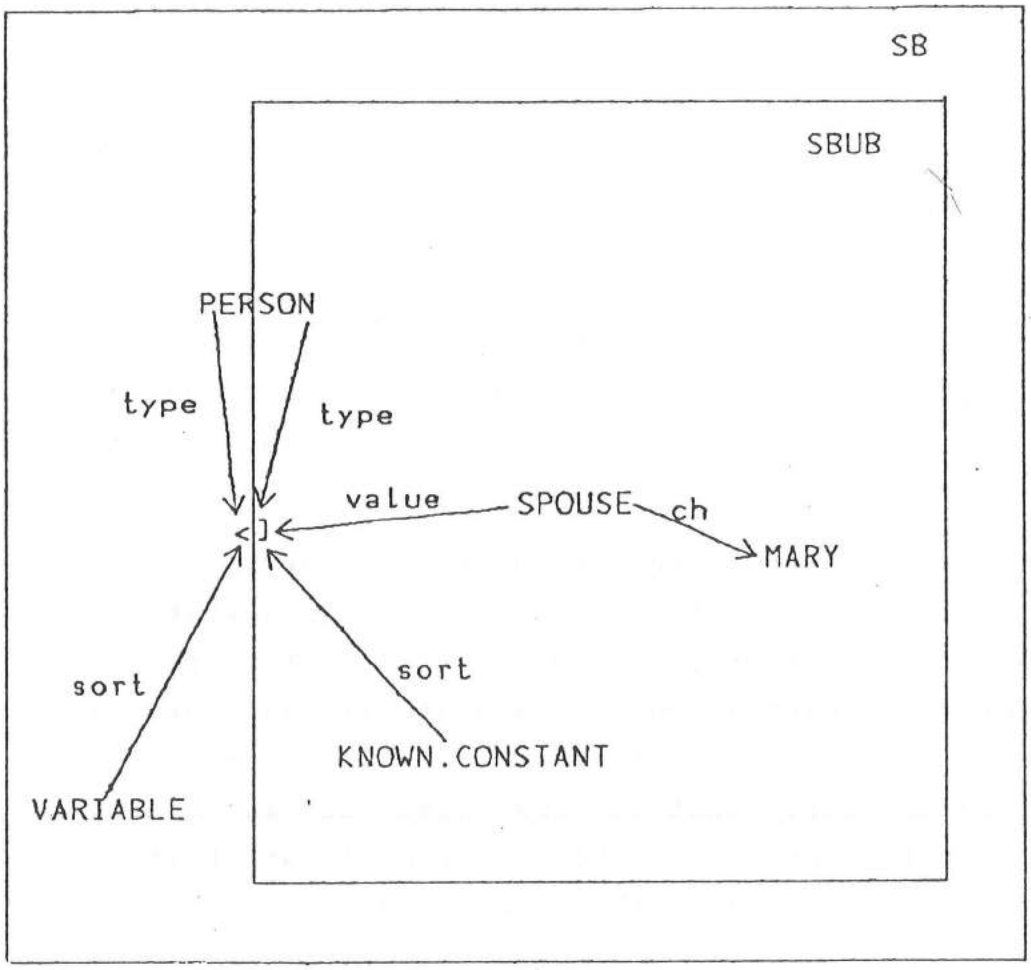

FIGURE 6.5d

REPRESENTATION OF "THE USER KNOWS WHO MARY'S SPOUSE IS"

The k.constant is now pictured as "<]" indicating its chimerical nature as being half constant and half variable. The SORT of the object is VARIABLE in space SB and is KNOWN.CONSTANT in space SBUB. In both spaces, the k.constant is of TYPE PERSON..

6.5.2 Creating Known Constants: Quantifying into a Belief

K.constants are created by "quantifying into" a belief space. For instance, to say that the user knows where Mary is, we would say:

```
SYSTEM BELIEVE
    THEREXISTS AN X SUCH THAT
        USER BELIEVE
            LOC OF MARY IS X
```

X is a k.constant within SBUB and a variable in SB. In order to indicate more explicitly the space in which we are to have a k.constant, I will use a different quantifer termed KC. (The next chapter presents a formal semantics for "quantifying into" a belief context.) The above proposition would then be written as:

```
SYSTEM BELIEVE
    USER BELIEVE
        KC OF X SUCHTHAT
            LOC OF MARY IS X
```

In this notational variant, Y is to be regarded as a variable within the space (in this case, SB) enclosing the one in which KC is found.

Now that we have represented "someone knows who Mary's spouse is", the behavior of these new objects can be described.

6.5.3 Matching Objects

The meaning of some object in the data base is dependent upon the operations that one can perform on it. For k.constants, the operations that are of crucial interest are the matching that occurs as part of the testing operation for classes. Testing routines, in performing deductions, can operate on a conjecture and produce a new one for which a match may be attempted. (This is what PLANNER ([Hewitt 1972]) does for consequent theorems.)

The standard matcher is recursively defined. Given a conjecture C and a proposition P, the matcher determines whether the predicate of P is of the same type as C (e.g. both are of type SPOUSE). If so, the corresponding arguments of C and P are recursively matched. When a variable matches, a binding is created so that future references to that variable must match the binding.

The matching operation can return a TRUE, FALSE, UNK, or (introduced for k.constants) a KNOWN truth-value. (abbreviated T, F, U, and K. The purpose of the KNOWN truth-value is to be able to say that the truth-value was not UNK.) Where appropriate, I will also refer to the corresponding outcome propositions. For the KNOWN truth-value KNOWN outcome propositions, which include SUPPORTs, will be returned.) Figure 6.5e defines the new MATCH function for objects of various SORTs.

Object from: CONJECTURE	PROPOSITION	
SORT:		Result:
const	const	T, if same object else F.
const	var.	U
const	k.const	K, if both are of same type, else F
var.	const	T, if both are of same type, else F.
var.	var.	T, if both are of same type, else F.
var.	k.const	T, if both are of same type, else F.
k.const	const	T, if both are of same type, else F.
k.const	var.	U
k.const	k.const	K, if both are of same type, else F.

FIGURE 6.5e
MATCH TABLE

If the variable in the conjecture has a binding, then matching reduces to seeing if the objects are EQ (and establishes a new binding if so). The only interesting case is matching a constant in the conjecture against a k.constant in a proposition. For instance, let SBUB contain the proposition representing the user's knowing who John's spouse is. Testing the conjecture USER BELIEVE(SPOUSE OF JOHN IS MARY), we match MARY against a k.constant. Intuitively, the match could be either a T or F (either John's wife is Mary or its someone else) and OSCAR doesn't know which, but it believes the user does! Hence, the match

is termed "known", or K. Clearly, the only way to obtain a K during matching is to be testing in an embedded belief space. Of course, if OSCAR were testing a non-functional relationship like CHILDREN, then its simply believing that John knows who one of Mary's children is does not entitle it to say anything about whether Fred is one of them.

Notice that k.constants behave in exactly the same way as constants with the exception that the successful matching of constants requires identity of physical locations in memory. I should emphasize that this matching operation does not say that a variable or k.constant is some constant. On the contrary, matching only determines potential identifications of k.constants and variables. More definite identifications of objects may require matching other propositions. A higher-level process, e.g. a testing routine, would have to determine which propositions (i.e. conjectures) to match in some arbitrary description.

Figure 6.5f contains the truth-tables for AND, OR, and NOT; these define how the values of submatches are combined.

AND	T	F	K	U
T	T	F	K	U
F	F	F	F	F
K	K	F	K	U
U	U	F	U	U

OR	T	F	K	U
T	T	T	T	T
F	T	F	K	U
K	T	K	K	U
U	T	U	U	U

NOT	
T	F
F	T
K	K
U	U

FIGURE 6.5f

TRUTH TABLES USING THE KNOWN TRUTH-VALUE

The truth-values do not form a Boolean algebra, but the associative, commutative, distributive and De Morgan's laws hold.

6.5.4 Testing for Known Constants

Before attempting to plan questions again we need to devise a way of enabling the precondition for CONVINCE to

fail while that of INFORM succeeds. What is needed, then, is a means for determining if an object is a variable, constant, or k.constant in some space.

This is done by creating a testing program for KC that examines the SORT propositions in the current world space. In some world space W, given some proposition P, testing KC OF X SUCHTHAT P will:

- Evaluate the "to test" of P. (This may cause a different world space to be used for the duration of the embedded testing.)
- If the test was a SUCCESS or a KNOWN, then the object in the SUPPORT that is bound to X should be a constant or k.constant in the current world space W. If so, then return a SUCCESS proposition employing as SUPPORT, the SUPPORT of the test. (This will change a KNOWN outcome proposition to a SUCCESS.) If we do not find an appropriate constant or k.constant, return an UNKNOWN outcome.
- Else, return the FAILURE or UNKNOWN outcome.

As an example, consider Figure 6.5d. The proposition stating that the user knows who Mary's spouse is, when examined in space SBUB, causes KC to succeed since the object bound to X is connected via a SORT proposition in SBUB to the class KNOWN.CONSTANT. However, if KC examines that proposition in world space SB, that same object is now of SORT VARIABLE. Hence, KC returns UNKNOWN.

6.6 Redefining CONVINCE and INFORM

Preconditions have been essentially regarded as patterns that describe the form that appropriate propositions must take. In order for proposition schemas to match propositions that quantify into belief contexts, we will create a "pattern element", QUANT, that will match the KC quantifier. "QUANT OF Y SUCH THAT P" will match a KC quantifier, its bound variable, and the quantified proposition. The purpose of QUANT is to aid in the shifting of spaces of the KC quantifier during the planning of questions. QUANT will be ignored by the planning algorithm when it fails to match a KC quantifier in the conjecture. For instance, QUANT OF X SUCH THAT P will match LOC OF JOHN IS INROOM despite the fact that QUANT has no KC quantifer to match. (P will be bound to the LOC proposition while QUANT and X will not have bindings.) This special case will enable the planning of Chapter 5 to be transparent to this addition to the precondition and effect machinery. (Another special case where QUANT must be ignored is in determining how to achieve a goal.)

We can change our earlier definitions of CONVINCE and INFORM to employ our new pattern QUANT. All that we need do is to ensure that KC is checked in the appropriate world space. CONVINCE is now defined as:

```
define convince as event
CASEFRAME:     (agt := a person) convince (recip := a person) of
               (obj := a proposition)
LOCALS:  y
EFFECTS: recip believe
               quant of y such that
                         obj
BODY:    std
CANDO.PR:      recip believe
                    quant of y such that
                         agt believe obj
.
.
.              [the rest is as in Figure 5.6b]
```

FIGURE 6.6a
A NEW DEFINITION OF CONVINCE USING QUANT

This definition says that RECIP can be convinced of the value of expression OBJ if he knows the value that AGT thinks OBJ has. Y has been stated as a variable local to the definition of CONVINCE. The manipulations performed on QUANT ensure that if OBJ is not a KC-quantified expression, planning will ignore QUANT. When QUANT matches in the EFFECT, KC will be checked in the RECIP's belief space. Compare this with our new definition of INFORM:

```
define inform as event
CASEFRAME:    (agt := a person) inform (recip := a person)
              of (obj := a proposition)
LOCALS:  y
EFFECTS: recip believe
           quant of y such that
              agt believe obj
BODY:    std
WANT.PR: std
CANDO.PR:     agt believe
                   quant of y such that
                        obj
.
.
.             [the rest is as in Figure 5.6a]
```

FIGURE 6.7b
NEW DEFINITION OF INFORM USING QUANT

For a question, the AGT of an INFORM will not be the RECIP of the CONVINCE and thus KC will be tested in different world spaces. INFORMing someone of the value of some expression now requires that the AGT know what the value is. The EFFECT of INFORM is that the RECIP (the hearer) then knows what the AGT believes the value to be. To see this in action, we will now plan a question.

6.7 Planning a Wh-Question

We will commence with WH-questions, the plans for which involve wanting to know the value of some expression. The goal will be to know where Mary is, i.e.,

```
SYSTEM WANT
  SYSTEM BELIEVE
    KC OF X SUCHTHAT
         LOC OF MARY IS X
```

where "X" is a k.constant. Of course, it is a variable in space SBSW. We will also assume that OSCAR believes the

user knows where Mary is. The belief model is represented in Figure 6.7a.

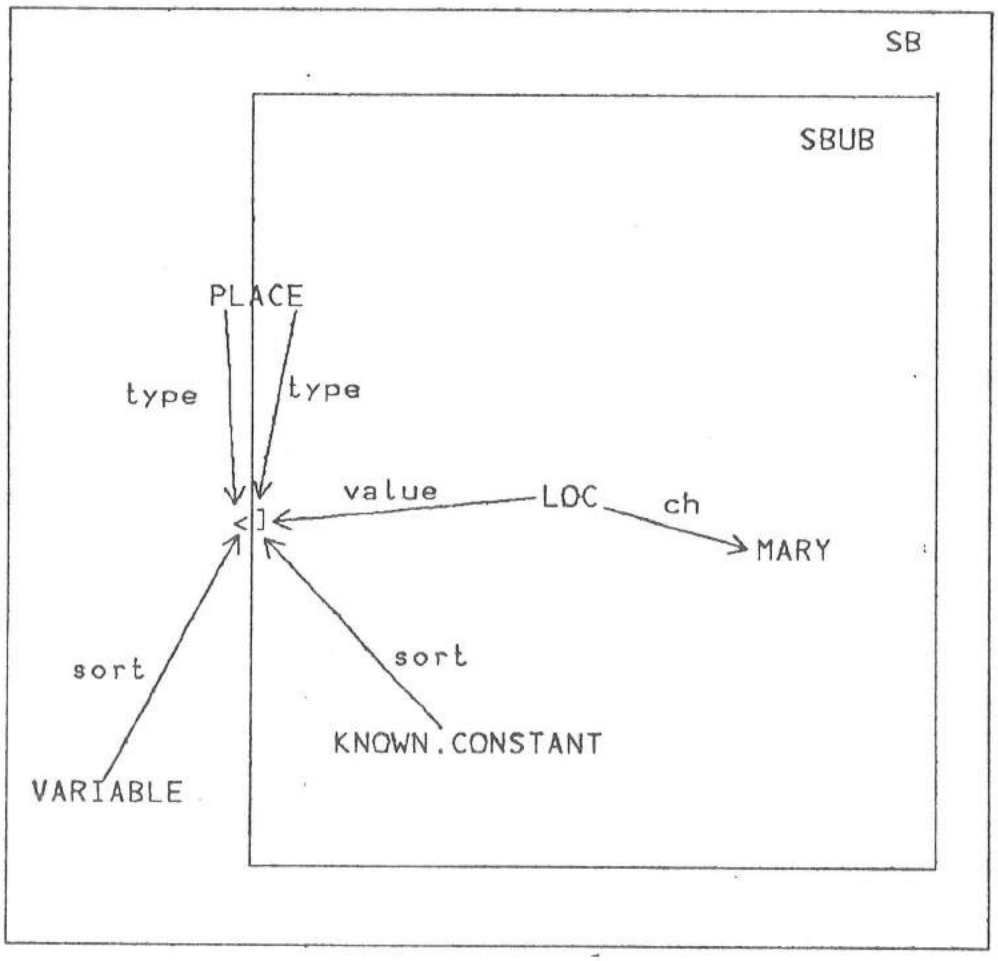

FIGURE 6.7a
INITIAL BELIEF MODEL FOR QUESTIONS

The k.constant in space SBUB straddles a space boundary indicating that it is a variable in SB. As usual, the SYSTEM BELIEVE goal is achievable via a CONVINCE (see Figure 6.7b).

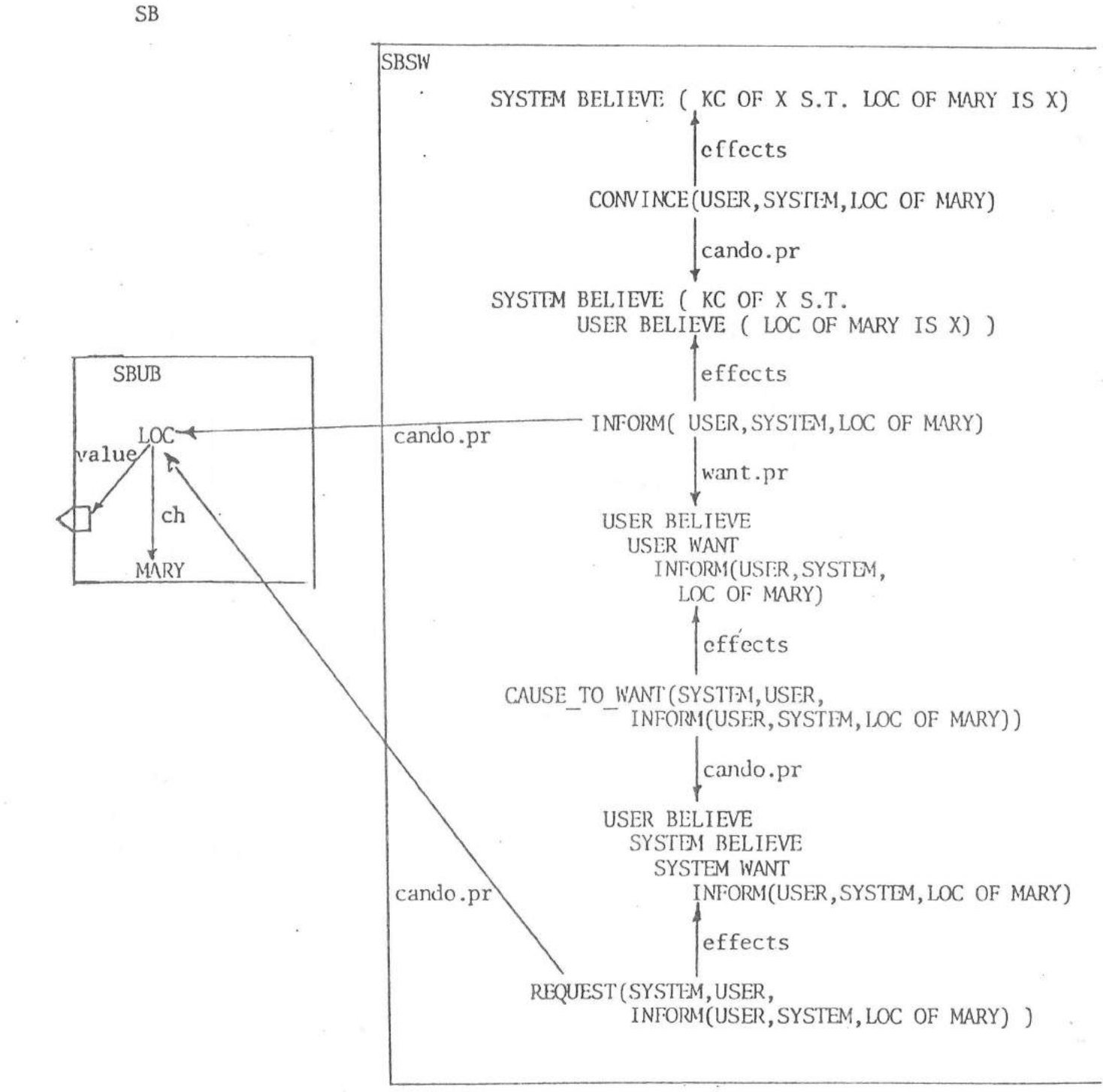

FIGURE 6.7b

THE PLAN FOR A WH-QUESTION

Assuming the user is chosen as the person whose beliefs are to be convincing, the precondition to CONVINCE that needs to be tested is:

```
SYSTEM BELIEVE
    KC OF X SUCHTHAT
        USER BELIEVE
            LOC OF MARY IS X
```

(Note that this is equivalent to a quantification through two levels of belief: ∃x SYSTEM BELIEVE USER BELIEVE (LOC OF MARY IS x).)

As in the earlier example, the test of KC in space SB fails since Mary's location is unknown in SB. At this point, the QUANT pattern in the EFFECT of INFORM matches a KC in the goal. Hence, when the precondition to INFORM is created, the KC is moved to an inner space. OSCAR must now test the CANDO.PR of this INFORM which is:

```
SYSTEM BELIEVE
    USER BELIEVE
        KC OF X SUCHTHAT
            LOC OF MARY IS X
```

Of course, this is OSCAR's representation of its believing that the user knows the value. The interpreter again starts in SB, shifts to SBUB and tests KC, which eventually succeeds since Mary's location is known in this space.

The planning of the question is not finished, however, since we still need to get the user to want to tell us where Mary is. This is the usual WANT.PR on INFORM and hence will eventually lead to a REQUEST. OSCAR thus plans REQUEST(SYSTEM,USER,INFORM(USER,SYSTEM,LOC OF MARY)).
All the preconditions on REQUEST are true since OSCAR is again testing those of INFORM. (Notice that one of the preconditions is of the form UB∃xUB(P(x)). By our space structuring, this reduces to ∃xUB(P(x)).) The plan has

progressed from wanting to know the value of an expression, to wanting to know the value that the user thinks the expression has, to finally just wanting to know that the user knows the value (which is true). The resulting question, for which the final plan is Figure 6.7b, would be "Where is Mary?"

6.7.1 Getting a Key

OSCAR's Wh-question of Chapter 1 can now be seen to arise naturally, provided one has an appropriately defined precondition for GET, the act of obtaining something. If we add a precondition to UNLOCKDOOR that AGT HAVE KEY1, OSCAR can produce a question by using the following definition of GET:

```
define get as event
CASEFRAME:    (agt := a person) get (obj := a physical-object)
LOCALS:  x
CANDO.PR:     agt believe
                kc of x suchthat
                   loc of obj is x
WANT.PR: std
EFFECTS: agt have obj
TO.TEST: std
TO.ADD:  std
```

.
.
.

This definition says that before someone can get something, he must know where it is. We can use KC directly, instead of QUANT, since the EFFECT does not state a quantified belief. This generates the appropriate goal for a WH-question. The plan is of the same form as that found in Figure 6.7b.

A virtually identical process, operating from a different goal, leads to the creation of a Yes/No question.

However, we need to define a new function to use as the content of these questions.

6.8 The T-V Function

Let us define a set of truth-values in SB, TRUTH-VALUE, with two elements TRUE and FALSE. We can now create a relation T_V, instances of which will not be explicitly represented in belief spaces, from PROPOSITION to TRUTH-VALUE.[3]

```
t_v is a new instance of relation
CASEFRAME:  t_v of (ch := a proposition) is
                   (value := a truth-value)
TO.TEST: t_v.test of instance
TO.ADD:  t_v.add of instance
TO.RETRIEVE:  t_v.retrieve of instance
```

When the value of T_V is existentially quantified across a belief (as in USER BELIEVE (KC OF X SUCHTHAT (T_V OF P IS X))), then the VALUE of T_V (X above) is a k.constant, called KNOWN or K. To specify the program's semantics for T_V, I will describe the testing, adding, and retrieving programs.

6.8.1 Testing T_V

T_V.TEST operates on a proposition, bound to the parameter INSTANCE, of the form T_V OF CH IS VALUE. It commences by invoking the testing program of the proposition bound to CH. This program returns an outcome proposition of the usual kinds (now including a KNOWN outcome.) The truth-value, TVAL, to be associated with the test of CH is one of T, F, U (a variable), or K (a k.constant) corresponding to the outcome proposition.

Establishing TVAL is complicated by the fact that while a test of CH may yield UNK, there is another way to establish that the truth-value is K. Recall that the motivation behind the KNOWN truth-value is the need to

represent that the truth-value of some conjecture is not UNK. We can represent this state of affairs whereby proposition A BELIEVE P is KNOWN by the proposition OR(A BELIEVE P, A BELIEVE NOT(P)). Person A thus is believed to know what the truth-value of P is, though the system does not.

The algorithm for T_V.TEST, which tests for such disjunctions, then is (for propositions of the form T_V OF CH IS VALUE in world space W):

```
TV <- TEST OF CH in world space W
If TV
Then TVAL <- TRUE
Else TVAL <- FALSE
UNK  Find the BELIEVE proposition identifying world space W.
     Fetch the AGT of this proposition.
     Find the belief space B containing W.
     If B ≠ W   /* The system cannot believe it knows what the
                   truth-value of CH is without in fact
                   believing CH is true or false.  */
     Then If TEST OF OR( AGT BELIEVE CH,
                         AGT BELIEVE NOT(CH) ) in B
          Then TVAL <- KNOWN
          Else TVAL <- UNK
          Unk  TVAL <- UNK
          Known TVAL <- KNOWN
     Else TVAL <- UNK
     Unk;
     Known;
Known  TVAL <- KNOWN

/* Notice that we are now using a four-way conditional. */

Return an outcome proposition corresponding to
MATCH(TVAL,VALUE), with supports and obstacles taken
from TV.
```

The routine matches TVAL, the established truth-value of the proposition bound to CH, against VALUE, the conjectured truth-value, that is found in the world space. The result of this match, as usual, is T, F, U, or K, since we may be comparing constants (T or F) with variables (U) or to k.constants (K).[4] T_V.TEST will construct and return an outcome proposition corresponding to the result of this match.

As an example, consider testing:

```
T_V OF
    USER BELIEVE
         SPOUSE OF MARY IS JOHN
IS TRUE
```

If we only believe that the user knows who Mary's spouse is, then the result of this testing would be K since that is the truth-value of the embedded proposition.

6.8.2 Retrieving T-V

T_V.RETRIEVE performs all the functions of the above T_V.TEST except the matching of TVAL and VALUE. (We are not given a VALUE for a retrieval.) This function thus returns the truth-value corresponding to the outcome proposition that resulted from testing CH. Clearly, T_V.TEST could have been defined in terms of T_V.RETRIEVE (and is implemented in that way).

6.8.3 Adding T-V

T_V.ADD is the program that asserts propositions with appropriate truth-values into the data base. Since truth-values are not explicitly represented in belief spaces, T_V.ADD must manipulate the world space before evaluating the "to add" of the proposition bound to CH. (CH and VALUE will refer to their bindings.) The algorithm is:

In world space W,

- If VALUE is T
  then ADD CH into W.

- Else if VALUE is F
  then find NOT space N in W
  ADD CH into N.

- Else if VALUE is U
  then DELETE CH in W
  if the deletion did not succeed
  then find NOT space N in W
  DELETE CH in N.

- Else if VALUE is K
  Then: Find the BELIEVE proposition identifying W.
  Fetch the AGT of this proposition.
  /* The above employs the "back-linking" */
  /* facilities of the network data structure. */
  Find the belief space B containing W.
  If B ≠ W
  Then: ADD((AGT BELIEVE CH)
  OR
  (AGT BELIEVE NOT(CH))
  into B.
  Else; /* W is the top space -- hence no addition
  is to be made. Either the system knows
  the truth-value or it doesn't. */

The last two cases of this algorithm require some explanation. When asserting a truth-value of UNK, all that we are seemingly entitled to do is to erase CH from that space. However, CH may be believed to be false and hence not be in the world space, but in an embedded NOT space. If so, CH must be deleted from that space.

The final case is where OSCAR is asserting, in the user's belief space, that the truth-value is known, i.e.,

ADD (USER BELIEVE (KC OF X SUCHTHAT (T_V OF P IS X))). Again, this can be represented using the explicit disjunction USER BELIEVE CH OR USER BELIEVE NOT(CH). However, in order to assert the disjunction of BELIEVE propositions for some agent A, OSCAR must have access to the belief space containing A's belief space. The representation resulting from ADDing this proposition would be exactly the same as Figure 4.4b.

6.9 Yes/No Questions

What these are, at least informally, should be obvious. One may ask such questions literally, simply wanting to know just "yes" or "no". On the other hand, one may ask "Is Mary at home" when what one really wants to know is where she is. This latter non-literal question is not currently plannable since I do not know what initial goal would lead to such an utterance. However, since polar questions can be meant and/or interpreted literally, the literal question ought to be directly deriveable from initial goals.

I will simply be concerned with producing Y/N questions when planning starts from an appropriate goal -- knowing the truth-value of some proposition. The kinds of processes that could give rise to such goals will be ignored for the time being. If the speaker could evaluate the truth of the particular proposition with respect to his model of the hearer's knowledge, then the question would be answered. What the speaker needs to discover is if the hearer knows whether the proposition is true or false. If he thought the hearer did not know the truth-value (i.e. the answer UNK results), he would have no grounds upon which to believe the hearer could answer the question.

In wanting to know whether Mary is in the room, OSCAR has the goal:

```
SYSTEM WANT
    SYSTEM BELIEVE
        KC OF X SUCHTHAT
            T_V OF
              LOC OF MARY IS INROOM
            IS X
```

This time, the k.constant in the goal is the truth-value of the proposition LOC OF MARY IS INROOM. It will be assumed, as with the previous WH-question, that OSCAR believes the user knows where Mary is.

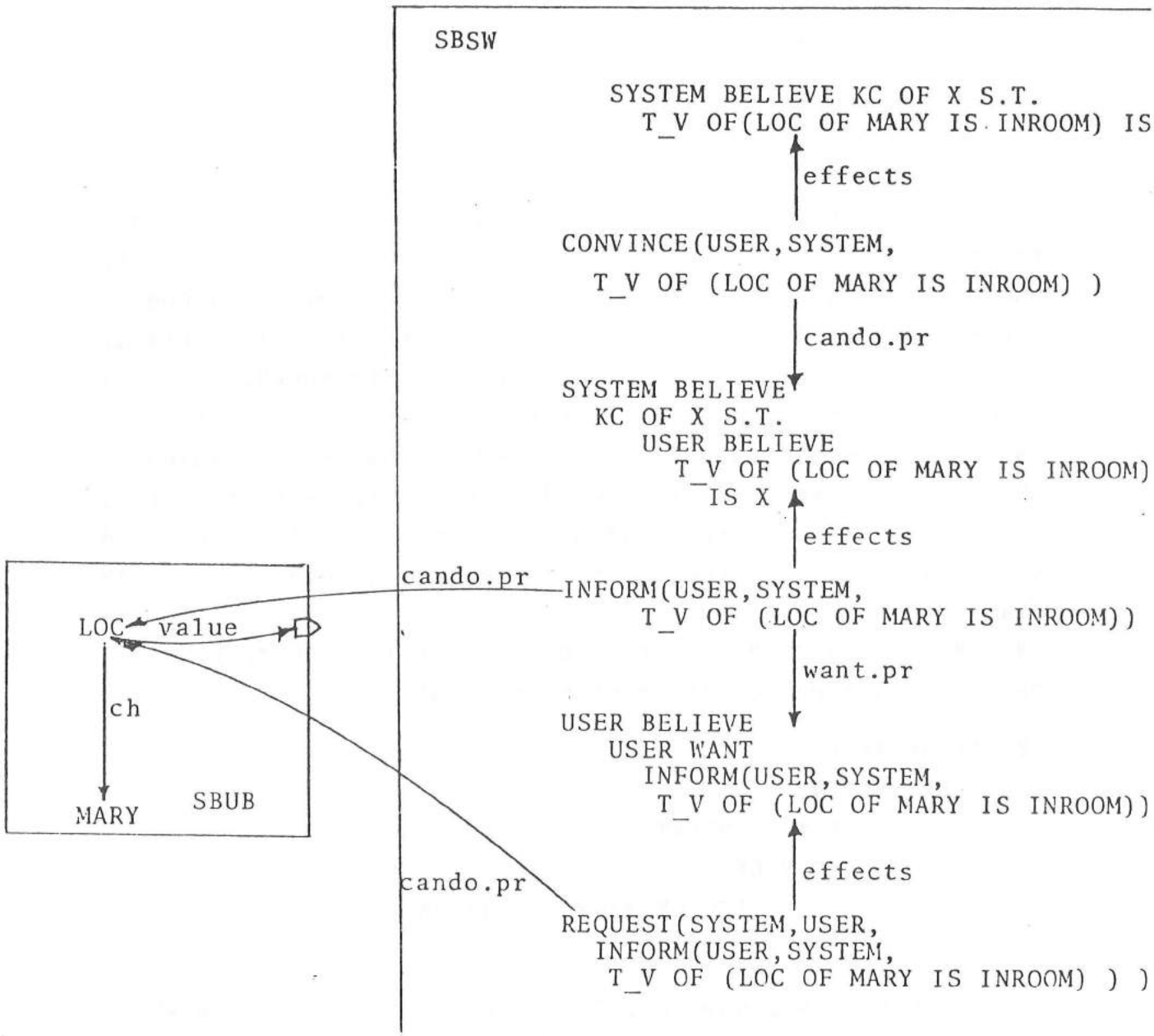

FIGURE 6.9a

PLAN FOR A YES/NO QUESTION

The planning of this question (in Figure 6.9a) is of the same form as before. The precondition to CONVINCE that is tested is:

```
SYSTEM BELIEVE
    KC OF X SUCHTHAT
         USER BELIEVE
              T_V OF
                   LOC OF MARY IS INROOM
              IS X
```

The interesting part again occurs when matching LOC OF MARY IS INROOM against the proposition in SBUB stating that the user knows where Mary is. This requires matching INROOM, a PLACE, against a k.constant also taken from the class PLACE. As Figure 6.5e shows, this sub-match is KNOWN. The T_V proposition then, after combining the results of the submatches with KNOWN, has a KNOWN SUPPORT -- the value of the T_V conjecture is discovered to be a k.constant. But, of course, since the binding of X, when examined in world SB, is a variable, the test fails. Consequently, OSCAR plans:
INFORM(USER, SYSTEM, T_V OF (LOC OF MARY IS INROOM)).
The precondition to INFORM then becomes:

```
SYSTEM BELIEVE
    USER BELIEVE
         KC OF X SUCHTHAT
              T_V OF
                   LOC OF MARY IS INROOM
              IS X
```

In this case, however, KC is tested in space SBUB where X is bound to a K truth-value (a k.constant). Thus, the test of KC is a SUCCESS, as is the rest of the proposition. The rest of the plan (to REQUEST this INFORM) proceeds as usual. The constructed question then is:

```
REQUEST (SYSTEM, USER,
    INFORM (USER, SYSTEM,
        T_V OF (LOC OF MARY IS INROOM) ) )
```

which could be realized as "Is Mary in the room?"

The system could know less about the user's beliefs than that he knows where Mary is, namely that he knows whether she is in the room or not. If this information, encoded as the disjunction OR(USER BELIEVE (LOC OF MARY IS INROOM), USER BELIEVE (NOT(LOC OF MARY IS INROOM))), is present in SB, then the yes/no question can also be planned.

6.9.1 A "Do You Know Whether" Question

The form of the Yes/No question plan above is used in the planning of the literal versions of "Do you know whether P?" or "Do you believe that P?" questions. As in the simple polar question, the only part of this plan that differs significantly from the WH-question is the propositional content, which in this case is:

```
KC OF X SUCHTHAT
  T_V OF
    USER BELIEVE P
  IS X
```

For the example dealing with Mary's location, the initial goal, leading to "Do you know if Mary is in the room?" is:

```
SYSTEM WANT
    SYSTEM BELIEVE
        KC OF X SUCHTHAT
            T_V OF
              USER BELIEVE
                 LOC OF MARY IS INROOM
            IS X
```

Unlike other goals, this one has a BELIEVE proposition dominated by a non-belief. This makes it impossible to directly perform an INFORM (as would be possible if the T_V

proposition were not present) since the matching of the effects of INFORM against the goal will fail. As usual, OSCAR can achieve this goal via CONVINCE, with the T_V proposition as what is to be believed. After the testing of the CANDO.PR of CONVINCE, OSCAR will eventually plan an INFORM(USER, SYSTEM, T_V OF (USER BELIEVE (LOC OF MARY IS INROOM))), whose precondition is:

```
SYSTEM BELIEVE
    USER BELIEVE
        KC OF X SUCHTHAT
            T_V OF
              USER BELIEVE
                 LOC OF MARY IS INROOM
            IS X
```

The only unusual aspect of this goal is the USER BELIEVE...USER BELIEVE. Clearly, based upon the memory structure of Chapter 4, the testing of this proposition presents no problem since the "to test" programs simply find SBUB as the new world space twice. The reader can easily verify that the testing of KC takes place in the appropriate spaces. Hence, this question proceeds in the fashion of the others.

OSCAR could create its own T_V OF P goals when the time specified for P is the current time and when P is fully grounded (i.e., it uses no variables). After receiving an UNK answer for the testing of FASTENING OF DOOR IS LOCKED, OSCAR could create the new goal:

```
SYSTEM BELIEVE
    KC OF X SUCHTHAT
        T_V OF (FASTENING OF DOOR IS LOCKED)
    IS X
```

leading to a polar question. This capability for creating its own T_V goals has not been implemented, but easily could be.

6.10 Teacher/Student Questions

Teacher/student (T/S) questions often occur in formal learning situations. In such environments, the teacher clearly knows the answer to such questions before asking them. The point is to determine, in order to pursue some teaching strategy, what the student believes the answer to be. There are, of course, questions in a classroom environment where the teacher really wants to know the answer, (e.g., "What time is it?" spoken to a college-level class.) The "knowing that vs. what" distinction thus plays a role not just in the asking of questions, but also in the hearer's determination of the kind of question that is being asked; different interpretations will yield different kinds of answers. Explaining how students determine whether or not the teacher knows the answer is a crucial problem for any model of question-answering. Since that is not my current goal, I will content myself with showing the plans for T/S questions.

The planning of T/S questions simply involves having different initial goals. In particular, the teacher (i.e. OSCAR) would not need to become CONVINCEd of anything. The starting goal would be the CANDO.PR of CONVINCE. For instance, the teacher/student question leading to "Where does Jimmy Carter live?" would be planned from the initial goal of:

```
SYSTEM WANT
    SYSTEM BELIEVE
        KC OF X SUCHTHAT
            USER BELIEVE
                RESIDENCE OF JIMMY.CARTER IS X
```

The plan would proceed as in Figure 6.7b (the Wh-question) provided that OSCAR (the teacher) believes the user knows Carter's residence. Of course, a teacher might have very little confidence in the student's answer but he

would not ask a question (or so it appears) unless he had reason to believe the student had an answer. The plan, then, develops in the usual fashion.

6.11 Three-Party Speech Acts

Everything that has been done so far has employed only two conversants. This section is intended to show that given certain domain-constraints, and appropriate decision-making, all of the planning of speech acts is extendable to multi-party conversations. To illustrate this, I will force OSCAR to direct the user to get some other person, say Tom, to do some action. By "force", I mean that I will make appropriate choices (when OSCAR cannot) of the agents to CONVINCE and CAUSE_TO_WANT. OSCAR will then check all the embedded belief models as needed.

6.11.1 An Example

Setting up the situation simply requires the addition of appropriate information for USER and TOM, in belief spaces within SB (i.e. SBUB and SBTB). Within each of these, we would find belief spaces for the other two participants. We simply state at the outset that OSCAR believes that:

- Tom knows where Mary is
- the user believes that Tom knows where Mary is.

Notice that OSCAR does not have a belief that the user knows Mary's location.

As usual, MARY, TOM, USER, and SYSTEM are constants in all belief spaces. The system's goal will be:

```
SYSTEM WANT
    SYSTEM BELIEVE
        KC OF X SUCHTHAT
            LOC OF MARY IS X
```

The situation will be externally constrained so that OSCAR cannot talk directly with Tom (e.g., it doesn't know his phone number); communication must go through the user. Tom, however, will be allowed to talk directly with OSCAR. We

shall assume that Tom is chosen as the person whose beliefs will be convincing.

The plan for a three-party speech act is shown in Figure 6.11a. From here on, I will abbreviate liberally:

SYSTEM ------> S
USER---------> U
TOM ---------> T
BELIEVE------> B
WANT --------> W
LOC OF MARY IS X -------> P

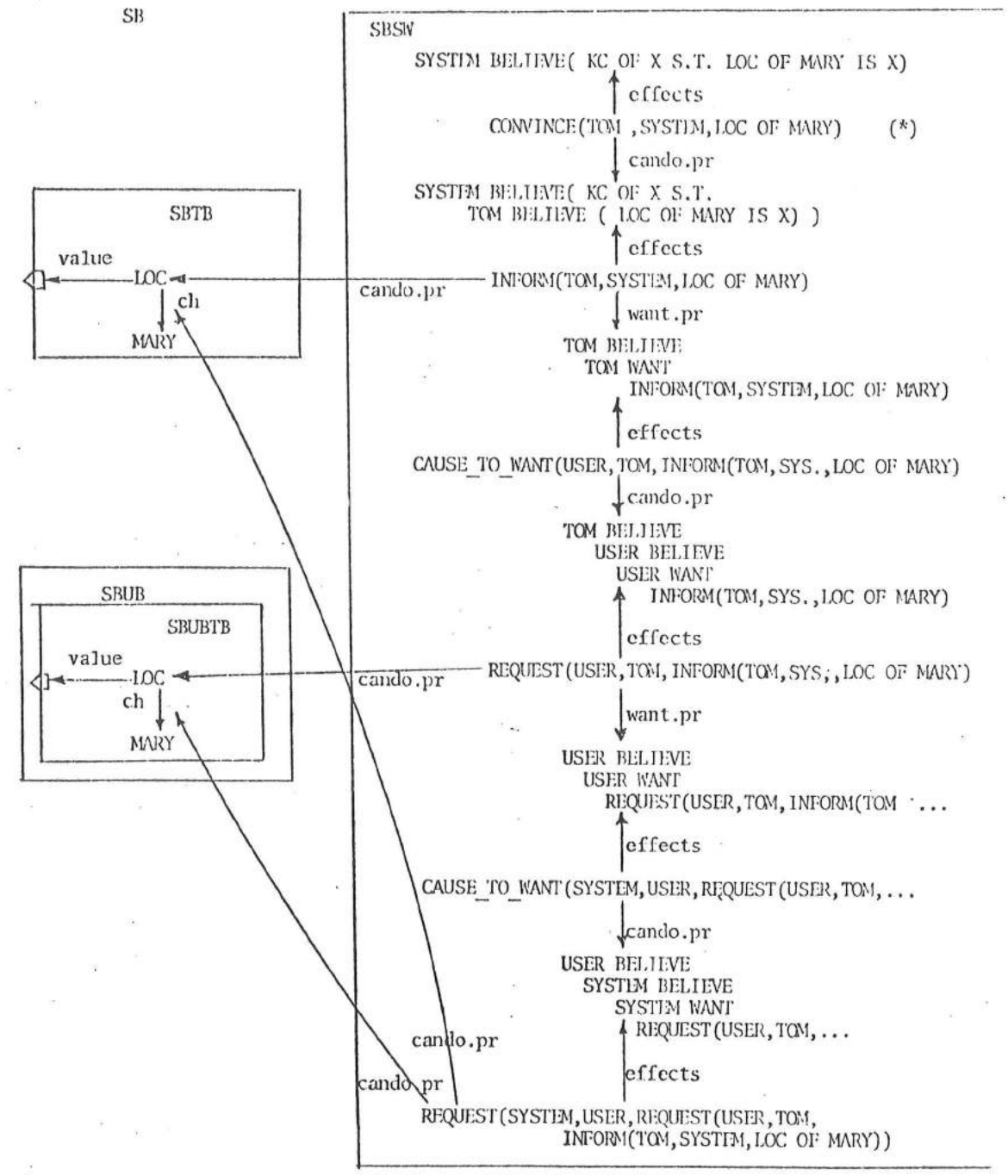

FIGURE 6.11a

PLAN FOR "ASK TOM TO TELL ME WHERE MARY IS."

The CANDO.PR of the first CONVINCE (labelled "*") is: SB (KC OF X SUCHTHAT TB(P))) and is UNKNOWN since OSCAR does not know where Tom thinks Mary is. OSCAR then plans INFORM(T,S,P) and checks its CANDO.PR, (SB (TB (KC OF X SUCHTHAT P))), which succeeds since OSCAR believes Tom knows where Mary is. As usual, planning of the WANT.PR to INFORM leads to a CAUSE_TO_WANT. However, here, since OSCAR cannot communicate directly with Tom, we choose USER as the one whose goals will cause Tom to want to INFORM. Hence, the new goal is

```
TOM BELIEVE
    USER BELIEVE
        USER WANT
            INFORM(TOM, SYSTEM, P)
```

This goal is achievable via a REQUEST(U,T,INFORM(T,S,P)) provided that U believes T CANDO the INFORM, i.e. that U believes T knows where Mary is. Since this was included in the initial configuration of beliefs, this CANDO.PR is true. The WANT.PR of this REQUEST, though, is not yet believed to be true -- the user has to want to REQUEST. This time, we choose SYSTEM as the one whose goals are influential allowing OSCAR to plan a

```
REQUEST(SYSTEM, USER,
    REQUEST(USER, TOM,
        INFORM(TOM, SYSTEM, P)))
```

The CANDO.PR of this REQUEST checks the CANDO.PR of the embedded REQUEST which, in turn, references the CANDO.PR of the INFORM. OSCAR is thus testing:

```
SYSTEM BELIEVE
    USER BELIEVE
        TOM BELIEVE
            KC OF X SUCHTHAT
                LOC OF MARY IS X
```

which is found to be true. The WANT.PR of the system's REQUEST is trivially true so the plan is complete. Its

surface realization might be: "Ask Tom to tell me where Mary is."

6.11.2 Variations of the Example

If, in planning the first CONVINCE of Figure 6.11a, the user were chosen as the "convincer", then OSCAR would plan for the user to tell the system Mary's location. But, in the starting state, the user is not believed to know where Mary is. Consequently, OSCAR needs to achieve the CANDO.PR of this INFORM (in addition to the WANT.PR). OSCAR's planning for the user's knowing the answer involves requesting the user to ask Tom a question! This replicates the form of the question plan of Figure 6.11a achieving the CANDO.PR of the INFORM. The additional REQUEST on the WANT.PR of the first INFORM then asks the user to tell the system Mary's location. This version of the three-party speech act might then be realized by: "Ask Tom to tell you Mary's location and then tell me." The plan developed for this three-party speech act is shown in Figure 6.11b.

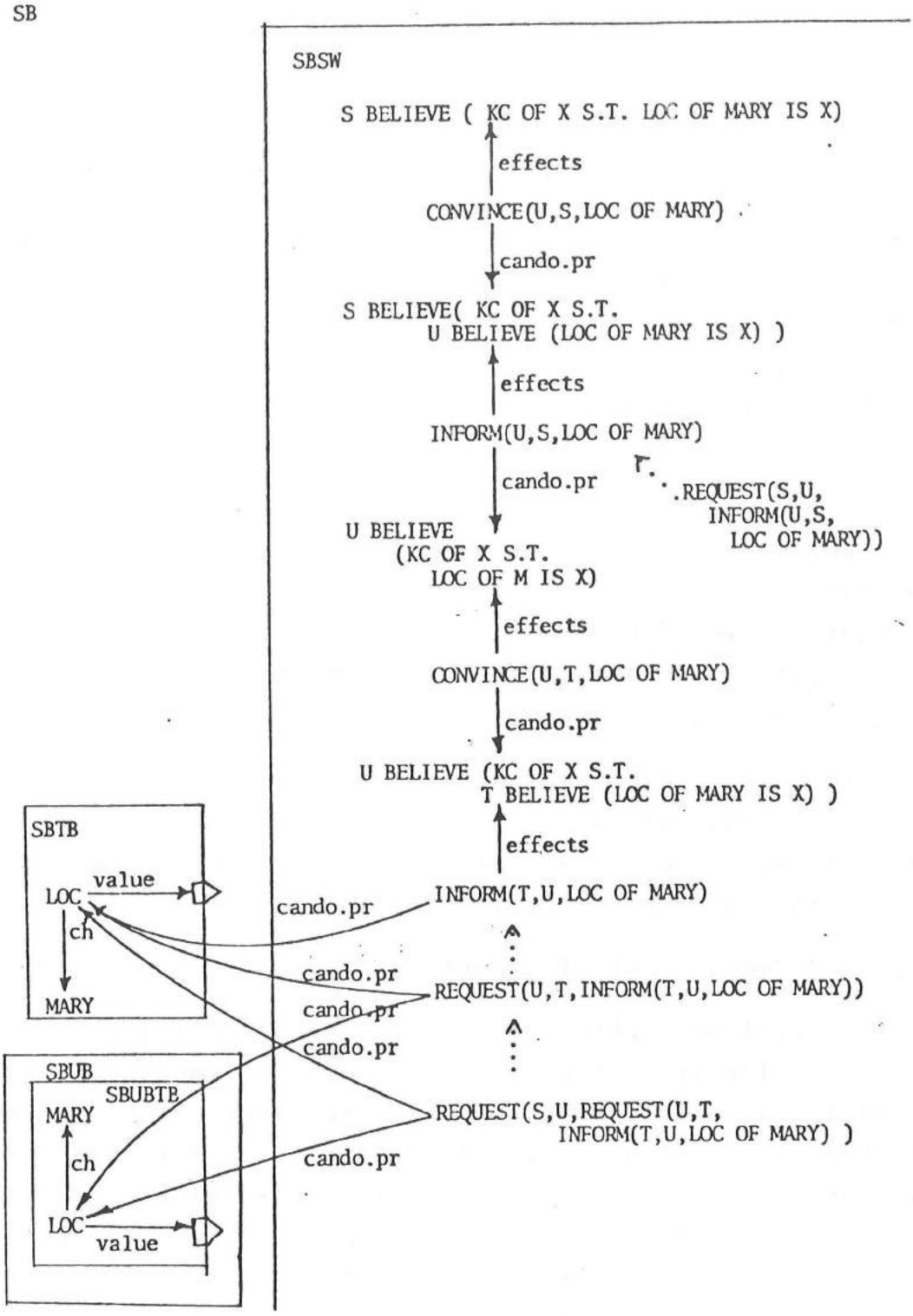

FIGURE 6.11b

PLAN FOR "ASK TOM TO TELL YOU MARY'S LOCATION AND THEN TELL ME."

It should be clear that the means for constructing three-party speech acts is appropriate for "polar propositional content" as well since plans for the polar type are of exactly the same form as for the WH-question. The same can be said for three-party teacher/student questions. In a situation where OSCAR is teaching a student-teacher (the user) how to ask questions (or prepare lessons) in a classroom setting, OSCAR can plan for the user to ask a T/S question. The initial goal would be:

```
SYSTEM BELIEVE
    USER BELIEVE
        KC OF X SUCHTHAT
            STUDENT BELIEVE
                RESIDENCE OF CARTER IS X
```

The speech act that OSCAR would plan is:

```
REQUEST(SYSTEM, USER
    REQUEST(USER, STUDENT
        INFORM(STUDENT, USER, RESIDENCE OF CARTER ) ) )
```

This is simply part of the plan of Figure 6.11b.

There is nothing peculiar about a three person conversation that allows the above planning to work. All we need is to have the appropriate knowledge of who knows what, and make the appropriate choices of agents.

6.11.3 A Recognition of Intention Problem

One level of sophistication that we are not yet able to capture is planning a three-party speech act where the user is simply a middle-man who need not know anything about Tom's beliefs. The user should be able to infer that Tom knows where Mary is by knowing that OSCAR thinks so. Such an inference would be part of the effects of a request -- namely that the hearer should believe that the speaker believes the preconditions of the request are true. However, OSCAR would have to plan for the user to recognize OSCAR's utterance as a REQUEST (and not, say, as a SUGGEST) in order

that he believe the side-effects of the REQUEST were true. Hence, OSCAR would be intending that the user make an inference by recognizing OSCAR's intentions.

OSCAR is currently not sophisticated enough to do such planning. In particular, its simple backward-chaining planning algorithm would have checked the precondition (and perhaps created a subgoal) for the user's beliefs before planning its own request that would have obviated the extra work. I suspect that a solution to this problem will require better planning algorithms incorporating criticism (see [Sacerdoti 1975]), and multiple models of the world within the want space, in addition to an understanding of how we can intend for someone to recognize our intentions.

6.12 Stating a "Knows That" Proposition

However, there is another way of producing the three-party speech act when the user is not believed to think that Tom knows where Mary is -- OSCAR could tell him! OSCAR would then plan "Tom knows where Mary is. Ask him to tell you where she is and then tell me."

The goal, precondition (+) in Figure 6.11a, that would lead to informing the user of Tom's knowledge, is:

```
SYSTEM BELIEVE
    USER BELIEVE
        TOM BELIEVE
            KC OF X SUCHTHAT
                LOC OF MARY IS X
```

This is achieved via the standard CONVINCE-INFORM route with TOM BELIEVE KC OF X SUCHTHAT LOC OF MARY IS X as propositional content. This works provided OSCAR believes Tom knows where Mary is, which we are assuming. There is no ambiguity with the QUANT pattern in the definitions of INFORM and CONVINCE since they are not looking for KC quantifiers in the propositional content.

6.13 Existential Questions

Questions like "Does Mary have a husband?" or "Did anyone win the World Series in 1943?" are termed existential questions since they question whether there exists some object satisfying a certain description (i.e., they question whether $\exists xP(x)$ is true, for appropriate P).

Such questions would be planned from initial goals like:

```
SYSTEM BELIEVE
    KC OF X SUCHTHAT
        T_V OF
            THEREXISTS A Y SUCHTHAT
                SPOUSE OF MARY IS Y
        IS X
```

The only way a question can be planned here is if OSCAR believes that either the user believes Mary is married or the user believes Mary is single. Such disjunctions are detected by the testing routine for T_V. Notice that if OSCAR believed the user thought Mary were married (i.e., SBUB contained a proposition of the form $\exists xP(x)$), then it would not have to ask a question.

6.14 Summary

I have presented a means for representing the information that someone knows what the value of an expression is; information that is vital to the planning of questions. OSCAR is thus able to reason with incomplete knowledge of someone else's beliefs. Such knowledge has been lacking from earlier AI natural language methodologies. In particular, Schank [1975], Meehan [1977], and Mann et al. [1977] all employ acts called ASK whose preconditions ignore this kind of information. It is also somewhat surprising that philosophers, when discussing the conditions on speech acts, have not recognized the issue of quantified beliefs that arises for inform speech acts, and hence for questions. Part of the reason for this omission, I conjecture, is that

they define speech acts without considering how those acts will be used.

The methodology was exercised in the planning of numerous types of questions, and was extended to multi-party speech acts. With these capabilities, some of OSCAR's invocations of its oracle for help in determining bindings can be replaced by its own planning to discover that information. It can easily create goals that lead to questions (for instance knowing the initial location of some object that it is planning to move). The goals with variables in them, as created in Chapter 5, would then become quantified beliefs, leading to Wh-questions. The "what's next" scheduling strategy, however, would want to pursue this sub-plan first in order to obtain the information necessary to enable it to continue with its main plan. In this way, OSCAR could mix questions, statements, and non-linguistic actions into its planning.

6.15 Notes

1. Since OSCAR can simulate someone else's planning, what is prefixed to each goal is PLANNER BELIEVE, where PLANNER is replaced by SYSTEM, when planning in SBSW, and by USER, when planning in SBUBUW, etc.

2. This goal, intended to lead to a Yes/No question, is not well-formed. Matters have been simplified here in order to make the more global difficulties clear. Subsequent sections will reformulate the propositional content above.

3. The proposition whose truth-value is in question (CH) and its discovered truth-value VALUE are contained in belief spaces even though the T_V propositions themselves are not. This will be relevant to the testing of quantified T_V propositions.

4. All K values are identical and thus matching two of them yields T.

CHAPTER 7

A FORMAL MODEL OF OSCAR AND ITS RELATION TO OPACITY

Contents:

7.0 Introduction

The purpose of this chapter is to formalize the representation and demonstrate its expressive power by examining its ability to handle certain classical problems involving propositional attitudes. The problems are not just cf historical interest but are critical to the planning of questions. What I will try to do is to present an early formulation of the problem of opacity, demonstrate a solution, and then expand this solution to other types of ambiguity discussed in [Partee 1972].

7.1 The Problem of Opacity

Quine, in his influential paper "Quantifiers and Propositional Attitudes" ([Quine 1956]) notes that there is something special about the "propositional attitudes", such as "believe", "want", "wish", etc.[1] An attempt to represent them as predicates in the standard logical way runs into difficulties. The canonical example is:

(I) "John believes the number of planets equals 6."

(II) "The number of planets equals 9."

If we treat "believe" as a predicate and follow the usual logical rules, we are thus, incorrectly, led to conclude that

(III) "John believes that 9 equals 6."

Obviously, this conclusion is absurd. Yet, if we confine ourselves to standard logical formalisms, we have great difficulty in representing (I) and (II) to prevent such inferences.

We encounter difficulties in representing "believe" as a predicate when we attempt to substitute expressions of equal value for one another such that the truth of the formula is preserved. This is a standard operation in manipulating logical formulas but when applied to formulas incorporating propositional attitudes, we derive obvious falsehoods.

Quine suggests that "believe" sets up an opaque context, one that prevents such substitutions from outside the context. In the example above, we are not permitted to substitute '9' for 'the number of planets' in (III) because the latter is in an opaque context. This rule is trying to capture our intuitive understanding that the reason why we could not draw the conclusion (III) from (I) and (II) is that John does not believe (II).

A slightly more complex example of Quine's draws a sharper distinction between opaque and transparent contexts. The example goes as follows:

> There is a certain man in a brown hat whom Ralph has glimpsed several times under questionable circumstances on which we need not elaborate here; suffice it to say that Ralph suspects he is a spy. Also there is a gray-haired man, vaguely known to Ralph as a pillar of the community, whom Ralph is not aware of having seen except once at the beach. Now Ralph does not know it, but the men are one and the same. [Quine 1956; p. 154, in Davidson and Harman]

(IV) Ralph believes the man in the brown hat is a spy.

(V) Ralph believes the man on the beach is not a spy.

(VI) The man on the beach = the man in the brown hat = Ortcutt.

At this point, we might be lead to accuse Ralph of holding contradictory beliefs since, ignoring the purported opacity of "believe", (IV) and (VI) contradict (V) and (VI) when 'Ortcutt' in (VI) is substituted for the appropriate expressions in (IV) and (V). But, of course, Ralph does not think there is a contradiction. If we regard the belief context as opaque to such substitutions, we derive no such contradiction.

Not only does the opacity of "believe" prevent the substitutions of expressions, it also, Quine claims, disallows quantification into such contexts. "Belief contexts are referentially opaque; therefore it is prima facie meaningless to quantify into them;..." (Quine [1956]; p156, in Davidson and Harman.) To see why Quine thinks such quantifications are nonsense, let us consider (IV) and (V). We would like those to be jointly true, given the setting above. By the rule of "existential generalization" one can obtain (VII) below from (IV):

(VII) $\exists x$ (Ralph Believes x is a spy)

Now, (VII) is true if and only if, under interpretation, there is some object in the model such that Ralph believes spyhood of that object. However, by the same line of reasoning, (IV) and (V) are predicating something about objects, namely Ortcutt, and thus could not both be true, in the presence of (VI), if Ralph is consistent. In other words, if we could quantify into the belief context, then we ought to be able to substitute the expression 'Ortcutt' for 'the man in the brown hat' in (IV), and similarly in (V), since the expressions have the same denotation. Quine claims that since we want (IV) and (V) to be jointly true, we must accept the restrictions of opaque contexts and thus force (VII) to be nonsense.

But, we want to be able to represent the statement "There is someone whom Ralph believes to be a spy" or "There is someone whom the eyewitness believes murdered Bill." In order to do this, Quine proposes two types of "believe" -- an opaque and a <u>transparent</u> one (that allows quantification and substitution from outside the context). Any arguments to such predicates that allow substitution, such as the arguments of the transparent "believe" predicate, are said to occupy <u>referential positions</u>.

The opaque vs. transparent distinction is evident as an ambiguity at the surface level. For example, the sentence

(VIII) "Ralph believes Ortcutt is a spy."

has two readings depending upon whose interpretation of 'Ortcutt' (the speaker's or Ralph's) is intended. paraphrased by:

1. "The person whom I call 'Ortcutt' is believed by Ralph to be a spy."

2. "The person whom Ralph calls 'Ortcutt' is believed by Ralph to be a spy."

The difference is based upon who uses the term 'Ortcutt'. The "transparent" representation of sentence (VII), given just the statements (IV) and (VI), appears to allow the speaker to import his description for Ralph's object, since the phrase 'the man in the brown hat' occupies a referential position, and the belief context is transparent to substitution of 'Ortcutt' for 'the man in the brown hat.' Given an opaque representation, the speaker can only utter the sentence when he believes that Ralph uses the term 'Ortcutt'. Quine then concludes there are two types of "believe", since sentence (VII) is ambiguous with respect to the above readings.

His solution to representing the transparent version is to adopt a new triadic predicate of the form:

agent Believe (attribute, object)

e.g., (IX) Ralph Believes ('x is a spy', Ortcutt)

which can be paraphrased as "Agent believes attribute is true of object." Quine's representation of (IX) then becomes (via existential generalization):

(X) $\exists x$ (Ralph believes('y is a spy',x))

The reason why there is no problem of opacity here is that the variable x is not quantified with respect to the context of belief (the expression). Since object is thus independent of Ralph's beliefs, substitutions can be performed.

There is no need for a new representation for "believe". Rather, we will show that a differently structured model for the formal language can produce interpretations for sentences (IV), (V) and (VI) so that all can be jointly true, and in agreement with our intuitions. With this different model structure, we need only have one representation for belief -- an opaque one. We will then

show how both readings of sentence VII can be derived from this one underlying representation. The model theory will also enable us to represent a number of surface phenomena that have been problematic for other formalisms.

7.2 Formal Model of OSCAR

This section will be concerned with specifying a formal model theory for the logical language that OSCAR encodes. We will then be able to say what a proposition means by giving the conditions under which it is true or false.

We seek a formalism for its twofold benefits: First, we would have a standard against which to judge the program's behavior. Secondly, the process of formalizing leads to discovering ambiguities and implicit assumptions hidden in the intuitive "interpretations" of our logical forms (semantic nets). Our criteria for the formalism is that it should mirror the program's structure as closely as possible, and should highlight the intuitions that led to our process-model.

The formal model was constructed from an egocentric point of view that does not map directly onto reality -- the interpretations of all formulas are relativized to the system's beliefs. This constraint on the model structure was inspired by the structure of the program. The program's nesting of belief and want spaces is directly reflected in the segmentation of the domain of the model. This property, in conjunction with the use of constants and known constants, is the key to solving the problem of Quine's posed in the previous section.

The rest of section 7.2 will formally specify the language and, based on its semantics, will show that a number of desirable properties of quantified beliefs hold. It is these properties that ultimately justify the structure

of the model. We then show that what is provable by the program is valid in the model. Finally, the formal semantics of our logical language is employed in clarifying the representation of various kinds of noun phrases.

7.2.1 The Language

The language, call it L, can be stated quite simply:

Term ::= constant | variable | function(term, ..., term)
Atomic formula ::= predicate(term, ..., term) | EQ(term,term)
Context ::= BELIEVE | WANT
Quant ::= ∃Variable | ∀Variable
Formula ::= Atomic formula | term Context(Formula) |
    Formula | Formula & Formula | Formula OR Formula |
        ¬(Formula) | Quant(Formula) [when the variable of
                    Quant is not bound in Formula.]

As usual, we assume infinite sets of constants, variables, function symbols, and predicate symbols.

7.2.2 Structure of the Model

A model structure for L is an 11-tuple <D,CT,S0,C,KC,A,eq,fb,fw,v,V>, where D is the domain, a non-empty denumerable set of objects; V is a valuation function mapping formulas in L to truth-values. V depends upon the values assigned to the variables of L and to the context in which it is evaluated. V employs the function v to maintain the assignments to variables of L. V also uses a special predicate Y: D --> {T, F}, as we shall see. The model also includes a partial equality predicate "eq".

CT, a denumerable subset of the powerset of D, is the set of contexts, while S0 ∈ CT is the distinguished context from which valuation will start. C, KC, and A, are functions mapping contexts to sets of elements of D.

For each context S, the following subsets of S are of interest: C(S) is the set of constants of S, KC(S) is the set of known constants of S, and A(S) is the set of agents of S. We assume that the union of all C(S) contains the distinguished element SYSTEM, and is a subset of S0. For all S in CT, A(S) SUB C(S).[2]

7.2.2.1 Belief and Want Contexts

The contexts are related by one of two partial functions fb and fw.

fb : A x CT --> CT

fw : A x CT --> CT

The function fb relates an agent and a context to that agent's belief context, accessible from the original context. Function fw does the same for want contexts. The relationship between the contexts forms a tree rooted at S0, where S0 = fb(SYSTEM,S0), i.e., S0 is the system's belief context. Want contexts for some agent are only defined in that agent's belief context. Thus, if Si is E's belief context, fw(B,Si) is undefined if B is not the same as E.

Two additional structural constraints are necessary to capture basic properties of "believe". First, "everyone believes that he believes":

∀i, ∀j, ∀x ∈ A(Si): fb(x,Si) = Sj => fb(x,Sj) = Sj

and secondly, each belief context is non-empty; each contains the agent whose belief context it is:

∀i, ∀j, ∀x ∈ A(Si): fb(x,Si) = Sj => x ∈ Sj

Thus, each C(Si) is non-empty.

In order to deal with "models of others" ("model" is used non-technically here):

∀i, ∀j, ∀x ∈ A(Si): fb(x,Si) = Sj => C(Sj) SUB C(Si)

This says that the set of constants in Si contain the constants in any belief context reachable from Si. Note that this reflects the treatment of constants in OSCAR. We

will simplify matters greatly if we allow constants to be in want contexts, but force them to be in the enclosing belief context. Hence,

$\forall j, \forall i, \forall x \in A(Si)$: $fw(x,Si) = Sj \Rightarrow C(Sj)$ SUB $C(Si)$

7.2.2.2 Known Constants

Just as there were a set of constants in each belief context so will we have a set of known constants in each belief context. Those in S0 are precisely the set of constants in S0. The sets of known constants thus have the following properties:

$KC(S0) = C(S0)$

$\forall i \geq 1$: $C(Si)$ SUB $KC(Si)$

$\forall i,j \geq 1 \; \forall a \in A(Si)$: $(fb(a,Si) = Sj \Rightarrow KC(Si)$ SUB $KC(Sj))$

The second restriction tells us that each KC(Si), where i indexes a belief context, is also non-empty because C(Si) is. We can see from the third restriction that the containment relationship between the sets of known constants is the reverse of the containment relationships for constants in those same belief contexts. This captures our intuitions that as we deal with more deeply embedded belief contexts, we know less about them -- there are fewer constants and more known constants than in an outer belief context.

7.2.2.3 Spaces vs. Contexts

The program's nesting of belief and want spaces directly influenced the structure of the belief and want contexts of the model. However, there are significant differences between the two. A belief space for OSCAR is a collection of propositions referencing network objects. Belief (and want) contexts in the model are simply collections of formal objects.

The program's treatment of terms mirrors that of the model in that OSCAR will find, or create if necessary, a

network object to be the value of any term. All propositions using the term are encoded in OSCAR as referencing this object. Similarly, the model assigns some object in the appropriate context to be the value of any term evaluated in that context. Equality of terms then depends upon the sets from which their denotations are drawn. Analogously, the program's matching routine depends upon the SORTs of its arguments.

The interpretation of formulas in the model will proceed in a fashion analogous to the way OSCAR tests propositions. Simplistically stated, the valuation function recursively winds its way through the various contexts until a formula containing no contexts can be interpreted.

7.2.3 The Valuation Function

All that remains in defining the semantics of the language is to discuss how one interprets formulas. This mapping is defined in the fashion of [Thomason 1970].

7.2.3.1 Valuation of Terms

First, v is a function that maps, in a given context, terms in L to objects in D, and also translates function symbols in L to functions over D. This dual use of v should cause the reader no difficulty.

The function v is constrained so that:

For all constant symbols K of L,
v(K,Si) is undefined or is $\in$ C(Si), where Si is some context.
(Note that the same constant symbol may be assigned different values in different contexts.)

v(Xi,Si) $\in$ Si, where Xi is any variable symbol of L.

The definition of v follows:

If F is an n-ary function symbol of L,
    v(F,Si) is an n+1-ary function over D.

v(F(T1, ..., Tn), Si) -->
    v(F,Si)(v(T1,Si), ..., v(Tn,Si), Si), where F is an
        n-ary function symbol of L and the Ti's are terms.

The above mapping of function symbols in L to functions over D depends upon the context Si. This leaves open the possibility of having different functions, in different contexts, corresponding to function symbols in L. We also maintain the possibility of having no function at all in a particular context. In fact, v(term, context) is undefined if the translation of "term" is not contained in the "context". The interpretation of a term one of whose arguments is undefined is also undefined. The rationale behind the partialness of v is that just because one can evaluate "Factorial(x)", does not mean that one believes a particular five year old child has a function corresponding to "Factorial".

7.2.3.2 Valuation of Predicates and Formulas

Let or, and, and not be the three-valued functions defined in Chapter 3. The function V interprets formulas by assigning them one of the three truth values according to the rules that follow. It is assumed throughout that evaluation proceeds as usual from inside formulas to outside ones, but if the evaluation of any argument of a formula is not defined, the truth value of the formula is U. V will also be used to map predicate symbols of L to predicates over D. If P is an n-ary predicate symbol of L, V(P,Si) is either an n+1-ary predicate over D, or is undefined.

For any predicate P:

V(P(T1,...,Tn),Si,v,Y) is defined to be

V(P,Si)(v(T1,Si),...v(Tn,Si),Si)

where Si is a context, v is the function that evaluates terms, and Y is the predicate mentioned earlier.

Even if all the arguments to the predicate which is the value of V(P,Si) are defined, V(P(T1,...Tn),Si,v,Y) may still be unknown. This form of incompleteness is allowed since the program cannot evaluate all its predicates on all objects.

For any formulas F1, F2:

V(¬F1,Si,v,Y) is not (V(F1,Si,v,Y)),
V(F1 & F2,Si,v,Y) is V(F1,Si,v,Y) and V(F2,Si,v,Y),
V(F1 OR F2,Si,v,Y) is V(F1,Si,v,Y) or V(F2,Si,v,Y), and
V(F1=>F2,Si,v,Y) is V(¬F1,Si,v,Y) or V(F2,Si,v,Y).

7.2.3.3 Valuation of Quantified Formulas, Beliefs, and Wants

∃xP(x) is taken to be true in context Si with respect to valuation v if P(x) is true in Si with respect to some v', which differs from v at most at x, and where v'(x,Si) ∈ Si. Whenever an existentially quantified variable x occurs in a subformula of the form A BELIEVE P(x) the value chosen for x must exist and be a known constant in the belief context for A accessible from Si. The predicate Y marks those values that must be tested, upon entry to belief context Si, for membership in the set of known constants in Si. More formally,

V(∃xF1,Si,v,Y) is T if there are v' and Y' such that
V(F1,Si,v',Y') is T where:
v'(x,Si) ∈ Si,
v'(x,Sj) = v'(x,Si) for all Sj ³
v'(y,Sj) = v(y,Sj) for all y ≠ x, whenever
v(y,Sj) is defined
v'(y,Sj) is undefined otherwise

and

Y'(a) is true iff a = v'(x,Si), or Y(a) is T.
Y'(a) is false if Y(a) is false.

V(∃xF1,Si,v,Y) is F otherwise.

Universal quantification will be defined in terms of negation and existential quantification, in the usual way. Note that a quantified formula is either true or false.

Now, we can define the valuation of BELIEVE and WANT as fcllows:

V(A BELIEVE(F1),Si,v,Y) is defined to be
V(F1,fb(a,Si),v,Y) and
for every x free in F1, if Y(v(x,fb(a,Si)))
then v(x,fb(a,Si)) ∈ KC(fb(a,Si)).
(Where a = v(A,Si).)

V(A WANT(F1),Si,v,Y) is V(F1,fw(v(a,Si),Si),v,Y)

7.2.4 Partial Equality: eq

In order to allow for unknown valuations of statements of equality, the model provides a partial equality predicate, eq, rather than the usual identity predicate. We consider identity to be a meta-level concept and hence the predicate is not accessible from the object language.

The identity predicate compares elements of the domain and thus is two-valued. However, the following is a situation where we would want the valuation of a statement of identity to be U. Assume a model that satisfies

(i) SBRB(Spy(Aunt(Mary)) & Professor(Mother(Sue))) is T (where B is an abbreviation for BELIEVE, and R abbreviates RALPH.)

This ensures that the terms evaluate to objects in Ralph's belief context. Then, the valuation of

(ii) SBRB(Aunt(Mary)=Mother(Sue)), would yield T or F

according to its interpretation below:

```
v(Aunt,fb(r,S0))(v(Mary,fb(r,S0))) =
    v(Mother,fb(r,S0))(v(Sue,fb(r,S0)))
```

Since there is no reason to assume that Ralph believes the identity is true or is false, we want the valuation to be unknown. A partial equality, EQ, is thus employed at the object language level such that:

```
V(EQ(T1,T2),Si,v,Y) --> eq(v(T1,Si),v(T2,Si),Si)
```

Below we state restrictions that eq must satisfy.

If x or y is undefined, then eq(x,y,context) is U.
/* i.e., the argument is not in the context */

If x is identical to y, then eq(x,y,context) is T.

If x ∈ C(context) and y ∈ C(context)
then eq(x,y,context) is F.
/* constants should be identical to be equal. */

If x ∉ KC(context) or y ∉ KC(context),
then eq(x,y,context) is U.

There are thus certain cases where we can guarantee that eq will yield T or F, viz. when it is comparing known constants.

7.2.5 Satisfiability, Etc.

A formula is satisfiable if there is at least one interpretation in which its valuation in S0 is T. A formula is unsatisfiable if there is no interpretation in which its valuation in S0 is T. A formula is valid if every valuation in S0 is T or U.

7.2.6 Theorems Regarding Quantification and Belief

Given this interpretation of the language, there are a number of theorems that can be proven regarding the behavior of constants and known constants that were previously argued for on the basis of the program's manipulations. These theorems will make use of, and justify, the formulation of the model's structure. Various alternative structures will be shown to violate at least one of the three theorems. In the statement of the theorems, let Q be the name of any agent; let A be some constant in the language; and let P be some predicate symbol.

1. Q BELIEVE(P(A)) => ∃x Q BELIEVE (P(x)) is valid.
2. ∃x Q BELIEVE(P(x)) => Q BELIEVE(∃x P(x)) is valid.
3. Q BELIEVE(∃x P(x)) => ∃x Q BELIEVE(P(x)) is not valid.

To prove the implications are valid, it is sufficient to show that any interpretation that satisfies the antecedent also satisfies the consequent. To simplify notation, let B be an abbreviation for BELIEVE.

Theorem 1: Q BELIEVE(P(A)) => ∃x Q BELIEVE(P(x)) is valid.

Proof:

Let the model structure
<D,CT,S0,C,KC,A,eq,fb,fw,v,V> satisfy QB(P(A)).

Then, V(QB(P(A)),S0,v,Y) is T.

Then: V(P(A),fb(q,S0),v,Y) is T where q is v(Q,S0)

Then: V(P,fb(q,S0))(a,fb(q,S0)) is T, where a is v(A,fb(q,S0)).

For this valuation to be T, a ∈ fb(q,S0) and, in fact, is a constant in context fb(q,S0) and thus also in context S0. (*)

Then: there is a v' like v except that v'(x,S0) = v(a,fb(q,S0)) and v'(x,Si) = v'(x,S0) for all Si, such that

V(P(x),fb(q,S0),v',Y') is T. (Y' is like Y except that Y'(a) is T.)

Then: V(QB(P(x)),S0,v',Y') is T. Finally,

V(∃xQB(P(x)),S0,v,Y) is T. Q.E.D.

Remarks:

Theorem 1 relies (in step (*)) on the containment relations for constants in the model structure. This theorem is thus a direct consequence of our modelling strategy.

Theorem 2: ∃x QB(P(x)) => QB(∃yP(y))

Proof: Let the antecedent be satisfied by the model structure <D,CT,S0,C,KC,A,eq,fb,fw,v,V>. Then, V(∃xQB(P(x)),S0,v,Y) is T.

Then: there is a v' like v except in assignment to x where v'(x,S0) ∈ S0 and v'(x,Sj) = v'(x,S0) ∀ Sj, and there is a Y' like Y except Y'(v'(x,S0)) is T, such that: V(QB(P(x)),S0,v',Y') is T.

Then: V(P(x),fb(q,S0),v',Y') is T, where v'(Q,S0) is q, and since Y'(v'(x,fb(q,S0)) is T (recall that v'(x,fb(q,S0)) = v'(x,S0)) then v'(x,fb(q,S0)) ∈ KC(fb(q,S0)). In other words, v'(x,fb(q,S0)) is a known constant in Q's belief context. Obviously, v'(x,fb(q,S0)) ∈ fb(q,S0).

Then: $V(P,fb(q,S0))(v'(x,fb(q,S0))$ is T.

Now, to show that the model structure satisfies $QB(\exists y P(y))$, we must realize that there is a v'' which is like v' except that $v''(y,fb(q,S0)) = v'(x,fb(q,S0))$ such that: $V(P(y),fb(q,S0),v'',Y')$ is T.

Clearly, then $V(\exists yP(y),fb(q,S0),v',Y')$ is T.
But, this is the interpretation of $V(QB(\exists yP(y)),S0,v',Y')$.
Since v' is exactly like v except in assignment to x (and there is no x in this formula), and analogously for Y' and Y, then

$V(QB(\exists yP(y)),S0,v,Y)$ is T. Q.E.D.

Remarks:

Theorem 2 relies on the interpretation of "Quantifying in" that says the object assigned by v' to the bound variable x is a known constant in the embedded belief context. However, even if the model structure did not have KC's, the definition of v would guarantee that P(x) would only be T in fb(q,S0) when the valuation of x is in that context.

Theorem 3: $QB(\exists x(P(x)) \Rightarrow \exists yQB(P(y))$ is not valid.

Proof: We shall show that there is an interpretation satisfying the antecedent for which the consequent is F.

- Let P translate to the predicate yahi (an extinct amer-indian tribe).
- Let Q be the constant john $\in$ S0, and let there be only 2 objects in fb(john,S0), john and object d $\in$ fb(john,S0), where d is neither a constant nor a known constant in that context. To make this concrete, d will be the valuation of Abraham Lincoln's cousin's sister-in-law in John's belief context. John is thought to believe she

was a Yahi, but SYSTEM does not believe Lincoln had such a relative. (Recall that S0 is fb(SYSTEM,S0).)

- Let john's belief space, fb(john,S0) be called S1. Let yahi(john,S1) be F.

Now, our interpretation of the antecedent, skipping all intermediate steps, is:

∃ a v' differing from v only in assignment, in all spaces, to x, where v'(x,S1) ∈ S1, and ∃ a Y' like Y except Y'(v'(x,S1)) = T. But, since d is the only element of S1 such that John believes it satisfies the predicate yahi, then let v'(x,S1) be assigned d. The antecedent now has the valuation T. However, we constructed the interpretation so that d was not a known constant in S1 and thus d cannot be the valuation of y. Every valuation of the consequent of the implication then yields F since, there being only two elements in S1, v''(y,S0) must be assigned the object john and yahi(john,S1) is F. Q.E.D.

7.2.6.1 Alternative Model Structures

Let us propose a number of alternative model structures and show why each is inadequate.

Alternative 1:

Eliminate the sets C and KC, and impose the constraints that if Sj = fb(a,Si) for some a ∈ A(Si), then Sj SUB Si.

This model structure would allow Theorem 1 to be true since we could existentially generalize from Sj to Si. However, Theorem 3 would be false.

Clearly, if we reverse the containment relationship, Theorem 2 is true but Theorem 1 is false. If we make Si = Sj, then Theorem 3 is false. So it seems that we need to

use a subset of Si. Let us then abandon Alternative 1 and propose

Alternative 2:

If Sj = fb(a,Si) for some a ∈ A(Si),
Then C(Sj) SUB C(Si)

The model structure thus allows for Theorems 1 and 3 to hold, while the definition of v ensures that Theorem 2 holds. Notice that there is no mention of KC here. The next section will point out the difficulty with this alternative.

7.2.6.2 Rejecting Alternative 2 Constants

The model structure specified by Alternative 2 does not use known constants. Recalling that the KC sets were used by Y in interpreting existentially quantified beliefs, let us now assume that, in the definition of the valuation of BELIEVE, those objects marked by Y were simply arbitrary elements of the embedded belief context. By this account, the system's "quantifying into" someone's belief context would have not effect on how it manipulates the formulas comprising his beliefs. We would have no way of representing that a person can tell if the values of two terms are the same since those values would be undifferentiated in kind (cf. the example regarding EQ). Hence, we need some subset whose members correspond to values of terms that he "knows".

If those objects marked by Y were constrained to be in the set of constants in the embedded belief context (say fb(t,fb(q,S0))). Under these conditions, the following undesirable generalization would take place:

SB QB ∃xTB(P(x)) => SB ∃xQB(TB(P(x)))

This would occur because x would be assigned a constant in T's belief context. But, because of the containment relation among the sets of constants, the valuation of x would be a constant in Q's belief context. The same can be said of its presence in the set of constants in S's belief context, thereby justifying this unwanted generalization. The known constants in any embedded belief context ensure that precisely one level of existential generalization takes place.

7.2.7 Difficulties with the Interpretation

The model structure defined above used two functions to relate belief and want contexts. However, while we were able to obtain the desired behavior for BELIEVE specified in the theorems, two desirable properties of BELIEVE do not hold. Namely, this interpretation equates the valuations of QB(¬P) with ¬QB(P) (and similarly equates QB(P) OR QB(R) with QB(P OR R)).

The value of using a "possible worlds" semantics, as in Kripke [1963], is that the interpretation of modal operators introduces a quantification over possible worlds that blocks the movement of negation, thereby enabling such a semantics to distinguish between the above cases.

A topic for future research, then, is to reformulate the semantics for our logical language so that we obtain the appropriate semantics for negated and disjoined beliefs. The most promising approach appears to be that of modal logic, using a "possible worlds" semantics. While we cannot simply extend the semantics given in this chapter to a three-valued modal logic, since the quantification over possible worlds that arise in formulating the semantics of BELIEVE are two-valued, we may be able to capture the program's four truth-values in a classically two-valued fashion. The approach is to view the program's answer of

UNK to the testing of USER BELIEVE P as corresponding to NOT (USER BELIEVE P) AND NOT (USER BELIEVE NOT(P)), where NOT and BELIEVE are two-valued. The program's KNOWN truth-value might then correspond to (USER BELIEVE P) OR (USER BELIEVE NOT(P)), enabling us to maintain the relationship that KNOWN means "not unknown".[4]

7.2.8 Relationship of the Program to the Model:

Apart from the structural similarities between the model and the program, we would ideally like to show the soundness and completeness of our proof system (the program) with respect to the model. While, no attempt has been made here to construct a complete deductive system, we shall, however, be able to show that the deductions that the program does make are sound, i. e., what is provably true (false) is valid (invalid) in the model.

Let A = A1 & ... & An be the conjunction of axioms contained in OSCAR. Then we shall show by induction on the length of the formula and on the embedding of belief contexts, that if test(F1,QB) is T (where F1 is any formula and QB is any space), then every valuation of A => F1 in the corresponding belief context is T. Similarly, if test(F1,QB) is F, then the conjunction of the axioms and F1 will be shown to be unsatisfiable.

More formally, let space-string be our usual form of labelling belief and want spaces (e.g., "SBUBRBRW"). We shall define a function, Corr that produces a correspondence between that string and the model's belief contexts as fcllows:

Corr(space-string, context) =

context, if space-string is nil

Corr(space-string', fb(a,context)), if space-string is of the form "aBspace-string'",

Corr(space-string', fw(a,context)),
if space-string is of the form "aWspace-string'".

We shall assume a function Space that finds the current belief or want space given a space-string. (This function is part of the "totests" of BELIEVE and WANT.) Of course, the space-string must start with "SB" in order that the correspondence with S0 be exact.

Lemma

For every formula F1 and for every Space-string:

If Test(F1,Space(Space-string)) = T
Then V(F1,Corr(Space-string,S0),v,Y) = T or U, ∀v.

If Test(F1,Space(Space-string)) = F
Then V(F1,Corr(Space-string,S0)) = F or U, ∀v.

Proof:

1.) Base Case: Assume F1 is atomic. Let sp_st be some Space-String.

a). If Test(F1,Space(sp_st)) = T, then F1 is an axiom. (Recall that OSCAR has no rules of inference.) We show that A => F1 is valid, or simply, F1 => F1 is valid. In other words, we shall show that V(¬F1 OR F1,Corr(sp_st,S0),v,Y) is T or U ∀v. This is valid by definition.

b). If Test(F1,Space(sp_st)) = F then ¬F1 is an axiom. Hence, we show that V(¬F1 & F1,Corr(sp_st,S0),v,Y) is F or U ∀v. Again, this is true by definition.

Induction Hypothesis (IH):

Assume the theorem is true for all formulas shorter than F1 and all Space-strings of which sp_st is an initial substring. (We assume here that all combinations of the form "aBaB" have been reduced to "aB".)

For non-atomic formulas F1 and arbitrary sp_st:

2). If F1 is of the form F2 & F3:

Test(F1,Space(sp_st)) = T iff Test(F2 & F3,Space(sp_st)) iff
Test(F2,Space(sp_st)) = T and Test(F3,Space(sp_st)) = T.

Then we show that:

V(¬A OR (F2 & F3),Corr(sp_st,S0),v,Y) is T or U ∀v, i.e.,
V(¬A OR F2,Corr(sp_st,S0),v,Y) and
V(¬A OR F3,Corr(sp_st,S0),v,Y) is T or U ∀v,

Each of these conjuncts is T or U for all valuations v by IH. (Similarly, for the case of Test(F2 & F3,Space(sp_st)) = F.

3a). F1 is of the form Q BELIEVE F2:

Test(Q BELIEVE F2,Space(sp_st)) = T =>
Test(F2,Space(sp_st||"QB")) is T. We now show that
V(¬A OR Q BELIEVE(F2),Corr(sp_st,S0),v,Y) is T or U ∀v.
This interpretation reduces to
not V(A,Corr(sp_st,S0),v,Y) or V(F2,fb(q,Corr(sp_st,S0)),v,Y)
But, by IH, F2 is true in a belief space whose prefix
is sp_st and hence all valuations of F2 are T or U in the
context corresponding to that space.

3b). Test(Q BELIEVE(F2),Space(sp_st)) is F.

Either i). Test(F2,Space(sp_st||"QB")) is F or
ii). Test(¬Q BELIEVE F2,Space(sp_st)) is T.

For case i), we show that V(A & Q BELIEVE(F2),
Corr(sp_st,S0),v,Y) is T or U for all valuations v. This
becomes V(A,Corr(sp_st,S0),v,Y) and V(F2,fb(q,
Corr(sp_st,S0)),v,Y). By IH, this latter conjunct is F or U
for all v and hence this case is established.

Case ii) is true iff ¬Q BELIEVE F2 is an axiom. We can show this case to be true in the same way as we established the base case of the proof.

4a). F1 is of the form ∃xF2:

Test(∃xF2,Space(sp_st)) is T iff either ∃xF2 is an axiom (a trivial case) or ∃ a C such that F with C substituted for x is T in Space(sp_st). We then show that V(¬A OR ∃xF2, Corr(sp_st,S0),v,Y) is valid. The interpretation of the above is that there is a v' like v except in assignment to x, and a Y' like Y except that Y'(v'(x,Corr(sp_st,S0))) is T such that:

V(¬A OR F2,Corr(sp_st,S0),v',Y') is T or U for all valuations.

Clearly, there always is such an assignment, v'(x,Corr(sp_st,S0)) = v(C,Corr(sp_st,S0)), and hence we are done.

4b). When Test(∃xF2,Space(sp_st)) is F, ¬∃xF2 is an axiom, and thus this case follows in the usual fashion.

The remaining cases ¬, OR, WANT are proved similarly. Thus, we have shown that for every space (and analogously, context) the lemma is true. The following theorem then follows directly:

Theorem: If Test(F1,Space("SB")) is T then V(A => F1,Corr("SB",S0,v,Y)) is valid, and if Test(F1,Space("SB")) is F, then V(A & F1,Corr("SB",S0),v,Y)) is invalid. (Recall that fb(S,S0) = S0, just as SBSB is the same space as SB.)

This theorem captures the relationship we desire since testing and valuation start at space SB and context S0, respectively.

For cases of incomplete knowledge, the program and the model can be related in the following ways: We can show

that if all valuations of a formula F1 in space S0 are unknown, then test(F1,SB) would yield unknown as well. This is easy to see since if there were a case where OSCAR could prove or disprove the formula, then there is a valuation, which could use the same objects as OSCAR) that would not be U. (This would contradict our assumption.)

Finally, we can show that when the program arrives at the answer KNOWN for some formula, there is some interpretation for which we can guarantee that the valuation of that formula is not U. This case will arise when determining if a known constant (typically, the value of a function) is equal to another known constant. Thus, as an example, consider a model whereby the valuation of ∃xQB(EQ(F(c),x)) is T. We can then show the valuation of QB(EQ(F(c),b)) is not U.

Let I = <D,CT,S0,C,KC,A,eq,fb,fw,v,V> be a model structure. Since V(∃xQB(EQ(F(c),x)),S0,v,Y) is T,

Then: there is a v' like v except that v'(x,S0) ∈ S0 and
∀Si, v'(x,Si) = v'(x,S0), and ∃ a Y' like Y except
Y'(v'(x,S0)) is T, such that:
V(QB(EQ(F(c),x)),S0,v',Y') is T.

Then: V(EQ(F(c),x), fb(q,S0),v',Y') is T where
q = v'(Q,S0) and v'(x,fb(q,S0)) ∈ KC(fb(q,S0))
(i.e., v'(x,fb(q,S0)) is a known constant in Q's belief context.)

Now, let f be the translation of the function F in context fb(q,S0). Then the above valuation becomes:

(1) eq(f(v'(c,fb(q,S0))), v'(x,fb(q,S0))) is T.

Note that v'(c,fb(q,S0)) is a constant in fb(q,S0).
We shall assume this property of "b" as well in trying to interpret QB(EQ(F(c),b))

V(QB(EQ(F(c),b)),S0,v,Y) is: (skipping all intermediate steps)

(2) eq(f(v(c,fb(q,S0))), v(b,fb(q,S0))),
which we have to evaluate.

Now, since v' agrees with v on the value of c, formula (1) becomes

(3) eq(f(v(c,fb(q,S0))), v'(x,fb(q,S0))) is T

Using the transitivity of eq, we discover that
eq(v'(x,fb(q,S0)), v(b,fb(q,S0))) has valuation

T if the two are identical
F if v'(x,fb(q,S0)) is a constant
Since both are at least known constants in context fb(q,S0), eq is constrained not to yield U.

The typical example of this is

∃x John Believe (Loc(Mary) is x)

Then, OSCAR's evaluation of John Believe (Loc(Mary) is Inroom) will yield KNOWN, while the valuation will not be U.

7.2.9 Summary

We have seen that we can construct a formal semantics for the higher-order language of beliefs and wants that OSCAR encodes. The model theory described here supports the kinds of inferences the program can draw from quantified belief formulas. As one would expect, if a formula is provable by the program, for a given set of axioms, it is valid in the model.

The model structure was directly inspired by the egocentric structure of the program. However, the program's memory was itself structured to support certain desired capabilities of a process-model of the human use of language. Hence, the motivation of modelling humans directly affected the shape of the formalism. This is evident from our use of partial valuation functions and

unknown truth-values. We are now able to use this formal semantics to explain a number of confusions that have been noted by linguists and philosophers.

7.3 Applying the Formalism

We shall illustrate the formalism by giving logical forms corresponding to different readings of a number of ambiguous sentences. Given the interpretation discussed above, the meanings of these logical forms should be clear. Unless otherwise indicated, the sentences will be analyzed as though the system spoke them and hence the logical forms presented indicate what the system's representations are that would lead to such utterances.

7.3.1 Simple Opacity of Belief

First, let us reconsider the problem of opacity in terms of the sentence

(A) "Ralph believes Ortcutt is a spy."

The first reading of this corresponds to "The man I call Ortcutt is believed by Ralph to be a spy." The logical form would be:

(A.1) System Believe ∃x (EQ(x,Ortcutt) &
Ralph Believe (Spy(x)))

Given our interpretation, (with "SB labelling belief context S0, and "SBRB" labelling fb(r,S0)), the value chosen for x is a known constant in SBRB and is equal to the constant Ortcutt in SB. Notice that Ralph is believed to know who x is, but is not believed to know (from just this statement) that x is Ortcutt. We are not entitled to conclude that SBRB(Spy(Ortcutt)) on the basis of the identity of the valuations of 'x' and 'Ortcutt' in context S0. Such a move fails since the valuation of 'Ortcutt' need not be the same in the two belief contexts.

While we have represented the sentence, it should be emphasized that we have not discussed why the speaker (OSCAR) would choose to refer to the man by the proper name "Ortcutt" rather than by some other description. Why a

speaker describes something in a particular way depends upon his knowledge of the hearer, since he wants to ensure that the hearer can understand the reference. Thus, for the system to inform the hearer of (A), intending for the hearer to represent the utterance as (A.1), it would have to believe the hearer would understand the name "Ortcutt". Throughout these examples, it will be important to keep in mind the question of why one would utter a sentence in the way shown.

The second reading of sentence (A), corresponding to "The man Ralph calls Ortcutt is believed by Ralph to be a spy", is:

(A.2) System Believe ∃x Ralph Believe
(EQ(x,Ortcutt) & Spy(x))

The value assigned to x in SBRB is a known constant that Ralph believes is equivalent to Ortcutt. For the same reasons as above, equality of 'x' and 'Ortcutt' cannot be concluded in context SC.

Finally, we can now present the system's representations for analogues of (IV), (V), and (VI):

"Ralph believes Mary's mother is a spy."

(A.3) System Believe Ralph Believe (Spy(Mother(Mary)))

"Ralph believes Judy's aunt is not a spy."

(A.4) System Believe Ralph Believe (¬Spy(Aunt(Judy)))

"Judy's Aunt is Mary's mother."

(A.5) System Believe (EQ(Aunt(Judy),Mother(Mary)))

No substitutions can be made, just on the basis of the system's belief that Judy and Mary are cousins, into either A.3 or A.4 since:

1. The functions might evaluate differently in different belief contexts, and

2. The system does not believe that Ralph believes the equality in (A.5).

Thus, all three formulas can be simultaneously true without our believing that Ralph holds contradictory beliefs.

7.3.2 Indefinite Noun Phrases

The problem to be considered is the nature of the logical forms underlying indefinite noun phrases in sentences using "believe" and "want". I shall only examine sentences discussed in [Partee 1972]. Undoubtedly, there are other forms of indefinite noun phrases.

Let us consider the following sentence:

(B) "John wants to marry a woman that his mother disapproves of."

Recall that a person's wants are always embedded within their beliefs. Hence, one representation of (B) (using abbreviations for System, John, BELIEVE, and WANT) is:

(B.1) SB JB JW ($\exists$x [Marry(J,x) & Woman(x) & Disapprove(Mother(J),x)])

In this case, John does not necessarily believe such a woman exists. He wants that whomever he marries be a woman in disfavor with his mother. Contrast this with the representation (B.2) of the reading of (B) where John and the system both believe she exists, John knows who she is, believes his mother disapproves of her, and simply wants to marry her:

(B.2) SB $\exists$x JB (JW(Marry(J,x)) & Disapprove(Mother(J),x) & Woman(x))

This could be glossed as: "John wants to marry a woman that his mother disapproves of, but I don't know who she is." Here, he is also thought to believe that his mother disapproves of her.

In another variation we can add John's mother's disapproval as one of his goals. Hence:

(B.3) SB ∃x JB JW (Marry(J,x) & Disapprove(Mother(J),x) & Woman(x))

In the next form, the system need not believe that John knows who the intended bride is -- he may only know that she exists. Thus we have

(B.4) SB JB ∃x [JW(Marry(J,x)) & Disapprove(Mother(J),x) & Woman(x)]

Here, John believes that there is (at least) one woman of whom his mother disapproves, and he wants to marry one of them. Lastly, another possibility is:

(B.5) SB JB ∃x [JW(Marry(J,x) & Disapprove(Mother(J),x)) & Woman(x)]

John does not necessarily believe that there is some woman who is out of favor with his mother. He simply wants that he and his mother disagree about whoever will be his wife.

The extra level of belief added to representations mentioning someone else's goals leads to new possibilities for quantification. This becomes the source of most of the variability needed to handle the numerous interpretations. We can now give representations where John may or may not believe the woman exists, or, when it is thought that John believes she does exist, the speaker may or may not know who she is.

7.3.3 Definite Noun Phrases: Referential vs. Attributive

The main differences between the logical forms of definite and indefinite noun phrases are the uniqueness and presupposed existence of the referent of the phrase. Partee's example is:

(C) "John wants to murder the man in Apt. 3."

Again, we could interpret this statement as saying that John wants to murder whoever lives in Apt. 3 (perhaps because he is making too much noise), or that John has a specific victim in mind. Donnellan [1966] refers to the former as the attributive reading, where the speaker does not know the referent and is attributing a property to whatever it is. He contrasts this with the referential reading where the referent is known to both speaker and hearer. I shall give a number of interpretations of (C) and then rearrange the logical forms to produce other legitimate logical forms, only some of which underlie the sentence.

Consider the system's referential representation of statement (C) (where the system is the hearer):

(C.1) SB ∃x [JB(JW (Murder(J,x)) & Live(x,Apt.3) & Man(x) & ∀y(Live(y,Apt.3) => EQ(y,x)))]

Here, the man chosen as value for x is a known constant for John and John believes that man lives in Apt. 3, alone. Notice that the system does not know who John's intended victim is.

The prototypical attributive reading would be:

(C.2) SB JB ∃x [JW(Murder(J,x)) & Live(x,Apt.3) & Man(x) & ∀y(Live(y,Apt.3) => EQ(y,x))]

In this case, John is believed to want to murder whoever lives in that apartment (and John believes there is only one

such man). The difference between the logical forms of the referential and attributive readings of noun phrases is that the former involves a quantification into a belief context.

Clearly, we can create new logical forms with the formulas representing John's uniquely living in Apt. 3 being inside the scope of John's wants, or outside the scope of John's beliefs. A few of these formulas will be presented, but making intuitive sense of them is left to the reader. Not all of them need be regarded as representations of sentence (C).

(C.3) SB ∃x [JB(JW(Murder(J,x))) & Man(x) & Live(x,Apt.3) & ∀y(Live(y,Apt.3) => EQ(y,x))]

Gloss: there is some person that John wants to murder; John knows who that person is; the system thinks that person is whoever is uniquely living in Apt. 3 and John is not believed to know this.

(C.4) SB ∃x JB (JW(Murder(J,x) & Live(x,Apt.3) & ∀y(Live(y,Apt.3) => EQ(y,x))) & Man(x))

Gloss: John wants to murder a particular man that he knows and wants to do it in Apt.3, where he wants that man to be living alone.

(C.5) SB JB ∃x JW(Murder(J,x) & Live(x,Apt.3) & Man(x) & ∀y(Live(y,Apt.3) => EQ(y,x)))

Gloss: John wants that whomever he murders be a man living alone in Apt. 3.

The next logical form I find to be unintuitive. It involves splitting up the Live predicate from the statement of its uniqueness by a WANT context.

(C.6)? SB ∃x JB (JW(Murder(J,x) & Live(x,Apt.3)) & Man(x) & ∀y(Live(y,Apt.3) => EQ(y,x)))

On the other hand, it is possible to make intuitive sense of:

(C.7) SB ∃x JB (JW(Murder(J,x)) & Man(x) & Live(x,Apt.3)) & ∀y(Live(y,Apt.3) => EQ(y,x))

John knows whom his victim will be and believes the unfortunate soul lives in Apt. 3. But the system, not knowing who John's intended victim is, believes that only one person lives in Apt. 3 and whoever that is is being stalked by John. The difference between this case and (C.6) is that for (C.6), x's occupying Apt. 3 is <u>desired</u>, while for (C.7), it is believed that Apt. 3 houses only one person.

Other rearrangements are possible but will not be presented here.

There are ways of issuing sentence (C) where the speaker knows more than he is saying. For instance, the system, as speaker, may have the proposition: (A combination of A.1 and C.1)

(C.8) SB ∃x [EQ(x,Ortcutt) & JB(JW(Murder(J,x)) & Man(x) & Live(x,Apt.3) & ∀y(Live(y,Apt.3) => EQ(y,x)))]

Here, S may know that the term 'Ortcutt' denotes the man that John wants to murder. Yet, S, as speaker, may or may not choose to issue the utterance as (C) depending upon his beliefs about the hearer.

Indeed, we can have:

(C.9) SB JB(Man(Ortcutt) & JW(Murder(J,Ortcutt)) & Live(Ortcutt,Apt.3) & ∀y(Live(y,Apt.3) => EQ(y,Ortcutt)))

The system believes both it and John know of Ortcutt and yet it may still refer to the targeted man as "the man in Apt. 3", if it believes the hearer will not understand the term 'Ortcutt'.

7.3.4 Definite Noun Phrases without Propositional Attitudes

The dependence of the speaker's choice of referring phrase on his knowledge of the hearer is what is behind the ambiguity of the statement:

(D) "The murderer of Kennedy is insane."

In this sentence, no one's beliefs or wants are mentioned and hence there is no explicit opaque context in the sentence. However, we must not forget that we are dealing with a speech act that encodes the speaker's beliefs. The system, as the speaker of this sentence, might have the following representations.

(D.1) SB(Murder(Oswald,Kennedy) & Insane(Oswald) &
∀x(Murder(x,Kennedy) => EQ(x,Oswald))), or,

(D.2) SB ∃x [Murder(x,Kennedy) & Insane(x) &
∀y(Murder(y,Kennedy) => EQ(y,x))]

Form (D.1) is the referential case -- one where the system knows the referent of the definite noun phrase. In order to utter (D) when having the information (D.1), the system would probably have to reason that the hearer did not know the term 'Oswald'. Form (D.2) is the attributive case -- whomever murdered Kennedy is insane. Notice that the system cannot issue (D) referentially (in a sentence that does not make use of propositional attitudes) without in fact knowing what the referent is. Since all the known constants in SB are constants in that space, the system either knows who the murderer is or it doesn't.

If the user spoke sentence (D) to the system, however, then (D.3) and (D.4) could be the system's referential and attributive representations.

(D.3) SB ∃x UB(Murder(x,Kennedy) & Insane(x) &
∀y(Murder(y,Kennedy) => EQ(y,x)))

(D.4) SB UB ∃x [Murder(x,Kennedy) & Insane(x) &
∀y(Murder(y,Kennedy) => EQ(y,x)))

7.3.5 Acquiring Quantified Beliefs

Since referential definite and "specific" indefinite noun phrases are now viewed as being represented by quantified beliefs, the system, as hearer, will continually be acquiring quantified beliefs about its conversants. The important point to notice is that such beliefs are what enables a hearer to plan further questions of the speaker of those noun phrases. For instance, upon interpreting the user's utterance "The murderer of Kennedy is insane" referentially, the system would have sufficient information to plan the question "Who murdered Kennedy?" if it could not identify the referent itself.

We have thus given a formal basis for the claim that by uttering a referential definite noun phrase, the speaker "implies" he can identify the referent if called upon to do so. (See [Searle 1969, ch. 4] for a discussion of this aspect of reference.) The formalism also suggests that the speaker must be prepared to further identify referents for any indefinite noun phrases for which he has intended the "specific" interpretation. However, we have not given criteria for how one would decide to interpret a definite noun phrase as referential or attributive (or, analogously, as specific or nonspecific).

7.3.6 Summary

Partee suggests that there is much confusion as to how to handle noun phrases such as the ones in sentences B, C, and D. She notes similarities between ambiguities in definite and indefinite noun phrases that argue against an explanation solely in terms of either opaque contexts or the presupposed existence of the referents of the noun phrases.

Sentence D shows that the same ambiguities can arise when neither of these two features are present.

We have shown that her examples can all be handled with the formal semantics proposed here. We can summarize the findings for these noun phrases as follows:

- The referential/attributive distinction for definite noun phrases involves quantification into a belief context. The extra level of belief brought into the representation of "want" is what enables us to represent the different readings of sentence D.

- The presupposed existence (or not) of the referent of an indefinite noun phrase involves an existentially quantified variable either inside a person's want context or outside it within a belief context (either that person's or the speaker's).

- The only difference between the representation of the above definite and indefinite noun phrases is the statement of uniqueness.

Finally, referential/attributive distinctions in readings of statements that do not mention "belief" or "want" were regarded from both the speaker's and hearer's points of view. A sincere speaker could intend his noun phrase to be interpreted referentially only if he knew what the referent was. The hearer's version of the speaker's statement might involve a quantified belief for the speaker. Such quantified beliefs then give the hearer the information needed to plan questions for the purpose of clarifying the referents that the speaker intended for noun phrases.

Donnellan mentions other cases where the speaker and hearer both know to whom he is referring and yet the referring phrase is intended to be attributive. Such cases are not considered here partly because I believe the source

of that mystery may lie in other aspects of the process of referring and not in the underlying representation.

7.4 Notes

1. I will use "believe" as the prototypical propositional attitude and will show how to deal with "want" as well.
2. For typographical purposes, the intersection operator will be INT, and the subset operator will be SUB (A SUB B means A is a subset of B), with SUB meaning proper subset.
3. "=" is the meta-language symbol for identity.
4. This view was suggested to me by James Allen who is working on such a modal logic for belief.

CHAPTER 8

EXTENSIONS OF OSCAR'S BELIEF REPRESENTATION FOR MUTUAL BELIEF

Contents:

8.1 Motivations

In chapter 2, it was claimed that a speaker intends to produce certain effects on his hearer partly by means of getting the hearer to recognize that he (the speaker) intends to produce those effects. This recognition of intention is said to form the basis of communication. If we ever hope to make this claim precise, we ought to try to understand what those effects are.

The approach taken thus far to modelling the effects of speech acts has been a simplified one in that it assumes that communication produces a finite set of effects. For instance, the effects of an INFORM involved only two levels of belief. While speakers may plan their speech acts to produce specific beliefs, they cannot, I claim, avoid producing an unbounded number of them. The reason for this is that in a communicating environment, each of the parties are aware, and are aware that the others are aware (and so on), of the effects normally produced by speech acts. The result of this mutual awareness is what will be termed mutual belief. [1]

In this chapter, I will show how OSCAR can represent mutual belief with a modified memory representation. I will

then define the relation MUTUAL_BELIEF and discuss its associated operations. Since the concept is motivated here by the effects of speech acts, I will redefine REQUEST and INFORM to incorporate MUTUAL_BELIEF and then briefly indicate how plans using these new speech acts might proceed.

8.2 Example

As an illustration of the concept, consider the situation where you and OSCAR are having an argument. You claim the Earth is round and OSCAR replies that it is flat. What we would like to know is what OSCAR's beliefs are about the effects of these utterances.

First of all, OSCAR's belief that the Earth is flat becomes shared knowledge -- you believe that OSCAR thinks the Earth is flat, but you do not adopt this belief yourself. Since this is what OSCAR thinks you believe, the representation of this is in its model of you.

In a similar fashion, upon OSCAR's understanding your statement that the Earth is round, the effects of your INFORM take place -- namely, OSCAR then thinks that you believe the Earth is round. However, OSCAR also knows that you believe the effects of your INFORM have taken place and hence OSCAR believes that you believe that OSCAR thinks .... But, there is no reason why we should stop here. In an abbreviated form, the following beliefs have now been created in OSCAR's memory.

SBUB(ROUND(EARTH))	SBUBSB(FLAT(EARTH))
SBUBSBUB(ROUND(EARTH))	SBUBSBUBSB(FLAT(EARTH))
:	:
:	:
:	:

In order to be faithful to the set of inferences that people can draw from an utterance, a representation scheme must be able to handle, in a finite fashion, the unbounded set of beliefs that such communication acts create. The

method proposed here provides a compact way of representing these mutual beliefs. The essence of the scheme is in realizing that since OSCAR's knowledge of the user's belief is usually acquired in communicative situations, it is knowledge that the user thinks OSCAR has (and OSCAR knows that). The unusual cases are those where OSCAR acquires information about the user's beliefs from sources other than the user; here, the user would not be believed to share such knowledge.

8.3 Definition and Representation of Mutual Belief

We will define a new statement in the language:
mutual_belief between (x := a person) and
(y := a person) that (prop := a proposition)
The effect that an assertion of such a statement should have is:

X BELIEVE PROP
X BELIEVE Y BELIEVE PROP
X BELIEVE Y BELIEVE X BELIEVE PROP ²

:

By adjusting the memory representation to make all these beliefs available at once. OSCAR's memory structure will now be the following:

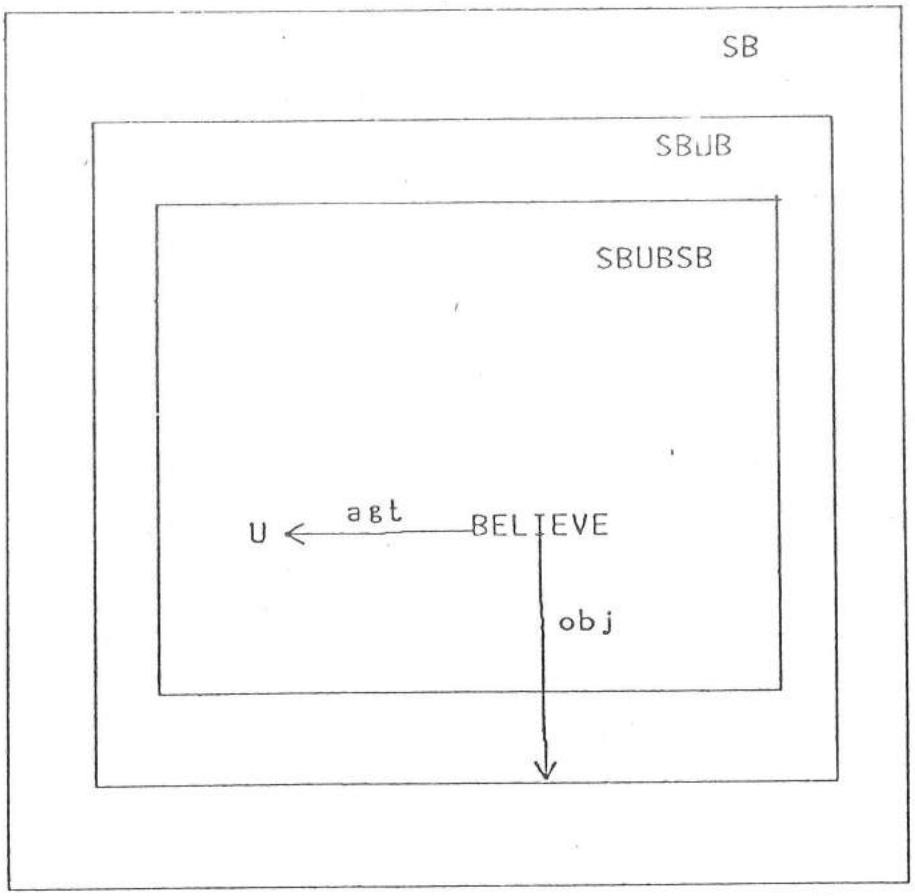

FIGURE 8a
MUTUAL BELIEF MEMORY STRUCTURE

This memory structure is the same as our previous one except that the last space, SBUBSB, contains a proposition that says that the user's belief space in SBUBSB (i.e., SBUBSBUB) is space SBUB! This loop now provides the finite representation for the unbounded knowledge that we needed. (Spaces SBUB and SBUBSB, those in the loop, will be termed "mutual belief spaces".)

Notice that space SB is not included in the loop since if it were, OSCAR would be assuming that the user knew everything about OSCAR's beliefs. Such a situation would never lead to speech acts directed at the user since the goals of such acts would already be achieved because of the user's omniscience. In general, whenever we create mutual

beliefs, we must take care not to make previously private beliefs into mutual ones.

The above representation thus states that every even (resp. odd, after the first) level is the same as the previous even (resp. odd) one. Also notice that SB => SBSB, etc., still holds. Thus, once we create such a memory structure, it should be clear that any combination of SB's and UB's can be handled.

Now, recall the argument about the Earth's shape where you, the user, said it was round and OSCAR said it was flat. We can reflect the effects of those statements in the following way:

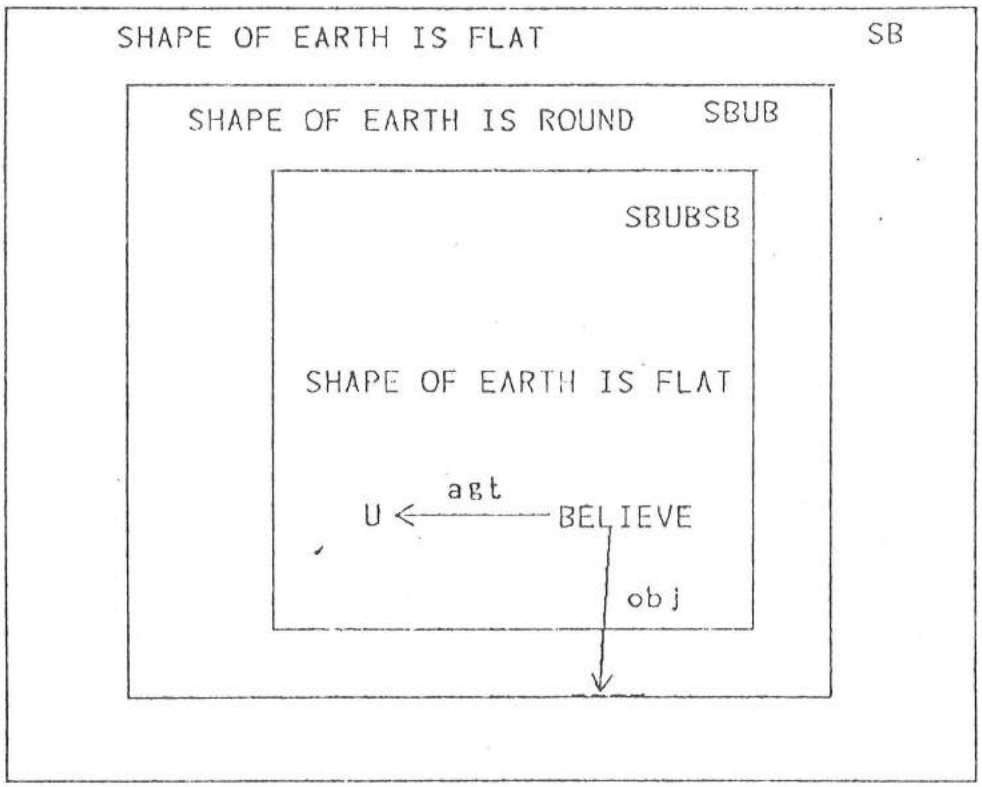

FIGURE 8b

A DISAGREEMENT

If we now test if SBUBSBUB(SHAPE OF EARTH IS ROUND), our answer will be TRUE since we are actually looking in space SBUB, where that fact was asserted. If instead we assume OSCAR believed the Earth were round but it decided, for the sake of argument, to say that the Earth were flat, then Figure 8c would represent OSCAR's memory.

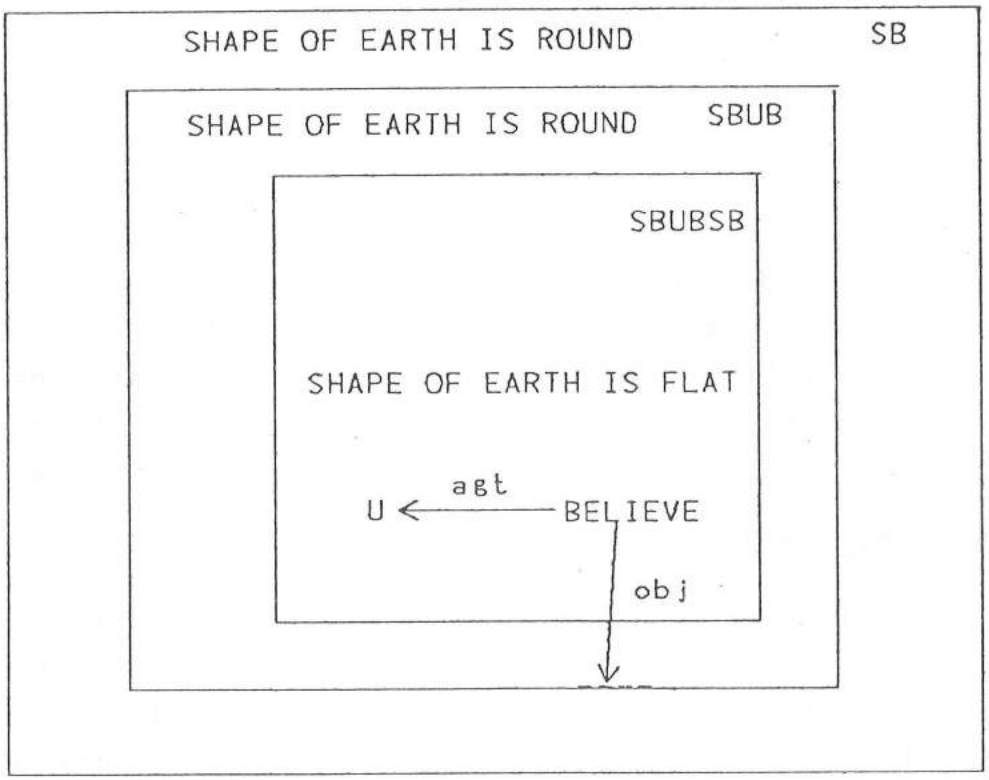

FIGURE 8c
OSCAR PLAYING DEVIL'S ADVOCATE

Any mutual beliefs that are acquired by direct communication with the user are placed in SBUB, while any mutual beliefs that OSCAR believes resulted from its utterances would appear in SBUBSB.

8.4 Operations on MUTUAL-BELIEF and BELIEVE

The introduction of this new memory structure requires that we revise the manipulation operations for BELIEVE in addition to defining those of MUTUAL_BELIEF. The testing algorithm for BELIEVE can remain in its earlier form -- the new memory structure will be transparent to it since the algorithm will simply follow BELIEVE propositions as long as necessary.

8.4.1 Testing MUTUAL-BELIEF

By expanding MUTUAL_BELIEF into the corresponding set of nested beliefs, it should be quite obvious how OSCAR would test to see if some proposition were mutually believed. OSCAR would simply look in each belief space, stopping if the proposition were believed false or if no information were found, until it came upon a belief space that contained the one it had just seen. This belief space loop would indicate that all the rest of the beliefs were the same.

8.4.2 Adding MUTUAL-BELIEF

The algorithm for asserting MUTUAL_BELIEF would be very simple if it had only to worry about mutual belief of (at most) singly embedded belief propositions (e.g., MUTUAL_BELIEF(USER,SYSTEM,SYSTEM BELIEVE P), abbreviated "MB(U,S,SB(P))"). However, P could be an arbitrarily nested BELIEVE proposition. To represent these beliefs, OSCAR may have to create new memory levels. As will be seen below, even a simple addition of a non-mutual belief may cause a similar expansion.

There are three cases to be considered. The first and simplest case is one of asserting MB(U,S,P), where P is some proposition not containing a BELIEVE for U or S, into a memory structure configured as in Figure 8a. Clearly, it is sufficient to assert that U BELIEVE P and U BELIEVE S BELIEVE P, within the existing memory structure.

The second case arises when asserting MB(U,S,P) into a memory structure that does not have a loop -- i.e., one where there are no mutual beliefs present. The proposition P must be added into all extant memory levels (of S and U, of course), and two additional spaces must be created to complete the mutual belief. For instance, starting with just space SB, the assertion of any mutual beliefs between the system and the user would result in the memory structure of Figure 8a.

The third case occurs when the proposition that is to be mutually believed is itself a BELIEVE proposition, as was the situation in the argument above. It was mentioned that we cannot let non-mutual beliefs be added into mutual-belief spaces. Consequently, the adding program for BELIEVE must be sensitive to whether or not it is examining a mutual belief space and whether or not it has been called via MUTUAL_BELIEF.TOADD since it would then be allowed to add into mutual-belief spaces. The basis for the new "to add" of BELIEVE is as follows:

8.4.2.1 A New "To Add" for BELIEVE

If BELIEVE.TOADD is called from MUTUAL_BELIEF.TOADD, then, BELIEVE.TOADD is allowed to add into either of the two mutual-belief spaces. Memory will only be "extended" when "looping" occurs -- i.e., when the new belief space contains the old.

Otherwise, memory must be extended when BELIEVE.TOADD is about to add PROP into the first mutual-belief space ("MB1"). After extending, then, since space MB1 is no longer a mutual-belief space, PROP may be added into it. A multiply embedded BELIEVE proposition may cause a number of such extensions.

To illustrate some of the changes that the adding program for BELIEVE needs to produce, consider the addition

of UBSBUB(T) into the memory structure of Figure 8d. The resulting memory structure is that of Figure 8e.

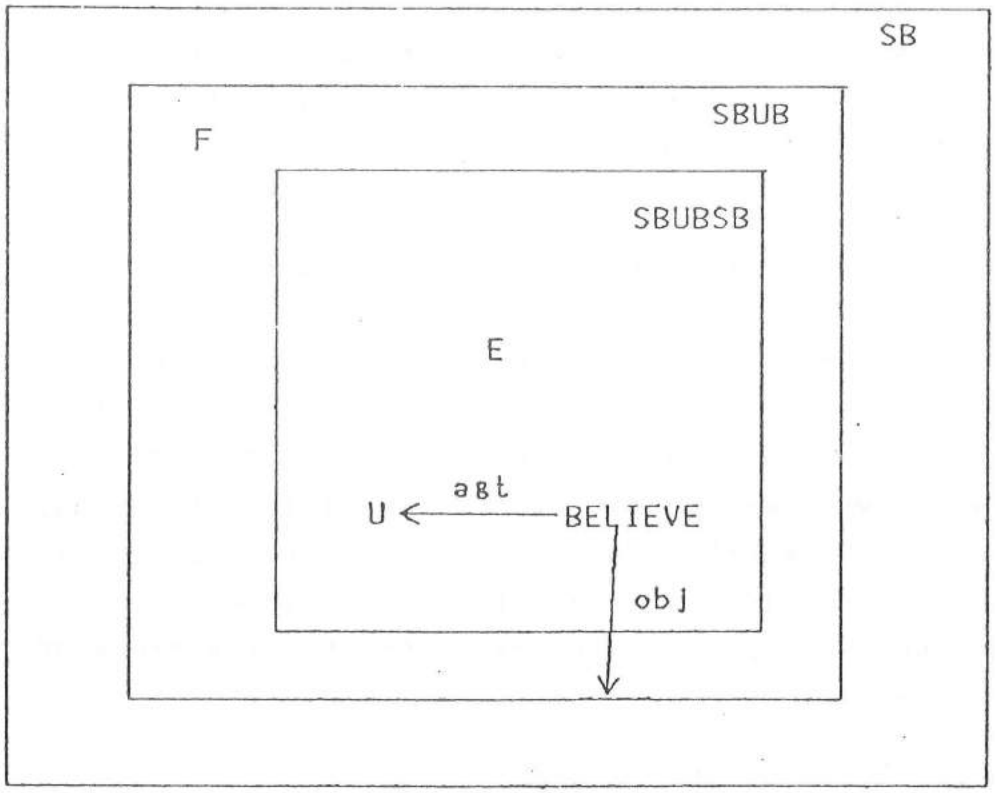

FIGURE 8d
BEFORE ADDING A NON-MUTUAL BELIEF

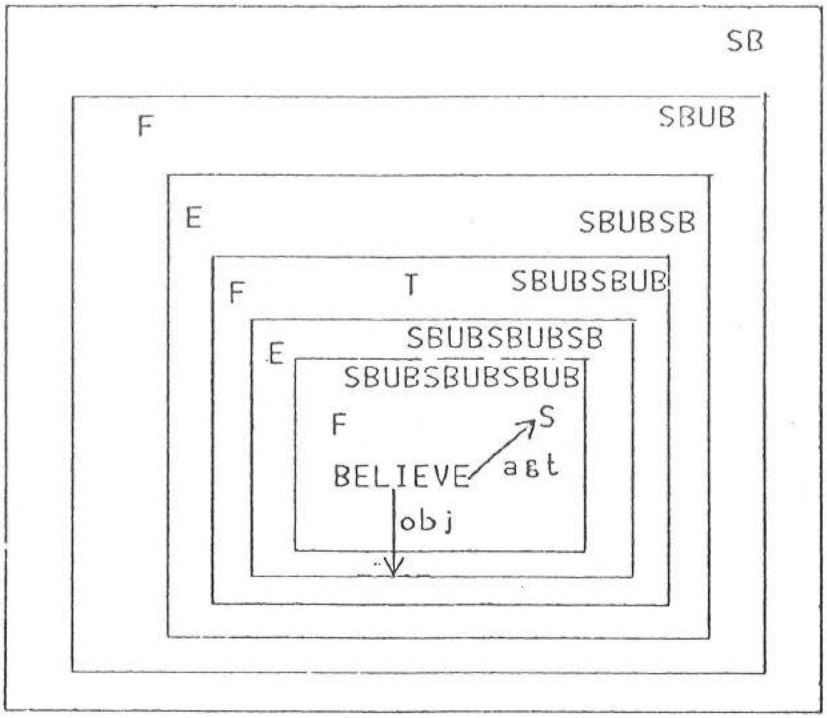

FIGURE 8e

AFTER ADDING A NON-MUTUAL BELIEF

F and E stand for the respective contents of spaces SBUB and SBUBSB. They are placed in the newly created spaces when memory is extended to maintain all mutual-beliefs that existed prior to the insertion of the non-mutual belief. Note that the proposition T is <u>not</u> mutually believed (where T does not begin with U BELIEVE).

If we asserted MB(U,S,UBSBUB(T)) into the memory structure of Figure 8d, we would obtain:

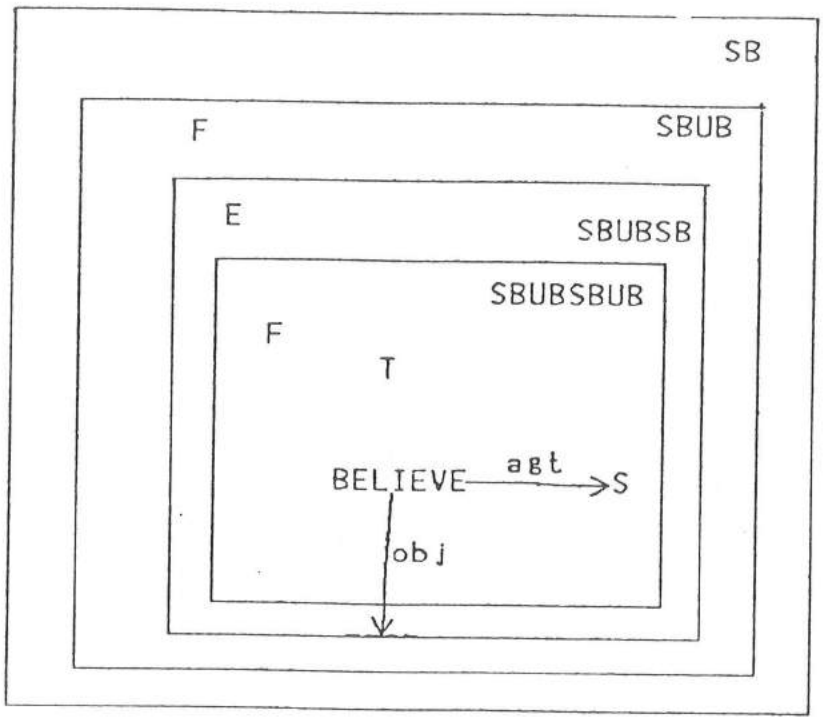

FIGURE 8f

RESULT OF ADDING MB(U,S,UBSBUB(T))

A comparison of Figures 8e and 8f yields the observation that mutually believing UBSBUB(T) requires two less levels of memory than simply believing UBSBUB(T). The intuition behind this perhaps surprising fact is that the addition of a "private" belief into a memory containing mutual beliefs requires that the mutually believed spaces be further embedded so as not to include non-mutual beliefs.[3]

Algorithms have been written for the new assertion and testing programs for BELIEVE and MUTUAL_BELIEF, but they have not been included here for the sake of brevity.

Now that we can represent mutual belief, we can apply the concept to the problem that originally motivated its use. The speech acts REQUEST and INFORM will be redefined

to include MUTUAL_BELIEF as effects and hence all information acquired by communication will be shared knowledge.

8.5 Redefining REQUEST and INFORM

The definition of INFORM will now be as follows:

```
define inform as event
CASEFRAME:    (agt := a person) inform (recip := a person)
                   that (obj := a proposition)
LOCALS:  x

EFFECTS: mutual_belief between recip and agt that
              quant of x such that
                   agt believe obj
CANDO.PR:     agt believe
                   quant of x such that
                             obj
WANT.PR: std
```

.
.
.

The first level of expansion of the effects of INFORM is identical to the old definition. Notice also that the pctential existential quantifier matched by the QUANT pattern will remain "stationary" while the levels of belief are added to the left, as in SBUB ∃xSB(P(x)). A stronger effect for an INFORM, that would imply the above version, would have the existential quantifier always before the first belief, thus giving us beliefs like: ∃x SBUBSB(P(x))

REQUESTs do not use the quantifier and thus are simpler, viz.:

```
define request as event
CASEFRAME:     (agt := a person) request (recip := a person)
                    to do (act := an event)
EFFECTS: mutual_belief between recip and agt that
              agt believe
                   agt want act
WANT.PR: std
```

:

Again, the first level of the mutual belief expansion is exactly the old version -- RECIP BELIEVE AGT BELIEVE AGT WANT ACT.

8.6 Using MUTUAL-BELIEF

The major difficulty caused by using MUTUAL_BELIEF in the effects of REQUEST and INFORM is that matching, during planning, can become more complex. The matcher must be prepared to try a number of different alternatives corresponding to the expansions of MUTUAL_BELIEF.[4] For instance, the pattern MB(RECIP,AGT,AGT BELIEVE P) (the effect of an INFORM) would match the (unlikely) goal SBUBSBUBSBUB(Q) as follows:

RECIP <- S

AGT <- U

P <- Q, or SBUB(Q), or SBUBSBUB(Q).

We probably would want the smallest possible proposition to be bound to P since stating the larger ones may cause the hearer, and thus the system's model of the hearer, to reorganize memory.

OSCAR would plan with MUTUAL_BELIEF by proceeding in exactly the same manner as before, given an appropriately extended matching algorithm. However, while a single belief is often the end goal (e.g., that the user believe something), the speech acts used to achieve these beliefs produce other beliefs as well. Thus, the same speech act could be used, if we could ever devise such a subtle

program, to plan to produce fourth-level beliefs rather than direct effects. Such planning is beyond our current theoretical understanding of the processes of communication. I conjecture, however, that mutual belief is crucial to the planning of referring expressions. The reasons for this conjecture are discussed in the next chapter.

8.7 Notes

1. Mutual awareness also occurs in face-to-face situations. More generally, it arises in situations of common experience, such as when two people are seated facing each other across a table that has a cup on it. It is then mutually believed (without there having been any overt communication) that there is a cup on the table since the people are normal humans with open eyes who cannot help but see each other and the cup. This situation, described in [Schiffer 1972], is simpler than one of communication since, for the latter case, it is beliefs that are mutually believed.

2. This definition of mutual belief differs from Schiffer's [1972] and Lewis' [1969] by taking a particular point of view -- X's, in my definition. Their attempt at a more neutral characterization would reduce to a variant of the formulation above when we discuss a speaker's beliefs or intentions about mutual-belief, as occurs in describing speech acts.

3. Viewing OSCAR as a psychological model, one might get some indication of why people get flustered after two or three levels of belief embedding by noting the amount of work needed to keep mutual and non-mutual beliefs separate. It appears to be easier to acquire a particular (embedded) belief by mutually believing lesser embedded ones than by creating an assertion of the embedded belief itself.

4. As an implementation detail, classes should have their own "to match" operations to allow specialized matches, such as that of MUTUAL_BELIEF, to easily take place.

CHAPTER 9

EXTENSIONS AND APPLICATIONS

Contents:

9.0 Introduction

This chapter presents a number of directions in which the methodology of this thesis can be extended towards our ideal model of conversation. Specifically, I discuss some capabilities OSCAR might need in planning to refer, to issue other speech acts, and to ask for help. This chapter also compares the thesis to other research in AI, and then presents potential areas for practical application of OSCAR-like techniques.

9.1 Connections to the Surface -- Planning Reference

The speech acts produced by OSCAR are still far removed from the actual generation of surface utterances. While there do exist computational methods for generating utterances from the semantic representations that OSCAR produces, simply including these methods into our system would do little for advancing our theoretical understanding of the language generation process. A goal for extending the view of language pursued here is to discover what

aspects of the production of an utterance are intentional. We want to know how the various kinds of effects a speaker may wish to have on his audience influence the form of the utterance.

With an eye towards such a goal, I suggest that a critical problem for this approach is the planning of referring expressions. The general framework in which to view this problem is the one advocated by Strawson [1950], Searle [1969], and others -- namely that speakers, not expressions, refer. Speakers use expressions to produce certain effects. Exactly what those effects are is currently a mystery.

A speaker's (illocutionary) goal in sincere and truthful communication is to produce correspondence between the speaker's beliefs and his model of the hearer's model of them. He attempts to bring about these correspondences partly by planning his references. As with other speech acts, the reference act is either successful or not in its function in communication.

There are many ways that speakers use referring expressions. The discussion here will only concentrate on situations where the speaker believes the hearer has a corresponding mental object to serve as the referent. In other words, we shall only deal here with referential definite and specific indefinite noun phrases.

Within this framework, there are two important consequences of viewing reference as an act: the emphasis to be placed on the speaker, and the role of the recognition of intention in the success of such an act. The problem of referring, regarded from the speaker's point of view, is this: the speaker has some mental object about which he wishes to say something. His goal is to devise a way to get the hearer to focus on the corresponding mental object for some later purpose (for instance, so that he may say

something new about it). Since the speaker only has access to his model of the hearer (that includes the hearer's model of him), he need only concern himself with constructing an expression that refers to the desired object in that model.

The role of intention can be seen by noting that a hearer understands a reference based on that to which he thinks the speaker intended to refer. Let us consider two examples of this point. A friend, who was standing with me in front of an art gallery, asked "Who is the person in the second painting?" Now, my friend was standing in such a position that I believed she was unable to see the first painting on the wall. When I said "Margaret Atwood", I was referring to the person in what I believed to be the third painting.

In reflecting upon my actions, I realized that I had interpreted her phrase "the second painting" with respect to my model of her. (Afterwards she informed me that I had indeed referred to the painting that she had meant.) If my interpretation strategy were to evaluate someone's referring expression with respect to my model of them only when that expression failed to pick out a unique referent among my mental objects, then I would have misinterpreted her expression (since the phrase did have reference for me.)

A similar example of Donnellan's [1966] is a case where the phrase, interpreted literally, does not have a referent and yet communication succeeds. At a party, someone asks you "Who is the person holding the martini?" You see a person holding a martini glass but you believe that person, a tee-totaller, is holding a glass filled with water. Nevertheless, you reply "John Smith". It does not matter that, as far as you are concerned, no one is holding a martini -- you identify the referent as that to which you believe the speaker intended to refer.

While as hearers we consider a speaker's intentions, as speakers, we clearly refer in a fashion designed to ensure successful communication. This involves considering our hearer's capabilities for understanding. (For instance, we might describe something to a small child in a different way than we would describe it to an adult.) Thus, we get our usual problem of intending that someone recognize our intention (that they recognize our intention...).

Speakers may not always go to such great lengths to avoid miscommunication. Since they can, however, we need to model their capacity for doing so. To model the act of referring, we shall need to break out of the recognition of intention spiral. I shall suggest (and only that) a possible escape route.

9.1.1 Reference and Mutual Belief

In designing a referential definite noun phrase, the speaker chooses a description that, when evaluated context-sensitively in the hearer model, yields the appropriate mental object. The concept of mutual belief may be of use in formalizing this process. The speaker believes his description will be successful if he believes that it is mutually believed between hearer and speaker that the description refers to the desired mental object. Assume that OSCAR is the speaker and that it wants to focus the hearer's attention on his analogue of its mental object G001. Using only its model of the hearer, OSCAR would want to simulate the hearer's focussing upon G001 in SBUB. OSCAR might:

1. Choose a description believed true of G001. (Of course, the crux of the problem is to determine which description to choose and in which space to choose it.)

2. Determine if it is mutually believed between the user (the hearer) and OSCAR that the expression refers

uniquely to G001. If so, then it has an appropriate expression that would need surface realization.

3. If not, choose a different expression based upon the original one and the kind of failure. Iterate to Step 2 until an appropriate expression is found.

We can represent Step 2 by: (Speaker = "Sp", Hearer = "H")

TEST(MUTUAL_BELIEF(H,Sp,EQ(F(t1,...,tn),G001))) in SpB.

For this test to work, the expression F(t1,...tn) must be understandable to the user, and the name G001 must map onto the same object in all spaces.[1]

However, this formulation is perhaps too strong. From the art gallery example, we can see that a hearer may interpret a reference in his model of the speaker. Then, as speakers who know about being hearers, we may only need to formulate mutual belief to produce hearer beliefs about us. In other words, as speakers we may only want to insist that:

TEST(MUTUAL_BELIEF(H,Sp,
Sp BELIEVE(EQ(F(t1,...tn),G001))) in SpB be true.

With OSCAR as the speaker, and employing its standard memory structure, the evaluation of the above would take place in SBUBSB. For the previous formulation, it would have used space SBUB. Notice that this new condition allows for OSCAR to believe that the user does not know what the expression (e.g., 'GXXYYZ') refers to; OSCAR need only believe that the user thinks that when OSCAR uses this expression, it is referring to a particular (mental) object (say G001) in SBUB.

In fact, OSCAR does not itself have to know the referent of the expression -- it can use the expression provided it believes that the user thinks it knows what it is talking about. Hence, the expression need only have reference in

space SBUBSB. However, what the system predicates about that object ought to be information that it believes; information in space SB. In this fashion, OSCAR would plan to produce correspondence between SBUBSB and SB.

Let us reconsider the two examples above. If OSCAR were the hearer in the art gallery example, the retrieval of the value of the expression would take place in SBUBSBUB, since the hearer's version of Step 2 is:

H BELIEVE (RETRIEVE (MUTUAL_BELIEF(H,Sp,Sp BELIEVE ("The second painting")) in SpB))

But OSCAR's usual mutual belief memory structure would equate SBUBSBUB with SBUB. Thus, testing is done at a reasonable depth. Similarly, if OSCAR were the hearer in the martini example, the mutual belief memory structure would cause interpretation of "the man with the martini" to take place in space SBUB as well.

The mutual belief condition might be a precondition to a REFER act, while focussing might be its effect. (See [Grosz 1977] for an interesting approach to the problem of focus.) I view the planning of the REFER act as proceeding along the lines of a NOAH plan; the planning algorithm would expand a speech act to a new level of detail, which incorporates the REFER act. This is in line with Austin's claim that it is by means of the locutionary acts that we perform the illocutionary ones. Future research should investigate the use of a NOAH-style methodology in supporting Austin's claim.

The proto-algorithm above provides for some of the flexibility that we want for our reference act. Of course, there are many other cases of referring that are not covered by it. For instance, a speaker might frequently intend for his hearer to "create" a new mental object, without his having to know what the referent of the expression is (see Chapter 7). Thus, the speaker might plan for SpBHB¬∃xSpB

(P(x)). In other words, the speaker is planning for the hearer to think that there is something that the speaker thinks has property P. Whether the speaker uses a definite or indefinite noun phrase to realize such a logical form might depend upon uniqueness conditions on P in space SpBHBSpB. Further research might proceed by trying to develop a unified approach to the planning of both specific and non-specific references.

Note also that the reference generation process needs access to a description interpretation (or matching) process. Bobrow and Winograd [1977] have suggested that matching should be based on differential accessibility of objects. Given different amounts of processing resources, the matching algorithm may find that the same description can refer uniquely, can be ambiguous, or does not refer at all. They posit that differential accessibility is part of the "cognitive hardware", and as such would radically affect the nature of our cognitive models. This conception of resource-limited processing may lead to a different definition of the term "semantics" for natural language.

These ideas are only intended to be suggestive of how a system like OSCAR could plan to refer. Definitive solutions to this question will have to solve long-standing philosophical problems. We should not, however, shy away from these questions simply because philosophers have argued about them for so long. We have some interesting new tools that can be applied, and the potential payoff is high.

9.2 Planning Other Speech Acts

OSCAR's plans currently only use REQUEST and INFORM speech acts. One area where OSCAR needs extensions is in the planning of other speech acts in the same classes (directive and representative, respectively). In additon, we might want OSCAR to plan speech acts from the other three classes -- commissives (e.g., promises), expressives (e.g.,

apologies), and declarations (e.g., betting, marrying, naming). I shall briefly discuss the possibilities for such extensions.

9.2.1 Directives and Representatives

Directive speech acts that a process-model of language use should handle include suggestions and commands. A speaker, before issuing a command, must believe he is in a position of authority over the hearer. The simplistic way of dealing with the authority precondition is to have a fixed scale of who has authority over whom. A more sophisticated method would deal with the likely consequences of disobeying the command.

Such an act (using the simplistic authority measure) could easily be incorporated into our model of speech act use. While from a practical standpoint, it is not clear whether we would want machines to issue command speech acts, a computer conversant would need a definition of the command speech act in order to recognize such acts.

Before making a suggestion, a speaker must determine whether the act to be suggested is to the hearer's benefit. However, it is often not clear who is the beneficiary of an act since, in cooperative situations, speaker and hearer are often involved in each other's plans. For instance, if your plan is "part" of mine (e.g., I'm counting on your doing something), can I then suggest that you do something that, while perhaps beneficial to you in the short-run, is ultimately beneficial to me? What if the long-range gain on my part is at your expense? Before modelling suggestions, we need to understand more about the concept of benefit, and how it relates to the successful performance of suggestions.

Representative acts, other than inform, that a process-model of language use should handle include deny and lie. The only differences among these acts that we can currently model in OSCAR are differences in the preconditions. Thus,

for DENY, if the effect is that ("SP" = speaker, "H" = hearer) H BELIEVE SP BELIEVE P, the preconditions would be: SP BELIEVE P, and SP BELIEVE H BELIEVE ¬P. The precondition for LIE would simply be SP BELIEVE ¬P, while its effect would be the same as that of DENY. Both of these acts could be included easily in the current system. (We probably would not want a program to plan LIEs, but we might want it to know one when it "sees" one.)

There are other representatives (e.g., insist) that communicate degrees of the speaker's belief. Any OSCAR-like system ought to handle beliefs of different credibilities, related to their justification and origin. This is an important problem, but I view it as secondary for the time being.

9.2.2 Other Classes of Speech Acts

The most important remaining class of speech acts is the commissive class. Again, the preconditions differentiate members of the class from one another. For instance, before OSCAR can issue a promise, it has to plan to make the hearer aware that it intends to do some action, and that it is committed to doing it. In addition, OSCAR must ensure that the act it is promising to do is one the hearer would want done. Clearly, the concept of benefit arises again.

The reason for planning a promise involves one's indicating that one is committed to a future course of action. However, I have yet to develop plans where commitment and obligation play a role. Before worrying about planning commissive speech acts, we should see in what ways its effects would be useful in forming plans.

The other classes are of marginal interest for a model of the instrumental use of language. Expressive acts indicate the speaker's psychological state -- i.e., his feelings. I cannot see how OSCAR could be apologetic,

guilty, regretful, or remorseful. I was willing to say OSCAR "believed" or "wanted" because, in some sense, it acted as if it did (in its reasoning). The best one could hope for would be for the system to say "I thank you", "I apologize" or "I regret". However, the system would not have any "feelings" behind such a statements. Thus it would seem inappropriate for an OSCAR-like system to reason about issuing such speech acts.

While we could attempt to model how certain feelings arise and why they lead to speech acts, I believe such a naive psychological model would be a gross oversimplification. Feelings may be only marginally related to reasoning. An attempt to model the generation of expressive speech acts by the methodology of this thesis would be an excursion outside of our domain of applicability -- purposeful dialogues. Hence, I make no claims that one reasons about the speech acts in this class and thus utters them instrumentally.

The last class, declarations, are an odd sort. In general, tc perform such actions, one needs extra-linguistic institutions to attribute certain properties to the speaking situation (and the dialogue participants) -- the institutions confer the right to issue such speech acts. There is no reason, in principle, why one could not give special status to a machine and have it make something the case by saying so (e.g., naming a ship). One might, however, question the judgement and wisdom of institutions that allow (or cause) this to happen.

On the whole, practical systems would have little use for declarations. At the theoretical level, we would want to discover how (if at all) we would plan such acts.

9.3 Helping

One of the primary reasons that I suggested was behind OSCAR's inability to plan a number of reasonable speech acts was that it did not have an action HELP. This section will make some suggestions for what such an act might look like.

9.3.1 Being Helpful

Recall the stockboy example of Chapter 1. The shopper stated a goal and the stockboy replied by: stating the obstacle (there was no Brillo), suggesting a functional alternative to Brillo, and then stating information that would be required in proceeding along the alternate path -- using SOS. The reason why the stockboy made a suggestion involved his knowledge of being helpful. We want to show how the action HELP could lead to these speech acts.

Let us assume OSCAR knows what the user's plan is. As a possible strategy for helping someone to achieve a given goal state,

- OSCAR could simulate the user's plan with its own knowledge
- If the simulation indicates that user's goal is achievable, OSCAR could attempt to correct his beliefs, based on where his plan deviates from its own.
- If the simulation of the user's plan, fails at the goal in question, then OSCAR could plan to achieve the goal. This may involve simply finding other acts to achieve it -- acts of which the user was perhaps unaware or could not perform. (For instance, the stockboy could have looked for more Brillo in the back room.) If no such acts were available, then OSCAR could try to find an alternate way of achieving a higher-level goal (scouring pots, in our example) and getting the user to want a new subgoal to this higher-level goal (e.g., wanting to have SOS).

This strategy would then propose a new goal regarding what the user is to want. Such a goal would be achievable via a directive; SUGGEST may be appropriate here since the user is the ultimate beneficiary of the plan.

There are many points in the "algorithm" where one could be more or less helpful. One could tell the user what needs to be done to achieve the goal or, at the other end of the scale, one could do it one's self (e.g., getting SOS for him). In this latter case, the helper is, in effect, "taking over" the plan. This may occur by choice or because there is no alternative. The following (actual) conversation illustrates such a move:

Shopper (me): "Where are the chuck steaks you advertised at 88¢ per pound?"

Butcher: "How many do you want?"

After answering the original question with a question, the butcher (on hearing my response "three") went into the back room and returned with the steaks. Unlike most questions, he no longer had to answer my original question, since it had become pointless. He appears to have reasoned that a correct reply to my question would not convey information that would enable my plan to succeed. Hence, he took over the plan and asked me a new question so that he could proceed with his plan to help me.

The essence of being helpful for these two examples is inferring someone's plans and then checking them against one's own knowledge. While we may lack plan-recognition techniques, I anticipate that we could begin to experiment with helping strategies, since we have the tools for simulating the user's planning (given knowledge of his goal) and for simulating the execution of the resulting plan. Future research should investigate helping strategies to see how they influence the production of goals leading to speech acts.

9.3.2 Asking for Help

If we treat HELP as an act in the usual fashion, then we may find that numerous lines of dialogues occur to satisfy its preconditions. (as in "I'm looking for Brillo). The following is a partial definition, which might be employed in an OSCAR-like system.

```
define help as event
CASEFRAME:    (helper := a person) help (helpee := a person)
              achieve (goal := a proposition) in the face of
              (obstacle := a proposition)

LOCALS:  x
CANDO.PR:     helper believe
                helpee believe
                   helpee want goal
                   and
              helper believe
                helpee believe obstacle
                   and
                   ∃x[ (helper believe
                        planof helpee to achieve goal is x)
                             and
                        (helper believe
                          helpee believe
                             obstacle block x)  ] ) ²
WANT.PR: std.
EFFECTS: ??   goal?
BODY:    "helping strategies"
```

The CANDO.PR first states that the helper needs to know what the helpee wants and what his problem is. When requesting HELP, one would have to state both the goal and the obstacle, leading to such utterances as "I want to come in but the door is locked" (cf. sentences 5 and 6 of Chapter 5). The next two conditions state that the helper has to know what the helpee's plan is and that he must believe the

helpee believes the obstacle blocks that plan. The person requesting help would then have to plan for the helper's being aware of his plan. One way to achieve this is to explicitly state the plan ("I'm trying to...") rather than relying solely on a recognition algorithm.

Alternatively, one might want to propose a RECOGNIZE-PLAN act that would produce a PLANOF proposition linking a person, his goal, and his plan to achieve that goal. The preconditions to such a plan-recognition act would be extremely interesting in that they would state conditions that would (hopefully) guarantee that the agent could recognize the intended plan. The helpee could then plan to make those conditions hold. (Of course, having such an act is crucial to modelling the understanding of speech acts.) Communication might be facilitated by a speaker's planning to establish the preconditions of this recognition act. This would allow the speaker to determine if the hearer could recognize his intention, without the speaker's having to engage in a recursive simulation.

By having the standard want precondition, OSCAR could plan to request that the user perform the act HELP. In circumstances where the user, or OSCAR for that matter, is presumed to be helpful, the want precondition on the act would always be true. Thus, an explicit request for HELP need not take place -- only the preconditions need to be satisfied.

Before OSCAR could ask for help, it would have to realize when helping is appropriate. OSCAR plans acts via their effects. However, I have been unable to determine an adequate set of effects for this act. One can regard HELP as being appropriate to any goal, and hence the (possible) effect would be the proposition GOAL. Getting someone to HELP would then become a "last resort" means of achieving anything.[3] This can be easily implemented in the current system, allowing OSCAR to plan sentences 5 and 6.

9.4 Related Work Within Artificial Intelligence

Apart from previously mentioned research that was related to various specific aspects of this thesis, there are a number of researchers whose goals are perhaps more directly aligned with mine. One can characterize many of them as dealing with, to coin a phrase, "Stereotypical conversation". In this category I place the work of Power [1974], Bruce [1975a], Mann et al. [1977], and Horrigan [1977]. I shall discuss how this overall approach, and that of Moore [1977], compares with mine.

9.4.1 Stereotyped Conversation

This approach to the modelling of dialogue is based on viewing conversations as exhibiting regular patterns. The terms for these patterns range from "game" (Power, Mann et. al.), and "social-action paradigm" (Bruce) to "dialogue grammar" (Horrigan). (Schank and Abelson's [1975] scripts are similar, though they do not refer to the speech act structure of dialogue.) For instance, adjacent "moves" by the participants in a conversation might be question-answer or state-acknowledge pairs (where for the latter, the speaker of the acknowledge speech act communicates that he has heard the previous utterance correctly).

Power and Mann et al. are concerned with, among other things, modelling how a speaker begins a dialogue game. For Power's system, the "dialogue" participants (simulated robots) explicitly named the game they wanted to play. The games formed rigid patterns for the conduct of the dialogue. However, the participants had no idea why the game was structured the way it was, and why they were uttering something. They only had minimal knowledge of the beliefs of the others.

Mann et al., in proposing a system to model dialogue (not engage in one), suggest ways of invoking the helping

game via inferences from utterances. Once invoked, the preconditions of the game are then imputed as beliefs of the dialogue participants. The helping game states very specific goals for who says what to whom, in the course of a helping interaction. In contrast, the view of helping proposed here would emphasize the reasons for the helper's utterances -- the strategies that lead him to propose the new goals from which the utterances were planned.

Bruce's system was intended to function as the pragmatics component of a functional speech understanding system. His social-action paradigms were to generate expectations about the next utterance. These educated guesses would be used in aiding semantic, syntactic, and lexical processes in the decoding and understanding of a speech signal. Bruce ([Bruce 1975a] and [Bruce 1975b]) suggests a system organization and philosophy that is similar to the one presented in this thesis.

Horrigan collected nearly one hundred dialogues between passengers and an information-booth clerk. Her program "parsed" about two-thirds of them with a "dialogue-grammar" developed from a small subset of the corpus. Given one utterance, her program predicted the next move (in type and content).

Clearly, there do exist situations where such a grammar-based approach is appropriate. Such models of common interactions could be useful in a number of ways. In appropriate circumstances, they may produce accurate predictions of the ways in which the conversation may proceed, and thus could aid in interpreting sentences as speech acts.

Games can be viewed simply as summaries of successful plans. While using them may be more efficient than either developing or recognizing plans, they need to be augmented with more general reasoning capabilities (as Schank and

Abelson [1975] note.) All of these researchers acknowledge the inflexibility of their schemes for handling utterances that fall outside of the scope of the dialogue game. They observe that what is needed are mechanisms for developing plans using speech acts and for recognizing speech acts by inferring a speaker's plans. In addition, they also suggest that these processes ought to be sensitive to a model of the user's beliefs and goals.

None of these papers, however, present detailed descriptions of intention-driven systems that are sensitive to their user's beliefs. (Mann et al. are constructing such a system; details should be forthcoming.) Other research is currently in progress (see, for instance, [Bruce 1977], [Schmidt and Sridharan 1977] and [Allen forthcoming]) on plan-recognition algorithms that could be used in identifying speech acts.

9.4.2 Comparison with Moore's Approach

Moore [1977] is trying to develop a formalism that can enable a program to reason about a person's knowledge and actions. He is interested in modelling how one's having certain knowledge enables one to perform actions, and how knowledge is affected by the actions that one performs. He is integrating, on the one hand, a semantics for a modal logic of "know" based on the formalisms of [Kripke 1963] and [Hintikka 1963], and on the other, a formalism for action described in [McCarthy and Hayes 1969].

The form of his model theory follows Kripke's [1963] possible worlds semantics for modal logics. In general, one determines the meanings of formulas not just with respect to the actual state of the world, but also with respect to alternative or possible states. For Moore, these become possible states of knowledge. Hence, the meaning of the statement that "A Knows P" then becomes "P is true in all possible worlds according to what A knows."

McCarthy and Hayes suggest that a logic of actions can be developed by using situations. The result of an action, executed in a situation in which it is applicable, is a new situation. Situations are viewed by Moore as possible worlds that could result from the execution of some action.

Moore defines CAN(A1,P1), meaning agent A1 can bring about condition P1, in terms of A1's knowing of some action that will result in a situation in which P1 is true. He illustrates his approach by axiomatizing the world of combination safes, and by showing that, provided that John knows what the combination of the safe is, and that he knows he is at the safe, John CAN get the safe open.

Moore has formalized the semantics of the modal logic in a first-order meta-language, where some of his axioms reflect the model structure. His intention, I gather, is to be able to prove mechanically object language formulas of knowledge and action by standard first-order manipulations of their (semantic) translations.

His approach is similar in a number of ways to the one developed in this thesis, but there are some crucial differences. I have already mentioned (in Chapter 4) that his discussion of the expressive power of the "data base" way of modelling knowledge is not to the point (though there are formal semantic problems with that approach, as mentioned earlier). I will discuss the applicability of his methodology to the modelling of people with respect to his handling of knowledge and action.

9.4.2.1 Knowledge

Moore's relation between possible worlds of knowledge is reflexive -- each world is related to itself. This property implies that everything that is known is true. Such a formalism would then have great difficulty representing John's knowledge of a disagreement between himself and someone else; i.e., that John knows P, and knows that Fred

knows ¬P. Clearly, one would need a concept of belief to deal with disagreements.

The treatment of "know" that appears to be useful for a program (and has been proposed by many philosophers) is one where "know" is defined in terms of "believe"; one compares a person's beliefs (and their justifications) to those of another person. With such a concept, OSCAR could say "John knows" or "Bill believes John Knows", but not "I know".

The possible worlds with which Moore is concerned are those of common knowledge (what everyone knows, and knows everyone knows, etc.) But, few facts are known by everyone. Moreover, there are few facts that are known even by every normal adult. Seemingly, one would want to represent common knowledge among a small group -- for instance, the people in this room. But then, only some knowledge is common. Obviously, there is more to what a person knows then common knowledge.

OSCAR employs the concept of mutual belief between two persons, and is able to distinguish between private, shared, and mutual beliefs. Moore might have difficulty in jointly representing both non-mutual knowledge and mutual knowledge that is not common to everyone.

This use of such possible worlds of common knowledge seems to justify his use of rigid designators -- "ordinary" terms that denote the same thing in all possible worlds. (He does not supply an interpretation for terms and thus what "ordinary terms" are is somewhat mysterious.) Since the worlds are of common knowledge, one might argue that all terms are defined and evaluate to the same thing in all possible worlds. Once one leaves the safety and comfort of such possible worlds, however, the rigidity of terms may become a liability in a model of human knowledge. For instance, just because I can take a derivative does not mean that my knowledge of the knowledge of a child should have a

corresponding function. Even in the cases where terms are defined, we would not want to say that they evaluate identically for different people.

Rigidity may also lead to some undesirable formal properties. Under certain reasonable suppositions about Moore's model structure and interpretations of terms, the following (undesirable) implication may be valid:

John Knows (f(t) = g(b) & Ralph Knows [P(f(t)) & Q(g(b))])
=>
John Knows Ralph Knows (f(t) = g(b)),

where f(t) and g(b) are (ordinary?) terms and P and Q are predicates. Assume "=" is treated as identity of domain elements, and f(t) and g(b) are rigid. Also assume the domain of each of Ralph's possible worlds contains the domain of the possible world of John's to which they are related, (as is usually done). In any model satisfying the antecedent, it seems that the consequent will be true, since the terms will be equal in Ralph's worlds. But this is parallel to Quine's Ortcutt example. As shown in (A.1) from section 7.3.2, this is not a valid implication for our model precisely because terms need not denote the same things in all belief contexts. Only a complete formulation of Moore's model theory can settle this issue.

9.4.2.2 Action

Moore's concept CAN relates a person to a condition. CAN(A,P) is true if A knows of some action E that, when executed, results in condition P. This does not say that A can plan to make P true since, while action E may achieve P, one may not be able to prove that A knows that E's preconditions are true. Moore's CAN, unlike OSCAR's CANDO, does not create subgoals to make E applicable.

OSCAR can say that A BELIEVE A CANDO some action provided certain conditions are true. Such conditions are

often discovered through a simulation of someone else's planning. OSCAR can believe that the person can do the action, while believing that that person does not believe he can do it. Under such circumstances, OSCAR would then plan to change that person's beliefs.

Finally, OSCAR's directive speech acts (among others) are planned to influence a person's WANTs. Moore gives no treatment of this concept.

While I have emphasized the differences between the two approaches, there are significant commonalities in the formal semantics and in the problems treated (e.g., quantified propositional attitudes). It will be interesting to see an implementation of his theorem-proving system to determine if it can be extended to reasoning about speech acts.

9.5 Possible Applications of the Techniques

A number of possibilities exist for applying OSCAR-like technology to the development of knowledge-based systems that are oriented towards helping a user perform some task. An example of such a system is one that would act as a personal assistant. It would be an active, helpful partner in an interaction whose goal is known to it (e.g., the user is trying to plan a trip or schedule a course). Such systems need not communicate in natural language, though that would certainly be preferable. Whatever mode of communication such systems employ, they ought to be helpful and intention-driven.

We can envision extending current information systems in this direction. A possible candidate would be an "intelligent" data base management system that would help its user, in pursuit of some known goal, acquire knowledge from its storehouse of facts.

A large-scale integrated data base system maintains a set of facts representing some aspect of the real world. Currently, systems are being built whereby the user need not care about the physical representation of the data base; he need only see its logical organization in terms of data base relations (see [Codd 1970]). Future systems may loosen these restrictions further so that the user need only care about the conceptual relationships instantiated in the data base. Proposed new systems (e.g., [Roussopoulos 1976]) would encode the knowledge describing the conceptual relationships as a semantic network. Thus one could anticipate using OSCAR's memory organization to maintain the conceptual information.

Casual users of such knowledge-based systems may each have a completely different view of the knowledge it contains. One would want such systems to be sensitive to the user's view of its contents and to phrase its responses, in both quantity and level of detail, accordingly. [4]

A potential way to do this is to create different belief spaces for different categories of users (e.g., administrators, secretaries, professors, etc.). For any particular user, certain assumptions can be initially made about his knowledge and intentions. (Such assumptions should, of course, be confirmed.) The user model could then be updated during the course of the interactions.

Such a system could be able to make suggestions and supply extra information when needed, based on its model of the user. For instance, it could display more or less of a certain table, menu, or map depending upon its knowledge of the person with whom it is interacting. (It might do this by giving some indication of the potentially relevant information that it is not showing.) Such a system might then plan its interactions and maintain shared knowledge with the user during the course of the session. I believe the techniques discussed in this thesis could be adapted to

such user-oriented systems that use standard modes of communication.

A second specific application area is to intelligent CAI systems. Collins and Stevens [1977] are investigating teaching strategies for such systems. These strategies could be encoded in OSCAR to make use of its ability to model the beliefs of others. The strategies could then lead to planning teacher/student questions in addition to planning statements, directives, and real questions.

9.6 Notes

1. Since constant symbols need not take the same value in each belief context (according to the formalism), we can get the effect we want by quantifying into the mutual belief, as in

   Test(∃x x = G001 & MB(U,S,EQ(F(t1,...tn),x)) in SB)

2. Planning to achieve a quantified conjunction is beyond OSCAR's capabilities. However, OSCAR would have no difficulty if we made the bound variable a parameter to the act.

3. The problem here is that the user's executing HELP may produce something completely different -- the satisfaction of a higher-level goal, or the removal of an obstacle, etc. After asking for, and receiving help, OSCAR would have to reconsider its overall plan.

4. The current meaning of "view" in the data base management literature pertains to the logical structuring of the data base.

CHAPTER 10

CONCLUDING REMARKS

Austin suggested that there was much to be gained from dealing with illocutionary acts in the same fashion as we would other forms of intentional behavior. His observation led to both the problem and the approach of this thesis.

The point of this research has been to investigate the structures and processes involved in modelling how we decide what to say in purposeful conversation. Our concern has been with showing how a speaker's knowledge of his hearer influences what he says.

The thesis shows that within a restricted class of dialogue situations, we can model the process of deciding what to say as one of problem-solving. Following this, we have shown that the same mechanisms used to plan non-linguistic actions can then include the planning of speech acts. However, these acts place additional demands on the problem-solving processes, not the least of which is the necessity for manipulating statements of beliefs and goals.

The methodology of this research employed a computer system that integrated beliefs, goals, problem-solving, and reasoning, with the use of language. The program did this by planning illocutionary acts when it determined that to achieve its situational goals, it needed to influence someone else's behavior. Formalizing the theory as a program was useful in a number of ways: First, it imposed requirements of precision on the specifications of the various components of the theory. Secondly, it provided a testbed in which to investigate the interactions of those components. In such an environment, one had immediate criteria for failure -- the model simply did not function.

However, just developing a functioning system does not guarantee "success"; we need to be able to learn something from it.

To be more specific, it was found that the definitions of speech acts, adapted from those found in the literature, needed refinement and reorganization. Unlike previous research, the criteria we used for successful speech act definitions were operational ones -- the acts so defined had to be includable in plans.

The definition of INFORM needed to be refined since its preconditions were inappropriate to the planning of questions. The original definitions of REQUEST and INFORM also proved inadequate in another way -- the kinds of plans produced for requesting that someone do action X differed from those produced for stating "I want you to do X". The differences were attributed to an improper association of preconditions to various acts. The process-model brought these failures immediately to light.

In order to present the system's conversational competence, we established the task of displaying a range of speech acts. The conclusion to be drawn from this exercise is that, in general, the system's speech acts are used appropriately -- in a fashion commensurate with the knowledge it has of its hearer's beliefs and goals.

The system was able to plan REQUEST and INFORM of WANT speech acts to influence its user's goals, and INFORM speech acts to influence his beliefs. In addition, we showed that illocutionary acts could be used to achieve subgoals of other such acts. For instance, in addition to requesting that the user perform some action, the system could also decide to supply him with information necessary to his performing that action -- information that it believes he does not already have. Hence, it plans INFORM speech acts when it believes the user needs to know something.

Planning decisions have been found to influence linguistic behavior since they refer to such obvious speech act determining factors as which act should be used to achieve a goal and who will perform a particular action. The choice of which act to use determines not just the act that will be requested, but also affects additional speech acts that might be generated to achieve that act's preconditions. The choice of who should perform an action directly determines whose goals need to be influenced and thus leads to requests to achieve those actions. Consequently, the choice of the agent of some non-linguistic action influences the content of request and inform speech acts.

The system's range was also demonstrated by its ability to plan numerous kinds of questions, viz. WH, yes/no, existential, and teacher/student, when it wanted to know something. This unique ability relied on the development of a representation representation for knowing <u>that</u> someone knows what the value of an expression is without having to know <u>what</u> they think that value is. Our model was then shown to be extendable to the more general problem of planning multi-party speech acts -- ones where acts requested by the speaker involved acts to be performed by parties other than the hearer. The same combinations of additional "need to know" INFORM speech acts could also be included.

The system does produce some illocutionary acts that people in the same situation would find unnecessary. Some of these failures can be attributed to its inability to plan for someone to make inferences from the preconditions of its speech acts. To do this, it would have to plan for the hearer to recognize its utterance as the intended act. In a more comprehensive model, the process that generates such acts would consider how they would be recognized.

The strategy pursued throughout the research has been to model conversation from the point of view of a participant, rather than from that of some neutral observer. This emphasis on egocentrism led to the choice of "believe" rather than "know" as the concept upon which to base the theory. Such a modelling perspective is crucial to building process-models but is not common in logical circles.

In order to plan speech acts, the process-model had to incorporate means for dealing with beliefs and goals that was both formally precise and yet implementable. The thesis presented a formal semantics for a logic of "belief" and "want" that was specially designed to give an interpretation to formulas containing existential quantifiers whose scopes include beliefs. Having an interpretation of these formulas was crucial to this system since they are the representation of "knowing that", the concept involved in planning questions.

The program dealt with such beliefs by distinguishing between existentially quantified variables whose scopes did not include BELIEVE from those whose scopes did. A parallel distinction was made in the formal semantics by restricting the set of objects (to "known constants") over which variables quantifying "into" beliefs could range. A natural language understanding system would need a mechanism for handling quantified beliefs since, from an egocentric viewpoint, they are the logical forms for referential definite and "specific" indefinite noun phrases. The system would frequently acquire such beliefs by way of the user's noun phrases. It was pointed out that this representation for certain noun phrases supplies a hearer with all the information he needs to be able to plan a question to further clarify the speaker's intended referent.

From a practical standpoint, this thesis has shown that a knowledge-based system could plan its communication acts and be sensitive to a model of its user. While certain

techniques were simplifications of the state-of-the-art, (notably the planning algorithm) the system's knowledge representation and organization made use of and extended current research in the area. We can improve upon the current program both in terms of its capabilities and its efficiency. Such a program could form the nucleus around which to develop helpful "assistant" systems.

We have said that the thesis should be regarded as a feasibility study of the view that speech acts can be regarded as another form of intentional behavior. As with most feasibility studies, more questions have been raised as a result of the foundation work. This thesis is but a very small step towards the goal of understanding our use of language. It is hoped that the new approaches to old questions presented here will prove to be fruitful perspectives from which to pursue further research.

BIBLIOGRAPHY

- Abrial, J.R., "Data Semantics", in Data Management Systems, Klimbie and Koffeman, eds., North Holland, 1974.

- Allen, J., "Recognizing Intention in Dialogue", Ph. D. thesis, Dept. of Computer Science, Univ. of Toronto, forthcoming.

- Austin, J. L., How to Do Things with Words, J. O. Urmson (ed.), Oxford University Press, 1962.

- Brown, J. S., Burton, R. R., and A. G. Bell, "SOPHIE -- A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting (An Example of AI in CAI)", Bolt, Beranek and Newman, Inc., Report No. 2790 (AI Report No. 12) Cambridge, Mass., March, 1974.

- Bruce, B. C., and C. F. Schmidt, "Episode Understanding and Belief Guided Parsing", Technical Report CBM-TR-32, Computer Science Dept., Rutgers University, 1974.

- Bruce, B. C., "Generation as a Social Action", Proc. of the Conference on Theoretical Issues in Natural Language Processing, Cambridge, Mass., 1975 (a).

- Bruce, B. C., "Pragmatics in Speech Understanding, Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 1975 (b).

- Bruce, B. C., "Case Systems for Natural Language", Artificial Intelligence, v. 6, no. 4, 1975(c), pp. 327-360.

- Bruce, B., "Plans and Social Actions", Technical Report No. 34, Center for the Study of Reading, Bolt Beranek and Newman, Inc., April 1977.

- Bobrow, D. G., and T. Winograd, "An Overview of KRL, a Knowledge Representation Language", Xerox Palo Alto Research Center Report, May, 1976.

- Carbonell, J. R., and A. M. Collins, "Natural Semantics in Artifical Intelligence, Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, California, 1973. pp. 344-351

- Chomsky, N., Aspects of the Theory of Syntax, M.I.T. Press, Cambridge, Mass., 1965.

- Chomsky, N., "Topics in the Theory of Generative Grammar", Janua Linguarum, Mouton and Co., The Hague, 1966. Reprinted in The Philosophy of Language, J. R. Searle, ed., Oxford University Press, 1972.

- Chomsky, N., Reflections on Language, Pantheon Press, New York, 1975.

- Codd, E. F., "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, June, 1970.

- Donnellan, K., "Reference and Definite Description", The Philosophical Review, v. 75, 1960, pp. 281-304. Reprinted in Semantics, Steinberg and Jacobovits, eds., Cambridge Univ. Press, 1971.

- Fikes, R. and G. Hendrix, "A Network-based Knowledge Representation and its Natural Deduction System",

Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, 1977. pp. 235-246

- Fikes, R. and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence, no. 2, 1971.

- Fillmore, C. J., "The Case for Case", Universals in Linguistic Theory, Bach and Harms, eds, Holt, Rinehart and Winston, New York, 1968, pp. 1-88.

- Gordon, D. and G. Lakoff, "Conversational Postulates", Papers from seventh regional meeting of the Chicago Linguistic Society, Chicago, University of Chicago, Dept. of Linguistics, 1973, reprinted in Syntax and Semantics Volume 3: Speech Acts, Cole, P. and J. Morgan, eds., 1975, pp. 83-106.

- Grice, H. "Logic and Conversation: Implicature", William James Lectures, Harvard Univ., 1967, reprinted in Cole & Morgan, Syntax and Semantics Volume 3: Speech Acts, Academic Press, New York, 1975.

- Grosz, B. J., "The Representation and Use of Focus in a System for Understanding Dialogs", Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, 1977. pp. 67-76.

- Guttag, J., "Abstract Data Types and the Develoment of Data Structures", Communications of the ACM, v. 20, no. 6, June 1977.

- Hayes, Phillip J., "A Representation for Robot Plans", Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 1977. pp. 181-188.

- Hendrix, G., "Expanding the Utility of Semantic Networks through Partitioning", Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, 1975, pp. 115-121.

- Hendrix, G., "The Representation of Semantic Knowledge", ch. 5 of Speech Understanding Research, Donald E. Walker, ed., Stanford Research Institute Report, October 1976.

- Hewitt, C., "Description and Theoretical Analysis (using Schemata) of PLANNER; a Language for Proving Theorems and Manipulating Models in a Robot", Ph.D. thesis, Dept. of Mathematics, M. I. T., 1972.

- Hintikka, J., Knowledge and Belief, Cornell Univ. Press, Ithaca, N.Y., 1963.

- Horrigan, M.K., "Modelling Simple Dialogs", Technical Report No. 108, Department of Computer Science, University of Toronto, 1977.

- Kripke, S., "Semantical Considerations on Modal Logic", Acta Philosophica Fennica, 1963, pp. 83-84, reprinted in Linsky, L., Reference and Modality, Oxford University Press, London, 1971, pp. 63-72.

- Lehnert, W. G., "A Conceptual Theory of Question Answering", Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, 1977.

- Kaplan, D. "Quantifying In", Synthese 19, 1968-69, pp. 178-214.

- Levesque, H.J., "A Procedural Approach to Semantic Networks", Tech Report, Dept. of Computer Science, University of Toronto, 1977.

- Levesque, H. J., and J. Mylopoulos, "A Procedural Semantics for Semantic Networks", to appear 1978.

- Lewis, D. K., Convention: A Philosophical Study, Harvard University Press, Cambridge, Mass., 1969.

- Mann, W.C., Moore, J.A. & J.A. Levin, "A Comprehension Model for Human Dialogue" Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, 1977. pp. 77-87

- McCarthy, J. and Hayes, Pat J., "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in Machine Intelligence 4, Meltzer & Michie, eds., American Elsevier, New York, 1969

- Meehan, J.R., "Tale-Spin , An Interactive Program that Writes Stories", Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass., pp. 91-98.

- Morgan, J. "Some Remarks on the Nature of Sentences", Papers from the Parasession on Functionalism, Grossman, San and Vance, eds, Chicago Linguistics Society, 1975.

- Morgan, J., "Some Interactions of Syntax and Pragmatics", in Syntax and Semantics Volume 3: Speech Acts, Cole and Morgan, eds., Academic, New York, 1975.

- Moore, J.A., and A. Newell, "How can MERLIN Understand?", Knowledge and Cognition, Gregg, L., ed., Lawrence Ehrlbaum Assoc., Baltimore, 1973.

- Moore, R.C., "Reasoning about Knowledge and Action", Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, 1977. pp. 223-227.

- Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N. Tsotsos, J. and H. Wong, "TORUS - A Natural Language Understanding System for Data Management" Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 1977. pp. 414-421.

- Nash-Webber, B. and R. Reiter, "Anaphora and Logical Form: on Formal Meaning Representations for Natural Language", Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, 1977. pp. 121-131

- Nilsson, Nils J., "Artificial Intelligence -- Research and Applications", Stanford Research Institute AI Center Progress Report, Menlo Park, Ca., 1975.

- Norman, D., and D. Rumelhart, eds., Explorations in Cognition W. H. Freeman and Co., 1975.

- Ross, J.R., "Where to do Things With Words", in Syntax and Semantics Volume 3: Speech Acts, Cole, P., and J. Morgan, eds., Academic, New York, 1975.

- Partee, B.H., "Opacity, Coreference, and Pronouns", in Semantics of Natural Language, Davidson and Harman, eds., D. Reidel Publishing Co., Boston, 1972.

- Power, R., "A Computer Model of Conversation", Ph.D. Thesis, University of Edinburgh, 1974.

- Quine, W. V. O., "Quantifiers and Propositional Attitudes", Journal of Philosophy 53, pp. 177-187.

- Rescher, N., Many-Valued Logic, McGraw-Hill, New York, 1969.

- Roussopoulos, N.D., "A Semantic Network Model of Data Bases", Tech. Report 104, Dept. of Computer Science, Univ. of Toronto, 1976.

- Rumelhart, D., Lindsay, P., and D. Norman, "A Process Model for Long-Term Memory", Organization of Memory, Tulving and Donaldson, eds., Academic Press 1972.

- Sacerdoti, Earl, "The Nonlinear Nature of Plans", Proceedings of the Fourth International Joint Conference on Artifical Intelligence, Tbilisi, USSR, 1975.

- Sadock, J.M. Towards a Linguistic Theory of Speech Acts, New York, Academic Press, 1975.

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

- Schank, R. C., "Identification of Conceptualizations Underlying Natural Language", Computer Models of Thought and Language, Schank, R. and K. Colby, eds., W. H. Freeman and Co., San Francisco, 1973.

- Schiffer, Stephen, Meaning, Oxford University Press, 1972.

- Schmidt, C. F. "Understanding Human Action", Proc. of Conference on Theoretical Issues in Natural Language Processing, Cambridge, 1975.

- Schmidt, C. F. and N.S. Sridharan, "Plan Recognition Using a Hypothesize and Revise Paradigm: An Example", Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, 1977. pp. 480-486.

- Searle, J. R., Speech Acts, Cambridge Univ. Press, 1969.

- Searle, J. R., "Indirect Speech Acts", Syntax and Semantics Volume 3: Speech Acts, Cole, P., and J. Morgan, eds., Academic Press, 1975.

- Searle, J. R., "A Taxonomy of Illocutionary Acts", Language, Mind and Knowledge, K. Gunderson, ed., U. of Minnesota Press, 1976.

- Simmons, R. F., and J. Slocum, "Generating English Discourse from Semantic Networks", CACM 15, 10, p.891-905, 1972.

- Simmons, R. F., "Semantic Networks: Their Computation and Use for Understanding English Sentences", Computer Models of Thought and Language, Schank, R. and K. Colby, eds., W. H. Freeman and Co., San Francisco, 1973.

- Srinivasan, C.V., "The Architecture of Coherent Information System: A General Problem Solving System", IEEE Transactions on Computers, c-25, No. 4, April 1976.

- Stevens, A.L. and Collins, A., "The Goal Structure of a Socratic Tutor", Bolt, Beranek and Newman, Inc., Report No. 3518, Cambridge Mass. March 1977.

- Strawson, P.F., "On Referring", Mind, 1950.

- Sussman, G. and D. V. McDermott, "From Planning to Conniving: A Genetic Approach", Proc. ACM Fall Joint Computer Conference, vol. 41, Part 2, 1972.

- Thomason, R.H., Symbolic Logic - An Introduction, London, 1970.

- Wilks, Y., "Primitives and Words", Proc. of Conf. on Theoretical Issues in Natural Language Processing, Cambridge, Mass., June 1975.

- Woods, W., "What's in a Link: Foundations for Semantic Networks", Representation and Understanding: Studies in Cognitive Science, Bobrow, D., and A. Collins, eds., Academic Press, New York, 1975.

- Winograd, T., "Towards a Procedural Understanding of Semantics", Stanford Artificial Intelligence Laboratory Memo AIM-292, Stanford University, November 1976.

- Sacks, H., Schegloff, E.A., Jefferson, G., "A Simplest Systematics for the Organization of Turn Taking for Conversation", LANGUAGE, vol.50, no.4, pp.696-735, 1974.

TECHNICAL REPORTS

Department of Computer Science

From September 1968

Number	Author	Title
* 1	T.E. Hull	The Numerical Integration of Ordinary Differential Equations
* 2	J.C. Mason	Chebyshev Methods for Separable Partial Differential Equations
* 3	Neil Frederick Stewart	The Comparison of Numerical Methods for Ordinary Differential Equations
* 4	J.N.P. Hume and C.B. Rolfson	Scheduling for Fast Turnaround in Job-At-A-Time Processing
* 5	G.F. Gabel	A Predictor-Corrector Method Using Divided Differences
* 6	Chandler Davis and W.M. Kahan	The Rotation of Eigenvectors by a Perturbation - III
* 7	D.J. Cohen and C.C. Gotlieb	The Syntax Graph: A List Structure for Representing Grammars
* 8	D. Tsichritzis	Measures on Countable Sets
9	Brian Smith	Error Bounds, Based Upon Gerschgorin's Theorems, for the Zeros of a Polynomial
* 10	M. Puzin	Simulation of Cellular Patterns by Computer Graphics
* 11	J.C. Mason	Orthogonal Polynomial Approximation Methods in Numerical Analysis
* 12	Donald M. Kaplan	Recursion Induction Applied to Generalized Flowcharts

* Not available

TECHNICAL REPORTS

(continued)

Number	Author	Title
* 13	Sunil K. Pal	Numerical Solution of First-Order Hyperbolic Systems of Partial Differential Equations
* 14	Pat Conroy	Simulation of Texture by Computer Graphics
* 15	J.C. Mason and I. Barrodale	Two Simple Algorithms for Discrete Rational Approximation
* 16	S. Gerschgorin and Paul Turan	Two Translations in Numerical Analysis
* 17	George Olshevsky Jr.	An Automated Aid to Visualizing Convex Hypersolids
* 18 (2nd ed.)	Derek Gordon Corneil	Graph Isomorphism
* 19	Alan Borodin	Computational Complexity and the Existence of Complexity Gaps
* 20	John David Duffin	A Language for Line Drawing
* 21	A. Ballard	Transformations on Programs
* 22	S.A. Cook	Linear Time Simulation of Deterministic Two-Way Pushdown Automata
* 23 (2nd ed.)	D. Tsichritzis (Editor)	Topics in Operating Systems
* 24 (2nd ed.)	R. Bunt and D. Tsichritzis	A Selective Annotated Bibliography for Operating Systems
* 25	J.C. Mason and I. Farkas	Continuous Methods for Free Boundary Problems

---

* Not available

TECHNICAL REPORTS

(continued)

Number	Author	Title
* 26	W.A. Walker and C.C. Gotlieb	A Top Down Algorithm for Constructing Nearly-Optimal Lexographic Trees
* 27	D. Tsichritzis	Iff Programs
* 28	L.W. Jackson	Automatic Error Analysis for the Solution of Ordinary Differential Equations
* 29	T.E. Hull W.H. Enright B.M. Fellen and A.E. Sedgwick	Comparing Numerical Methods for Ordinary Differential Equations
* 30	Stanley Cabay	Second-Order Hyperbolic Equations with Data on Two Intersecting Boundaries
* 31	Douglas Dale Olesky	Inclusion Regions for Partitioned Matrices
* 32	J. Ian Munro	Some Results in the Study of Algorithms
* 33	D. Tsichritzis	Modular System Description
* 34	J. Mylopoulos	On the Relation of Graph Automata and Graph Grammars
* 35	P.H. Roosen-Runge	The Algebra of Distributionally Defined Classifications
* 36	T.E. Hull	Proving the Correctness of Algorithms
	T.E. Hull W.H. Enright and A.E. Sedgwick	The Correctness of Numerical Algorithms

* Not available

TECHNICAL REPORTS

(continued)

Number	Author	Title
37	Erold W. Hinds	Square Roots of an Orthogonal Matrix
* 38	D. Tsichritzis	Topics in Operating Systems Revisited
* 39	David G Kirkpatrick	On the Additions Necessary to Compute Certain Functions
* 40	P. Keast	Multi-Dimensional Quadrature Formulae
* 41	Gordon D. Mulligan	Algorithms for Finding Cliques of a Graph
* 42	Stephen A. Cook and Robert A. Reckhow	Diagonal Theorems for Random Access Machines
* 43	C.J.C. Lee	Translation of Context-Free Programming Languages Using Semantic Trees
* 44	D. Tsichritzis	Lecture Notes on Operating Systems
* 45	David Blair Coldrick	Methods for the Numerical Solution of Integral Equations of the Second Kind
* 46	Wayne Enright	Studies in the Numerical Solution of Stiff Ordinary Differential Equations
* 47	Barry Graham	An Algorithm to Determine the Chromatic Number of a Graph
* 48	Philip A. Bernstein	Description Problems in the Modeling of Asynchronous Computer Systems
* 49	Rudolf Mathon	On the Approximation of Elliptic Boundary Value Problems by Fundamental Solutions

---

* Not available

TECHNICAL REPORTS
(continued)

Number	Author	Title
* 50	George Tourlakis	Some Results in Computational Topology
* 51	R. Michael Wharton	Grammatical Inference and Approximation
* 52	John Mylopoulos Norm Badler Lou Melli and Nicholas Roussopoulos	An Introduction to 1.pak, A Programming Language for AI Applications
* 53	Arthur Sedgwick	An Effective Variable Order Variable Step Adams Method
* 54	Frank Wm. Tompa and C.C. Gotlieb	Choosing a Storage Schema
55	John Mylopoulos Norman Badler Walter Berndl and Lucio Melli	The 1.*pak* Reference Manual
* 56	Keith O. Geddes	Algorithms for Analytic Approximation
* 57	Robert Moenck	Studies in Fast Algebraic Algorithms
* 58	Daniel Brand	Resolution and Equality in Theorem Proving
* 59	R.D. Tennent	Mathematical Semantics and Design of Programming Languages
* 60	G. Hall W.H. Enright T.E. Hull and A.E. Sedgwick	DETEST: A Program for Comparing Numerical Methods for Ordinary Differential Equations

---

* Not available

TECHNICAL REPORTS

(continued)

Number	Author	Title
* 61	C.C. Gotlieb	Data Types and Structures, A Synthetic Approach
* 62	C.A. Steele	DEFT: A Disciplined Extension of Fortran
* 63	T.E. Hull J.J. Hofbauer	Language Facilities for Multiple Precision Floating Point Computation, with Examples and the Description of a Preprocessor
* 64	Philip R. Cohen	A Prototype Natural Language Understanding System
* 65	D.G. Corneil	The Analysis of Graph Theoretical Algorithms
* 66	T.E. Hull W.H. Enright	A Structure for Programs that Solve Ordinary Differential Equations
* 67	Bengt Lindberg	Optimal Stepsize Sequences and Requirements for the Local Error for Methods for (Stiff) Differential Equations
* 68	W.H. Enright R. Bedet I. Farkas T.E. Hull	Test Results on Initial Value Methods for Non-Stiff Ordinary Differential Equations
* 69	W.H. Enright T.E. Hull B. Lindberg	Comparing Numerical Methods for Stiff Systems of Ordinary Differential Equations
* 70	C.C. Gotlieb A.L. Furtado	Data Schemata Based on Directed Graphs
71	T.L. de Carvalho	Some Results in Automatic Theorem-Proving with Applications in Elementary Set Theory and Topology
72	Yiu-Sang Moon	Some Numerical Experiments on Number Theoretic Methods in the Approximation of Multi-dimensional Integrals.

---

* Not available

TECHNICAL REPORTS

(continued)

Number	Author	Title
* 73	Lucio F. Melli	The 2.pak Language: Primitives for AI Applications
* 74	David G. Kirkpatrick	Topics in the Complexity of Combinatorial Algorithms
75	Gerald A. Gorelick	A Complete Axiomatic System for Proving Assertions about Recursive and Non-Recursive Programs
76	Richard B. Bunt	Self-Regulating Schedulers for Operating Systems
* 77	James F. Allen	A Prototype Speech Understanding System
* 78	Alexander T. Borgida	Topics in the Understanding of English Sentences by Computer
* 79	Stephen A. Cook	Axiomatic and Interpretive Semantics for an Algol Fragment
* 80	Norman I. Badler	Temporal Scene Analysis: Conceptual Descriptions of Object Movements
81	R.A. Bedet W.H. Enright & T.E. Hull	Stiff Detest: A Program for Comparing Numerical Methods for stiff Ordinary Differential Equations
* 82	Derek C. Oppen	On Logic and Program Verification
83	Patrick Keast	The Evaluation of One-dimensional Quadrature Routines
* 84	Harry K.T. Wong	Generating English Sentences from Semantic Structures
85	Walter Berndl	An Analysis of the SPITBOL System

* Not available

TECHNICAL REPORTS

(continued)

Number	Author	Title
86	Eshrat Arjomandi	A Study of Parallelism in Graph Theory
*87	Robert A. Reckhow	On The Lengths of Proofs In The Propositional Calculus
88	Martin J. Dowd	Primitive Recursion Arithmetic With Recursion On Notation And Boundedness
89	L.W. Jackson & R.D. Skeel	Convergence And Stability Of Nordsieck Methods
90	Corot C. Reason	A Bi-Directional Speech Parsing Technique
91	Grant A. Cheston	Incremental Algorithms in Graph Theory
92	Peter B. Gibbons	Computing Techniques for the Construction and Analysis of Block Designs
93	John K. Tsotsos	A Prototype Motion Understanding System
94	L.W. Jackson	The Computation of Coefficients of Variable-Step Adams Methods
*95	Stephen A. Cook	Soundness and Completeness of an Axiom System for Program Verification
96	G. Fairweather & P. Keast	A Comparison of Non-Adaptive Romberg Quadrature Routines
97	Charles Rackoff	The Covering and Boundedness Problems for Vector Addition Systems
*98	W.H. Enright	Improving the Efficiency of Matrix Operations in the Numerical Solution of Stiff ODE's

* Not available

TECHNICAL REPORTS

(continued)

Number	Author	Title
*99	Daniel Brand	Proving Programs Incorrect
*100	T.E. Hull W.H. Enright K.R. Jackson	User's Guide for DVERK - a Subroutine for Solving Non-stiff ODE's
*101	K.R. Jackson W.H. Enright T.E. Hull	A Theoretical Criterion for Comparing Runge-Kutta Formulas
*102	R.L. Johnston C. Addison	STUNT - Software for Teaching Undergraduates Numerical Techniques
*103	Pavol Sermer	Finite Element Methods in the Numerical Solution of Mixed-Type Partial Differential Equations
*104	Nicholas D. Roussopoulos	A Semantic Network Model of Data Bases
*105	Hector J. Levesque	A Procedural Approach to Semantic Networks
106	Manfred Maier	Some Numerical Results from the Study of Steady Fluid Flow
*107	Robin Cohen	Computer Analysis of Temporal Reference
108	Mary Katherine Horrigan	Modelling Simple Dialogs
109	Stephen A. Cook Robert A. Reckhow	The Relative Power of Propositional Proof Systems
110	R.L. Johnston Rudolph Mathon	The Computation of Electric Dipole Fields in Conducting Media
111	W.H. Enright	The Efficient Solution of Linear Constant Coefficient Systems of ODEs
*112	Alexander T. Borgida	Formal Studies of Stratificational Grammars
*113	Patrick W. Dymond	Complexity Relationships among some Models of Computation

---

* Not available

TECHNICAL REPORTS

(continued)

Number	Author	Title
114	Leslie M. Goldschlager	Synchronous Parallel Computation
115	Peter F. Schneider	Organization of Knowledge in a Procedural Semantic Network Formalism
116	P. Keast	Families of s-Dimensional, Degree 2t+1 Quadrature Rules for Product Spaces
117	Stephen A. Cook and Charles W. Rackoff	Space Lower Bounds for Maze Threadability on Restricted Machines
118	Philip R. Cohen	On Knowing What to Say: Planning Speech Acts

---

* Not available