SRI International

INFERRING DOMAIN PLANS IN QUESTION-ANSWERING

Technical Note 403

December 1, 1986

By: Martha E. Pollack, Computer Scientist

Artificial Intelligence Center
Computer and Information Sciences Division

This paper is a slightly revised version of a thesis submitted to the Department of Computer and Information Science of the University of Pennsylania on May 9, 1986, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

SRI International

The portion of this research conducted at the University of Pennsylvania was supported by an IBM Graduate Fellowship. The portion conducted as SRI was made possible in part by a gift from the Systems Development Foundation, and was done as part of a coordinated research effort with the Center for the Study of Language and Information, Stanford University. The portion conducted at SRI was also supported in part by the Defense Advanced Research Projects Agency under Contract N00039-84-K-0078 with the Space and Naval Warfare Command, and by the Office of Naval Research under Contract N00014-85-C-0013. The views and conclusions contained in this document are those of the author and should not be interpreted as representative of the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

ABSTRACT

The importance of plan inference in models of conversation has been widely noted in the computational-linguistics literature, and its incorporation in question-answering systems has enabled a range of cooperative behaviors. The plan inference process in each of these systems, however, has assumed that the questioner (Q), whose plan is being inferred, and the respondent (R), who is drawing the inference, have identical beliefs about the actions in the domain. I demonstrate that this assumption is too strong and that it often results in failure not only of the plan-inference process, but also of the communicative process that plan inference is meant to support. In particular, it precludes the principled generation of appropriate responses to queries that arise from invalid plans. I present a model of plan inference in conversation that distinguishes between the beliefs of the questioner and the beliefs of the respondent. This model rests on an account of plans as mental phenomena: "having a plan" is analyzed as having a particular configuration of beliefs and intentions. Judgements that a plan is invalid are associated with particular discrepancies between the beliefs that R ascribes to Q, when R believes that Q has some particular plan, and the beliefs that R herself holds. I define several types of invalidities from which a plan may suffer, relating each to a particular type of belief discrepancy, and show that the types of any invalidities judged to be present in the plan underlying a query can affect the content of a cooperative response. The plan inference model has been implemented in SPIRIT, a System for Plan Inference that Reasons about Invalidities Too, which reasons about plans underlying queries in the domain of computer mail.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Bonnie Webber, for her comments on my work, and for letting me go where it led, even to California. I also thank the members of my committee: Tim Finin, for his continual encouragement; Aravind Joshi, for a number of lengthy conversations that proved critical both to the development of my thesis and to my graduate education as a whole -- and for asking me how to reach Kathy at the hospital when in fact she had already gone home; and Barbara Grosz, for initially inviting me out to SRI, for finding me a copy of the out-of-print book *A Theory of Human Action*, and for immeasurable guidance, friendship and support: I cannot overemphasize her influence on my work. Julia Hirschberg and Kathy McCoy, the best of colleagues and the best of friends, spent hours reading various drafts of this thesis, discussing with me its problems and strengths, and cheering me up when the inevitable dissertation-blues hit. I am also grateful to Elena Levy, Gopalan Nadathur, Robert Rubinoff, Ethel Schuster, Gregory Ward, Cliff Wulfman, and all the others whose time in the Penn NL community overlapped with my own, for enlightening and interesting discussions on this work. Richard Billington provided computing assistance and good cheer. Henry Kautz read a draft of this thesis and sent an extremely useful set of criticisms and suggestions.

This work was begun, and later finished, at the Artificial Intelligence Center at SRI International and at the Center for the Study of Language and Information. There I have benefitted greatly from discussions with Michael Bratman, Barbara Grosz, Kurt Konolige, Ray Perrault, and the rest of the NL and R&R groups at SRI, and from the lively meetings of the CSLI Rational Agency Project: Michael Bratman, Phil Cohen, Todd Davies, Mike Georgeff, David Israel, Kurt Konolige, and Amy Lansky. Stan Rosenschein asked some crucial questions while I was putting together my proposal. And Brian Fromme helped me cope with a move from one machine to another, just a couple of months before my defense, by answering a million questions and fixing (at least) two million bugs.

For financial support I gratefully acknowledge the IBM Corporation, which provided me with a generous fellowship, and SRI International, which provided me with a job. Finally, many thanks go to my parents, Martin and Susan Pollack, who always encouraged me to learn and who never doubted my abilities, and to my husband, Ken Gottschlich, who provided the love, laughter and understanding that were needed to write this dissertation (and who cooked many more than his fair share of our dinners while I was doing so).

Table of Contents

1. Introduction **1**
1.1. The Problem 1
1.2. Claims of the Thesis 5
1.3. Overview of the Thesis 7
2. Approaches to the Plan Inference Problem **9**
2.1. Three Dimensions of Plan Inference 9
2.2. Existing Methods for Plan Inference 17
2.3. Necessary Assumptions of Existing Models 24
2.4. Explicitly Identified Goals 28
3. Plans as Mental Phenomena **33**
3.1. The Data-Structure View and the Mental Phenomenon View 33
3.2. Types of Plans 35
3.3. What is a Plan? 36
4. Representing Simple Plans **45**
4.1. Toward a Formal Representation 45
4.2. Representing Actions 46
4.3. The Generation Relation 52
4.4. The Executability Relation 67
4.5. Representing Beliefs and Intentions 73
4.6. Combining the Pieces 78
4.7. Reconsidering the Standard Model 83
5. Invalidities in Simple Plans **87**
5.1. Invalidity Judgements 87
5.2. Unexecutable Acts 91
5.3. Ill-Formed Plans 98
5.4. Incoherent Queries 107
5.5. An Alternative Taxonomy 115
6. Inferring Simple Plans **118**
6.1. The Role of Plan Inference in Question-Answering 118
6.2. Representing Queries 123
6.3. The Plan Inference Process 126
6.4. Plan Inference Rules 132

7. Automatic Plan Inference: SPIRIT **147**
7.1. SPIRIT 147
7.2. The Domain 148
7.3. Overview of the System 149
7.4. System Components 152
7.5. SPIRIT in Operation 159
8. Conclusion **182**
8.1. Summary 182
8.2. Future Directions 183
APPENDIX A. Axioms for Plan Inference **186**
APPENDIX B. Compendium of Examples **192**
INDEX **209**

List of Figures

Figure 2-1: Interaction of Awareness and Cooperation 12
Figure 2-2: Examples of Plan Inference Problems Characterized by the First Two Features 14
Figure 2-3: AI Research on Plan Inference 16
Figure 2-4: Constructing a Subgraph from an Operator 20
Figure 2-5: Rules for Constructing Plan Graphs 21
Figure 4-1: Partitioning the Set of All Act-Types 69
Figure 4-2: G's Plan to Join the Mayday Late-Night Club 79
Figure 4-3: G's Plan to Get His Driver's License (adapted from Goldman, pp. 69-60) 82
Figure 4-4: G's Plan to Get His Cousin to Leave 82
Figure 4-5: G's Plan to Speed up Entry into the Mail System 83
Figure 4-6: G's Plan to Find out the Phone Number of the Hospital 83
Figure 4-7: Interpretations of the Standard Relations 85
Figure 5-1: Q's Plan to Establish a Communication Channel to Kathy 99
Figure 5-2: Combinations of Types of Invalidities 105
Figure 5-3: Validity Properties and their Objects 115
Figure 5-4: Joshi, Webber and Weischedel's Situations 116
Figure 6-1: The Process of Answering a Question 120
Figure 6-2: Plans in Question-Answering 122
Figure 6-3: The Process of Inferring an Eplan 133
Figure 6-4: Inferred Eplan for Q to Establish A Channel to Kathy 138
Figure 6-5: Inferred Eplan for Q to Prevent Tom from Reading a File 141
Figure 6-6: Alternate Eplan for Q to Prevent Tom from Reading a File 143
Figure 7-1: Schematic Diagram of SPIRIT 150

CHAPTER I
Introduction

1.1. The Problem

If you overheard the following conversation, you would probably find it quite unremarkable:

> (1) Q: "I want to talk to Kathy. Do you know the phone number at the hospital?"
> R: "She's already been discharged. Her home number is 555-8321."

Yet for designers of computer systems that answer questions, this conversation offers a serious challenge. R's response, although wholly appropriate, does not include the information requested by Q. In fact, given what R knows, a direct response, stating the hospital's phone number, would have been quite inappropriate. How can we account for this apparent paradox? A commonsense analysis is that Q has asked for information that is not appropriate to his goal, and R, realizing this, has attempted to provide information that *is* appropriate, i.e., that will help Q achieve his goal. Further, R has told Q why she believes that the information requested is not appropriate to his goal: she has explained why his plan is invalid. If R realizes the inappropriateness of the query, it is, course, equally inappropriate for her to answer it directly and without further comment.

To satisfy their users' needs, computer systems that engage in answering questions, including expert and help systems, should be able to perform as well as R does. If the commonsense analysis of Example (1) is correct, then these systems must be able to assess the questions they are asked to determine whether they are appropriate to the questioners' goals. To do this, they will need to be able to infer the questioners' plans, including plans that may be invalid.

The importance of plan inference in models of conversation has been widely noted in the computational-linguistics literature. An ability to reason about plans and goals has been shown to be useful in such tasks as resolving referring expressions and understanding ellipsis in natural-language interpretation [Grosz 77], and selecting illocutionary force, referring expressions, syntactic forms, and even accompanying gestures in generation [Cohen 79, Appelt 85]. Studies of conversation have inspired claims that "[h]uman conversational participants depend upon the ability of their partners to recognize their intentions, so that those partners may be capable of responding appropriately" [Sidner 81, p. 203], and that people who use question-answering systems will "expect to engage in a conversation whose coherence is manifested in the interdependence of their often unstated plans and goals with those of the system" [Cohen 82, p. 245]. Incorporating plan inference capabilities into systems that answer users' questions has enabled such systems to handle indirect speech acts [Perrault 80], to supply more information than is actually requested in a query [Allen 83a], to provide helpful information in response to a yes/no query answered in the negative [Allen 83a], to disambiguate requests [Sidner 83], to resolve certain forms of intersentential ellipsis [Carberry 85, Litman 85] and to handle such discourse phenomena as clarification dialogues [Litman 85], and correction or "debugging" subdialogues [Litman 85, Sidner 85].

The plan inference process in each of these systems, however, has assumed that the agent whose plan is being inferred (whom I call the *actor*, or *he*), and the agent drawing the inference (whom I call the *inferring agent* or *she*), have identical beliefs about the actions in the domain. Thus, James Allen's model, which was one of the earliest accounts of plan inference in conversation[1] and inspired a great deal of the work done subsequently, includes, as a typical plan inference rule, the following: "SBAW(P) $\rightarrow_i$ SBAW(ACT) if P is a precondition of ACT" [Allen 83a, p. 120]. This rule can be glossed: "if the system (inferring agent) believes that an agent (actor) wants some proposition P to be true, then the system may draw the inference that the agent wants to perform some action ACT of which P is a precondition." Note that it is left unstated precisely who it is--the inferring agent or the actor--that believes P is a precondition of ACT. If we take this to be a belief of the inferring agent, it is not clear that the latter will infer the actor's plan; on the other hand, if we consider it to be a belief of the actor, it is unclear how the inferring agent comes to have direct access to it. In practice, there is only a single set of operators relating preconditions and actions in Allen's system: the belief in question is regarded as being both the actor's and the inferring agent's.

[1]Allen's article [Allen 83a] summarizes his dissertation research [Allen 79].

In many situations, an assumption that the relevant beliefs of the actor are identical with those of the inferring agent results in failure not only of the plan inference process, but also of the communicative process that plan inference is meant to support. In particular, this assumption precludes the principled generation of appropriate responses to queries that arise from invalid plans. Example (1) involves a query that arises from an invalid plan. As I will show, the type of invalidity exemplified there, which amounts to a necessary precondition for some intended act not holding, can, in principle, be handled by existing models of plan inference, although these models have neither focussed on such invalidities nor considered the content of appropriate responses to queries that are detected to have them. However, there are a range of other invalidities that cannot be handled by the existing models; these are illustrated in the following examples.

Imagine a computer user saying to an expert:

> (2) Q: "I want to prevent Tom from reading my file. How can I set the permissions on it to faculty-read only?"

Depending on how the world is, a number of different responses to Q might be appropriate. In one set of circumstances, we might see

> (3) R: "There's no way for you to set the permissions to faculty-read only. What you can do is move it to a password-protected subdirectory. That will keep Tom out."

If we were to attempt a commonsense analysis of R's response in Example (3), we would say that R has made sense of Q's plan, but that she believes that the act he asks how to do, setting the permissions on a file to faculty-read only, cannot be performed. R, however, provides Q with an alternative way to achieve his goal of preventing Tom from reading the file.

In different circumstances, we might see a response like Example (4), which we could explain by saying that R believes that Q's goal act, as well as the act he asks how to do, cannot be performed:

> (4) R: "There's no way for you to set the permissions to faculty-read only, nor is there any way to prevent Tom from reading the file."

In yet other circumstances, we might see a response such as

(5) R: "Well, the command is
SET PROTECTION (Faculty:Read)
but that won't keep Tom out: file permissions don't apply
to the system manager.

Here, our commonsense account is that R believes that Q mistakenly believes that by setting the permissions on his file he can affect Tom's access to it. Further, R, in her response, refutes what she believes may be the source of Q's error, namely Q's incorrect belief that file permissions apply to the system manager. In this example, R believes that the act that Q asks how to do can be performed, although it will not lead to Q's intended goal. In other circumstances, R might believe that the act about which Q asks cannot be performed; still, she may be able to assess that, if it could be, it would not lead to Q's goal. We might then see a response like

(6) R: "There's no way for you to set the permissions to
faculty-read only, and even if there were, it wouldn't keep
Tom out: file permissions don't apply to the system
manager."

In Examples (3) through (6), R seems to have "made sense" of Q's query. However, there can also be circumstances in which R cannot determine why Q asks his query; we might then see responses like

(7) R: "Huh? Tom *is* a faculty member."

Here, R believes that Q believes that Tom is a faculty member, and so R cannot understand why Q might believe that by setting the permissions to faculty-read only, he can prevent Tom from accessing his file.

Taken together, the analyses of Examples (3) through (7) illustrate the sort of reasoning that can support the generation of appropriate responses to queries. The analyses include several commonsense notions: of acts not being performable, of acts not fitting together in the way they were intended, of queries not making sense. The challenge for designers of systems that answer questions is, first, to construct a model of plans that accounts for these commonsense notions of invalidities in plans; second, to construct a model of the process of plan inference in which plans that suffer from one or more of these invalidities can be inferred; and, third, to enable their systems to generate responses that take into account any invalidities detected in the plans inferred to underlie questions.

This thesis is intended to be a response to that challenge. Like all work in artificial-intelligence, it plays a dual role. On the one hand, it contributes to our understanding of how to construct intelligent artifacts. In this role, although the work derives from a commonsense analysis of human behavior, it does not rest on the cognitive reality of that analysis. Even if it turns out, for example, that the notion of plans plays no part in human intelligence, it may still be a useful concept in machine intelligence. Nonetheless, the successful construction of artificially intelligent artifacts provides support for the commonsense analysis of human behavior upon which it is based--albeit weak support, just one of many factors that together conspire towards "proof". This then is the second role of research in artificial intelligence. The emulation of human behavior using notions of plans may be considered support for the claim that such notions play a role in naturally intelligent systems.

In this thesis, I have will very little to say about the second role of this work. I will make no specific claims about psychological reality, and any statements to the effect that "The (human) agent does X" should be taken to mean that "One way to account for the (human) agent's behavior is to say that she does X." My focus will be on using such accounts to enhance the intelligence of computer systems, especially computer systems that provide advice, by enabling them to reason about the plans that may underlie the questions they are asked.

1.2. Claims of the Thesis

I have already alluded to the principal claims of this work: that in order to provide appropriate answers to the questions they are asked, intelligent agents must be able to recognize when an invalid plan underlies a question; and that simply knowing that a plan is invalid is not sufficient--agents must also be able to locate the source(s) of the invalidity. Further, there are several different sorts of invalidities that agents must be able to distinguish between.

Support for these claims comes from the study of transcripts of naturally occurring dialogues. Many of the examples discussed in this thesis were inspired by four sets of dialogues. The first is a set of dialogues recorded by M. K.

Horrigan [Horrigan 77] at a Toronto train-station information booth; these have also been the source of data for several other plan inference systems [Allen 83a, Litman 85]. The second is a set of conversations from a "naturally occurring expert system," a radio talk show, in which callers solicit advice from a financial expert [Pollack 82]. The third is a set of transcripts collected at the University of Pennsylvania, in which users of an electronic mail system were asked to submit questions to the designer of the system in order to assist her in the task of writing a user manual. Both the questions sent to the designer and her responses to them were automatically archived, resulting in a permanent record of the conversations.[2] The fourth set of dialogues, collected in the Wharton School's public terminal room, records instances of computer users seeking advice from a consultant [Eccli 83]. Additionally, a number of naturally occurring examples were noted in more informal circumstances.

Examination of these dialogues reveals the importance in question-answering of being able to reason about invalid plans. I will demonstrate that in existing models of plan inference, which have grown out of Allen's paradigm, only a very restricted class of invalid plans are inferable. As I have already mentioned, the inference of invalid plans has not been a concern of existing systems.[3] Further, these systems have not at all explored the relationship between types of invalidities and types of response strategies.

I will develop, and describe an implementation of, a computational model of plan inference that permits the inference of a wide range of invalid plans. I will also show how such inference, when performed in question-answering, can effect the content of an appropriate response. In support of this, I will provide an analysis of plans as mental phenomena: I will analyze "having a plan" as having a particular configuration of beliefs and intentions. This is something of

[2]My thanks to Sharon Perl, the mail-system designer, for giving me this set of transcripts.

[3]Two exceptions can be found in Sandra Carberry's thesis research [Carberry 85], and in the work on the Consul system [Mark 81]. Each of these include techniques for handling queries that arise from invalid plans, under certain restricted conditions. Carberry's system can process queries that presume relationships that do not exist in the underlying model, but to do so successfully, it must already have inferred, from previous discourse, the questioner's plan. In effect, the plans underlying the queries are valid, and the queries themselves invalid only in the incorrect use of some term. Consul can handle queries that arise from plans that are inappropriate in one domain (electronic mail) if it can determine that they are appropriate in another (U.S. mail).

a departure from the usual accounts of plans given in AI, in which plans are primarily viewed as data structures encoding sets of actions that have some known effect on particular states of the world, and in which the view of plans as mental phenomena, when it arises at all, is derivative on the view of plans as data structures. Plan inference, I will suggest, demands that we take more seriously an account of plans as mental phenomena. I will show how such an account can be used to explain commonsense notions of invalidities in plans in terms of particular types of discrepancies between the beliefs that an inferring agent ascribes to an actor, when she believes that he has some plan, and the beliefs that she herself holds.

I will also argue that there are regularities in cooperative conversation that cannot be explained in a framework in which the inferring agent ascribes to the actor an arbitrary set of beliefs and intentions that satisfy the requirements of "having a plan." Instead, I will claim, it is necessary for the purpose of formulating a response also to ascribe to the actor a set of beliefs that explain the beliefs his plan includes.

In addition, I will make a claim in the form of a disclaimer: I will argue that a model of plan inference, even one that can infer invalid plans and determine the sources of their invalidities, is not by itself sufficient to explain the generation of appropriate responses to queries. Rather, plan inference is but one of several processes that contributes to the determination of a cooperative response to a query.

1.3. Overview of the Thesis

In the next chapter, I will situate this work within the panorama of existing research on plan inference. In so doing, I will identify those assumptions made in earlier research that preclude the inference of a wide range of invalid plans. I will also argue that the need for plan inference in question-answering is even more wide-ranging than has previously been noted. Then in Chapter 3, I will develop an account of plans as mental phenomena, analyzing the state of "having a plan" as having a particular configuration of beliefs and intentions. In Chapter 4, I will formalize this model for a restricted subset of plans which I call *simple plans.* This formalization will require adopting representations for actions; for two relations over actions--generation and executability; and for the attitudes of belief and intention.

In Chapter 5 I will define types of invalidities that can occur in simple plans. I will show how observed regularities in naturally occurring discourse can be explained in terms of discrepancies between the beliefs that the inferring agent herself holds and the beliefs that she ascribes to the actor. I will also discuss a problem for the plan inference model as it has so far been developed, and will suggest a solution, in the form of *explanatory plans*, or *eplans*.

In Chapter 6, I will develop a model of the process of inferring simple plans, even when they are invalid. Because the principal concern of this thesis is the inference of plans in question-answering, in this chapter I will also discuss the relationship between plan inference and the remaining components of a model of the question-answering process. Next, in Chapter 7, I will discuss a small demonstration system, SPIRIT, implemented in Prolog to illustrate the plan inference model. SPIRIT infers and evaluates the plans underlying questions asked by users about the domain of computer mail. Finally, in Chapter 8, I will summarize the contributions of this work and describe potential extensions to it.

CHAPTER II
Approaches to the Plan Inference Problem

This chapter begins by surveying existing work in plan inference. It identifies three parameters of plan inference problems: the cooperativity of the actor; the mode of interaction between the actor and the observer--conversation or observation; and the type of inferred plan--communicative or domain. The problem addressed in this thesis is defined to be inferring the domain plan of a cooperative actor through conversation. Standard AI methods for this class of plan inference problem are described. Implicit assumptions of the standard methods are identified and shown to preclude the inference of a great many invalid plans. The particular assumptions that will be avoided in this work are specified. The chapter concludes by demonstrating that the need for plan inference in question-answering is even more wide-ranging than has been noted in research that uses the standard methods.

2.1. Three Dimensions of Plan Inference

The importance of providing intelligent computer systems with an ability to infer plans has been widely discussed in the AI literature, and there have been a number of studies directed toward what might be called "the plan inference problem," which has the goal of automating the process by which one agent can infer another agent's plans. Such a description is somewhat misleading, since there are actually a cluster of related problems that can be distinguished from one another along three interrelated dimensions, specifically:

1. Is the agent whose plan is being inferred cooperating in the inference process?

2. Does the inferring agent converse with the agent whose plan is being inferred, or instead merely observe his actions?

3. Is the plan being inferred a "communicative" plan or a "domain" plan?

In this section, I provide an overview of these characteristics of plan inference, using them to categorize existing research. Then in the remainder of this chapter, I situate my own work with respect to other plan inference research.

To simplify the discussion, I will use the terminology introduced in Chapter 1. I will refer to the agent whose plan is being inferred as the actor (alternately "he"), and to the agent who is inferring the plan as the inferring agent (alternately "she"). Because I will usually be concerned with dialogue situations in which a respondent is inferring a questioner's plan, I will also use Q to refer to an actor and R to an inferring agent.

Cooperation, Passive Noncooperation, Active Noncooperation

The first feature with which I will categorize a plan inference problem is the cooperativeness of the actor. I will say that an actor *cooperates* in the plan inference process insofar as he is aware that his plan is being inferred and behaves in a manner that will facilitate that inference. Thus an actor may be noncooperative either passively, by being unaware of the inference process and consequently simply failing to behave in such a manner as to facilitate it, or actively, by behaving in a manner intended to hinder the inference process.

Several AI systems have attempted to infer the plans of an actor who is passively noncooperative. Some of these have been constructed for their psychological interest [Schmidt 78], while others are attempts at designing practical systems that can "watch over the shoulder" of a computer-system user, attempting to recognize his plans in order to critique them for errors and inefficiencies [Finin 82, Genesereth 79, McCue 83, Fischer 85]. Philip Cohen, C. Raymond Perrault, and James Allen [Cohen 82] have called this class of problems "keyhole recognition" problems, because the amount of information with which the inferring agent has to work is not much more than she would have watching the actor through a keyhole. They have argued that without the cooperation of the actor, the plan inference problem is extremely difficult in general; the systems that perform this type of inference all rely on very strong domain constraints.

Plan inference should be even more difficult when the actor is actively or intentionally noncooperative. An example of this situation is courtroom discourse, in which a prosecuting attorney, in questioning the defendant, might attempt to hide his plans so as to elicit answers that the defendant would not give if he recognized how they fit into the attorney's plan to prove his guilt. To the best of my knowledge, there have been no attempts made in AI to model this sort of situation.

Within this thesis, I deal with the inference of plans of cooperating actors: people seeking advice are aware that the person providing the advice needs to know what they are trying to accomplish, and in general, in order to get the advice, will assist the inferring agent in her task. Let us turn our attention, then, to the problem of plan inference with a cooperating actor.

Plan-based analyses of human conversation [Cohen 79, Perrault 80, Allen 83a, Cohen 80, Allen 83b, Sidner 81, Sidner 83, Sidner 85, Carberry 85, Litman 85] present the most obvious example of this class of plan inference problems. When these analyses focus on the recognition of communicative plans (for the definition of which, see p. 13), their assumption of cooperativeness is a consequence of H.P. Grice's notion of *intended recognition* [Grice 68, Grice 69]. Grice claims that in conversation, each speaker has a distinguished set of goals that he intends to have recognized; the meaning of his utterance depends directly upon such recognition. As a result of his intention a speaker encodes in his utterance information that will facilitate the recognition process; thus speakers can be said to be cooperating in that process. This notion of intended recognition has been extended to apply to domain plans (again, see p. 13); as such it is the basis of a number of specific heuristics used to control search in plan inference systems, among them Allen's important "forking heuristic"[4] and Candace Sidner's "single-branch assumption"[5]. These heuristics are based on arguments of the form "Inference X would be extremely difficult, and actor A, knowing that I have to infer his plan, would not put me in the position of having to make such a difficult inference."

[4]Namely, the likelihood that a partially inferred plan is part of the "correct" plan--the one the speaker intends--is inversely proportional to the number of alternatives into which it can be expanded [Allen 83a].

[5]Namely, if the speaker believes that more than one plan might be inferred at some stage of the discourse, it is his responsibility to make known the one he intends [Sidner 85].

Systems that understand stories by reasoning about the characters' plans [Bruce 75, Schank 77, Wilensky 83] can be seen as hybrid systems: on the one hand, the characters whose plans are being inferred are passively noncooperative, but on the other, the author presumably is actively cooperative. In literature there are thus two actors: the cooperative author and the passively noncooperative actor (who, while noncooperative with the reader's inference of his plans, may in fact be depicted as cooperative in the inference process of other characters, as when they converse with him.[6]) The reader must sometimes infer the plans of the character, sometimes the plans of the author, and sometimes both. Further, these plans may be quite interdependent. For this reason among others, modeling deep understanding of literature is a very difficult problem.

Figure 2-1 summarizes the dependencies between an actor's awareness of the plan inference process and his cooperation in it. To cooperate, the actor obviously must be aware that his plan is being inferred. But, although necessary, such awareness is not sufficient to guarantee his cooperation: as we saw with the example of courtroom discourse, the actor may instead choose to be actively noncooperative. However, once aware that his plan is being inferred, the actor must choose between cooperativeness and active noncooperativeness: passive noncooperation ceases to be an option. Lack of awareness is thus necessary for passive noncooperation. It turns out also to be sufficient: if the actor is unaware that his plan is being inferred, he can neither intentionally facilitate nor thwart that process.

```
AWARE ACTOR          <----->        COOPERATING ACTOR
                                           or
                                ACTIVELY NONCOOPERATING ACTOR


UNAWARE ACTOR        <----->    PASSIVELY NONCOOPERATING ACTOR
```

Figure 2-1: Interaction of Awareness and Cooperation

[6] Bruce and Newman [Bruce 78] discuss a theory of story understanding in which interpretation is seen as an attempt to understand the interacting plans of the story's characters.

Conversation/Observation

So far, the examples mentioned of plan inference with a cooperating actor have involved conversational settings: the inferring agent operates by conversing with the actor, and her inference is a function of the actor's verbal utterances. Other situations, however, are possible. The parlor game charades, for instance, provides an example of a cooperating actor, indeed an essentially cooperating actor (the pantomiming player), whose plans are being inferred by other agents (the guessing players) on the basis of the actor's physical actions, rather than by any verbal exchange.

This then brings us to the second way in which plan inference problems can be categorized: does the inferring agent infer the plan by observing the actor or by conversing with him? The charades example, along with the examples of everyday conversation, illustrate that plan inference with a cooperating actor can occur in both settings. Courtroom discourse provides an example of an actively noncooperating actor in a conversational setting, and we can construct examples of actively noncooperating actors in an observational setting: for instance, a mobster who reaches into his empty coat pocket to deceive someone into believing that he is about to pull out a gun. We can call this an instance of "anticharades." Thus, plan inference with an aware actor, cooperating or actively noncooperating, can occur in both conversational and observational settings. In contrast, cases of plan inference in conversation with an unaware actor are impossible--that is, speakers always know that their plans are being inferred--since knowing how to speak entails "knowing" about intended recognition. As Grice explains, a speaker may opt out of the cooperative principle, but he cannot be ignorant of it [Grice 75] . Plan inference with an unaware actor, i.e., one who is passively noncooperative, is restricted to observational settings. Figure 2-2 summarizes the picture so far and recapitulates the relevant examples.

In the question-answering situations that are of primary concern in this thesis, the actor's plan is inferred by conversing with him.

Communicative Plans/Domain Plans

Although the two categorizing features of plan inference problems discussed above have involved the circumstances under which the plan is inferred, the final feature, to which we now turn, involves the type of plan inferred. This feature, like the preceding one, is significant only when the plan inference involves an

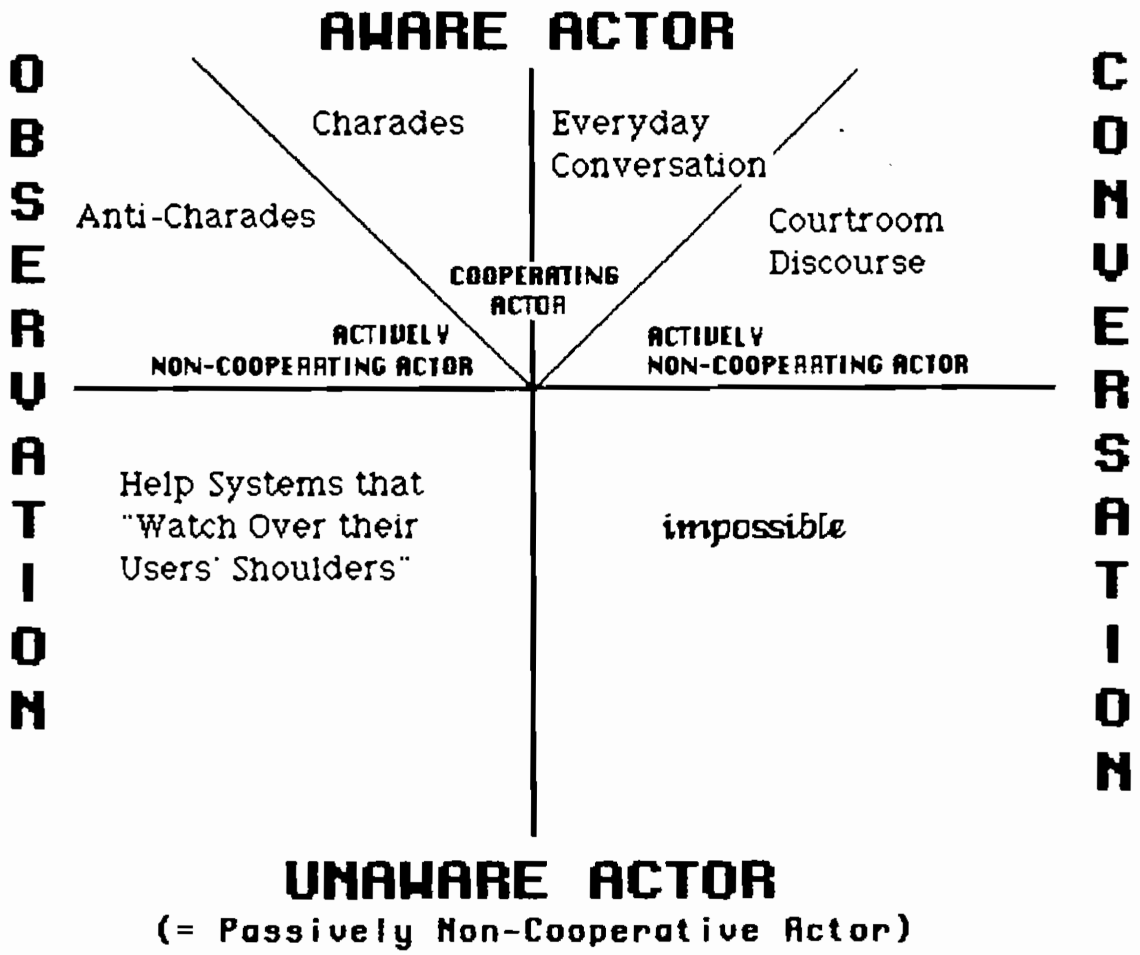

AWARE ACTOR
OBSERVATION
CONVERSATION
Charades
Everyday Conversation
Anti-Charades
Courtroom Discourse
COOPERATING ACTOR
ACTIVELY NON-COOPERATING ACTOR
ACTIVELY NON-COOPERATING ACTOR
Help Systems that "Watch Over their Users' Shoulders"
impossible
UNAWARE ACTOR
(= Passively Non-Cooperative Actor)

Figure 2-2: Examples of Plan Inference Problems Characterized by the First Two Features

aware actor. In such cases, we can distinguish between the inference of communicative plans and domain plans. This distinction can probably best be illustrated with an example. Consider the question "Do you know when the train to Windsor leaves?" discussed by Allen [Allen 83a]. If an inferring agent were, upon hearing this query, to infer the communicative plan of the actor (speaker), she would recognize this as an (indirect) request to be told the departure time of the Windsor train: communicative goal is roughly equivalent to illocutionary force in the sense of J. L. Austin [Austin 75]. In contrast, an agent who inferred domain plans would determine that the speaker may be asking this question because he wants to take the Windsor train or because he wants to go to Windsor.

Several of the works cited above as examples of plan-based analyses of communication focus on the inference of communicative plans [Cohen 79, Perrault 80, Cohen 80]. In contrast, other papers [Allen 80, Sidner 81, Sidner 83] emphasize the problem of inferring domain plans with a cooperative actor in a communicative setting. The Consul System [Mark 81] is another example of a system that infers domain plans. There has also been work [Allen 83a, Carberry 85, Litman 85, Allen 83b, Sidner 85] that attempts to infer both the communicative and domain plans underlying an utterance. Although several of these works have argued that the two processes (of inferring communicative and domain plans) are synergetic [Allen 83b, Litman 85], In this thesis, I downplay the question of inferring communicative plans and instead concentrate on the problem of inferring the domain plan of a cooperating actor.

My decision to concentrate on domain-plan inference rests on the fact that my primary concern is with enabling intelligent systems to cope with potentially invalid plans underlying the questions they are asked, especially where those plans are novel to them. As speakers of a common language, we seem to share knowledge both of a set of communicative goals and a set of conventionalized plans, upon which we often draw, for achieving those goals. Speakers all have roughly equivalent knowledge about communicative acts, in contrast to the gross imbalances that may exist in their knowledge about the actions in any domain.[7] Such imbalances are the source of one agent coming to have a plan that is novel to another agent.

Figure 2-3 summarizes the distinctions that I have been drawing and shows how existing AI research on plan inference fits into this categorization scheme. Because, to my knowledge, there has been no research on the problem on plan inference with an actively noncooperating actor, I have omitted this feature value from the figure. Thus, everything above the center horizontal line in the figure involves plan inference with an aware, cooperating actor, and everything below

[7]Knowledge about communicative acts may not be strictly equivalent, since some people do seem to be better communicators than others, but the range of variation is probably smaller: there are not people who are experts at generating or understanding indirect speech acts, nor linguistically normal people who are unable to use them.

involves plan inference with an unaware actor. The center vertical line delimits plan inference in an observational setting (to its left) and in a conversational setting (to its right).

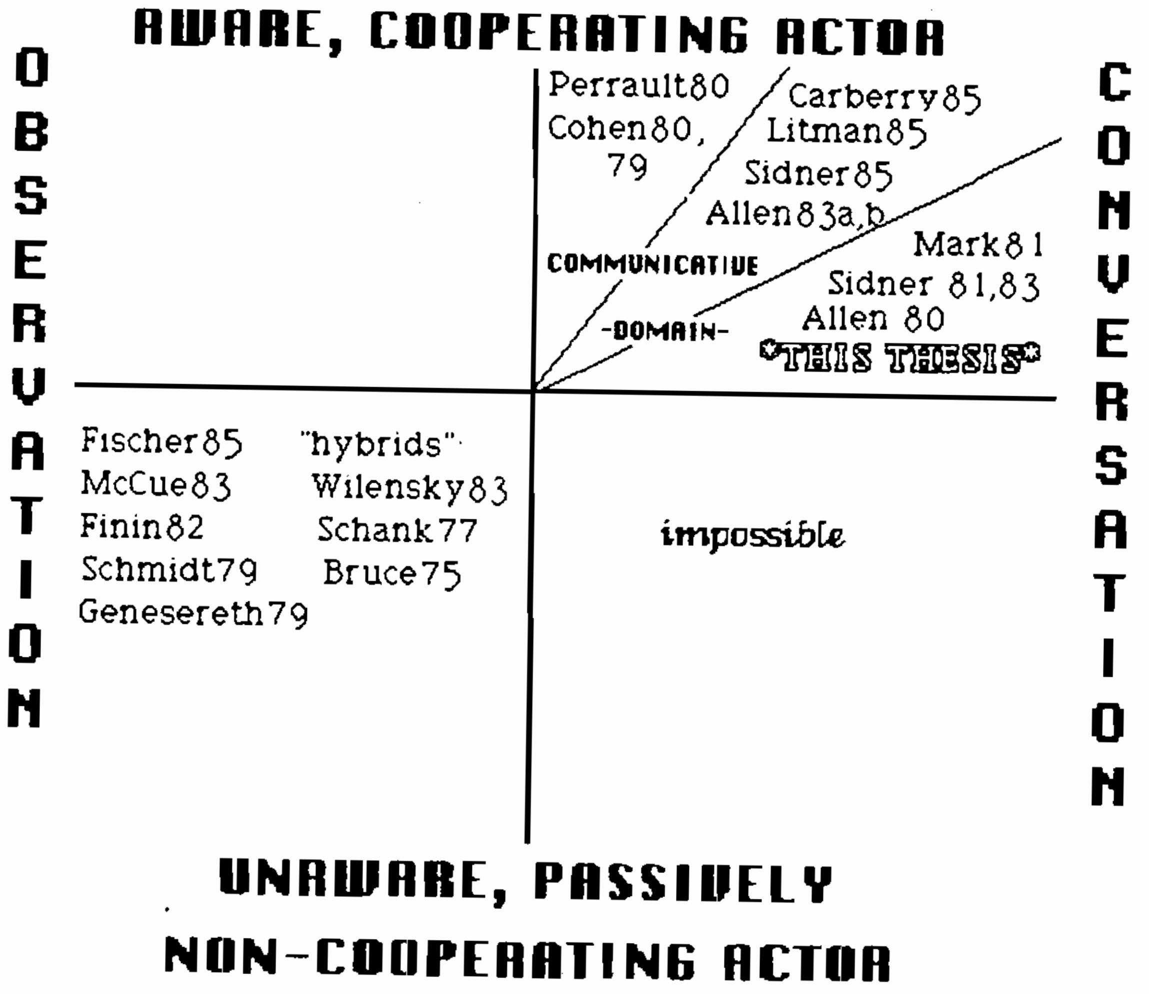


Figure 2-3: AI Research on Plan Inference

Notice that AI research on plan inference clusters into two of the quadrants: the lower left, representing the inference of an unaware actor's plans through observation, and the upper right, representing the inference of the plans of an aware, cooperating actor in conversation. The latter has been subdivided in the figure to distinguish between work on inferring communicative plans and work on inferring domain plans; as well, some research is shown as involving the inference of both. Notice also that this thesis is listed in the lower third of that quadrant: it concerns the inference of domain plans with a cooperating agent in conversation. Henceforth, unless I explicitly state otherwise, when I speak of plan inference systems or research in plan inference, I will be referring to that which would be classified in the upper right-hand quadrant and will similarly mean by the plan inference problem that which would be addressed by such research. We can now turn our attention to a closer examination of the methods that have been proposed in AI for plan inference of this type.

2.2. Existing Methods for Plan Inference

The Representation of Plans and Actions

The representation of plans and actions used in most plan inference systems, and in particular, in those that involve a cooperating actor in conversation, is a direct outgrowth of the representation first developed in the STRIPS system [Fikes 71], and later expanded in the NOAH system [Sacerdoti 77]. In these representations, each action α is modeled by an *operator* that may contain some or all of the following parts:

1. *a header*, which names α;

2. *a precondition list*, which describes what must be true for α to be performed;

3. *an effect list*, which describes what will be true after α is performed;

4. *a list of constraints*, which describes restrictions on legal instantiations of the operator[8] ; and

5. *a body*, which may be a set of subactions whose performance constitutes performance of α, or a set of subgoals whose achievement constitutes performance of α.

[8]For example, restrictions on the types of parameters, on the relations between parameters, and on the ordering of the subactions into which an action is decomposed. The distinction between constraints and operators was introduced by Diane Litman [Litman 84, p. 23]. Constraints can be thought of as preconditions that the planning agent never attempts to achieve.

Operators, which are also called action-schemas, may be parameterized: for example, one typical operator is

Header: PICKUP(x)
Precondition: ONTABLE(x) $\wedge$ HANDEMPTY $\wedge$ CLEAR(x)
Effect List: $\neg$ ONTABLE(x) $\wedge$ $\neg$ HANDEMPTY $\wedge$ HOLDING(x)

Each operator can thus actually represent a class of actions: in this case, the class of actions including picking up a ball, picking up the red block, and so on. A particular instance of an action in that class is represented by an *operator instance*, which is an operator along with a list of parameter bindings and a time specification; in a multi-agent situation, the agent of the action needs also to be specified. Unfortunately, much of the planning literature fails to maintain the distinction between actions and action instances carefully: operator instances are often written without the argument representing their agent, who, after all, is obvious in single-agent planning situations; and time of execution is often represented only implicitly, by the ordering of the operators with respect to one another.

Plan inference systems have access to a set of operators representing actions that can be performed in a domain; we can call the set of such operators the *operator library*. Operator libraries typically contain only operators that model valid domain actions.

Because plans, intuitively, are collections of actions whose performance in particular circumstances leads to some goal, it is not surprising that in systems that model actions with operators, plans are modeled as collections of operators. More specifically, they are modeled as directed acyclic graphs whose nodes may be operators or propositions; when a node is labeled with a proposition p, it should be interpreted as representing any action that would achieve p. Graphs, rather than linear orderings, are necessary, because plans have been seen as having both a hierarchical and a temporal dimension. The situation becomes further complicated by the fact that the temporal ordering of certain of the component actions of a plan may be indeterminate. Analogous to the distinction between actions and action instances is a distinction between plans and plan instances; the latter distinction has been no more carefully maintained than the former.

Yet another distinction that has been blurred within the literature on plan synthesis and on plan inference is that between plans and actions. Because an operator for an action can include a body, or decomposition, of subactions that will achieve the header action, the operator is often taken to be a plan for the header action. Sometimes a distinction is drawn between operators that contain decompositions, which are thereby said to represent plans, and operators that do not, which are thereby said to represent actions.

Several alternative formalisms for representing plans and actions are under development [McDermott 82, Allen 83c, Georgeff 85, Lansky 85]; they have not yet been put to extensive use within any plan inference work (but see [Kautz 85]). These recent models are being developed largely to handle issues such as multiple agents engaging in simultaneous actions, reactive planning, and the representation of continuous, nondiscrete actions. I will be using Allen's [Allen 83c] representation of action and time, which means that the plan inference process that I develop will be capable of handling plans with continuous actions. In principle, it will also be capable of handling plans with multiple agents engaging in simultaneous activity, although I leave consideration of such plans to future research.

Constructing Plan Graphs

The next question to ask is, How do the existing plan inference systems work? That is, how do they construct plan graphs that represent the inferred plan of some actor? Each of the major systems for plan inference in conversation, including those shown in the upper right-hand quadrant Figure 2-3[9], has a set of inference rules for building plan graphs. Each rule states the conditions under which a piece of a plan graph--a plan subgraph--can be constructed; constructing a plan subgraph corresponds to inferring part of a possible plan. The conditions for constructing a subgraph always refer both to plan subgraphs that have already been inferred as candidate representations of the actor's plan and to operators in the operator library. A typical rule, for example, allows a plan subgraph that includes the node α to be expanded by adding an arc from α to a new node β, if there is an operator whose header is β and which includes α in its body. Thus if the operator in the top part of Figure

[9]Strictly speaking, the Consul system differs from the other systems in that it encodes plans as pieces of KL-One semantie networks, and infers them by doing pattern matching on pieces of existing networks. However, what is encoded within Consul's semantic networks turns out to correspond closely to the inference rules that I describe. For example, one piece of the network encodes the rule that "a request for a user action with a particular result can be redescribed as a call to an operation whose effect is that result" [Mark 81,p. 378]; cf. Rule 1 in Figure 2-5 below.

2-4 were in an operator library, the expanded plan subgraph shown could be constructed from the initial subgraph shown.

```
                         Given:   OPERATOR

                  Header:  Display the current
                           message on the screen

                      Body:  Type 'TYPE .'

 Expand:  INITIAL SUBGRAPH        To:  EXPANDED SUBGRAPH

                              Display the current message
                                     on the screen
      Type 'TYPE .'                         ^
                                            |
                                            |
                                     Type 'TYPE .'
```

Figure 2-4: Constructing a Subgraph from an Operator

There are three basic conditions under which the various plan inference systems will construct a plan subgraph. These are shown in Figure 2-5. Each condition corresponds to one of the ways that actions and propositions can be related in a NOAH operator. Several of the plan inference systems have additional rules which apply only to nodes that encode information-seeking actions. For example, many systems will construct a link from a node encoding "finding out whether P" to one encoding "achieving P." This construction is meant to capture the intuition that if an agent want to know whether P, he may want P. Henry Kautz [Kautz 85] has shown how these rules, along with rules for handling nesting plan inference, can be viewed as special cases of the three relations shown in Figure 2-5.

Starting with some action or actions believed to be in the plan, a plan inference system will repeatedly apply its inference rules to construct candidate plan graphs until one of the candidates satisfies some termination condition. Depending on the details of the particular system, the resulting plan graph may only represent part of the actor's plan. For instance, imagine that β is believed

A plan subgraph with nodes α and β and an arc from α to β, i.e.

$$\begin{tikzcd}\beta \\ \alpha \arrow[u]\end{tikzcd}$$

can be constructed provided either α or β is already in the subgraph, and $\alpha\mathbf{R}\beta$ holds, where $\mathbf{R}$ is one of the following relations:

1. $\mathbf{R}$ = *causes*, i.e. β is on the effect list of α, so there is an operator in the operator library of the form

   ```
   Header: α
   Effects: . . . β . . .
   ```
   [10]

2. $\mathbf{R}$ = *is-a-precondition-of*, i.e. α is on the precondition list of β, so there is an operator in the operator library of the form

   ```
   Header:  β
   Preconditions:  . . . α . . .
   ```

3. $\mathbf{R}$ = *is-a-way-to*, i.e. α is part of the body of β, so there is an operator in the operator library of the form

   ```
   Header: β
   Body: . . . α . . .
   ```

Figure 2-5: Rules for Constructing Plan Graphs

[10]The ellipses denote the fact that β need not be the only member of the list.

to be in the plan, and α is inferred using Rule 3 (α *is-a-way-to* β). Further imagine that α has some sister actions on the body list of the β operator. Some plan inference systems will introduce those sister actions into the plan along with α, while others will not. In the latter case, however, the additional actions could easily be filled in by referring to the operators in the operator library.

In the literature on plan inference plans are viewed as mental attitudes. Thus Allen [Allen 83a] writes AW(P) to express the fact that A has a plan to achieve P. He then uses this notation in his plan inference rules; recall the example discussed in Chapter 1:

$$\text{SBAW(P)} \rightarrow_{i} \text{SBAW(ACT), if P is a precondition of ACT}$$

(the "precondition-action rule").[11] This rule represents the fact that if the system believes that the actor wants to make true some proposition P, then the system may come to believe that the actor wants to perform some action ACT of which P is a precondition. As was already noted, such a rule leaves unstated whether the system believes that P is a precondition of ACT, or whether the actor believes this. In practice, there is only a single set of operators, the operator library, and the rule can be applied so long as some operator therein with header ACT has P on its precondition list; this information is treated as mutual knowledge. The "SBAW" context is largely transparent to the reasoning process that is performed by Allen's system: the reasoning is all performed directly on its object, and the B and W operators are carried directly from antecedent to consequent in each inference rule. Without any repercussions, the B and W operators can be omitted, resulting in rules that are completely equivalent to those in Figure 2-5. In fact, in his example plan graphs, Allen often leaves off the B and W operators entirely, and this practice has been

[11] The subscripted arrow in the rule is not a logical connective; it indicates a nondeductive (or likely) inference. Allen distinguishes between nondeductive plan inference rules, which encode forward-chaining from some observed act and which are written with the symbol $\rightarrow_{i}$, and nondeductive plan construction rules, which encode backward-chaining from the goal and which are written with the symbol $\rightarrow_{c}$.

continued in more recent work in plan inference [Litman 84, Carberry 85].[12]

What is important to observe is that the plan inference rules shown in Figure 2-5 operate on the members of the operator library. This means that a subgraph joining two nodes X and Y will be constructed only if (i) X and Y are both encoded in the operator library; and (ii) they are encoded in the operator library in one of the configurations shown in Figure 2-5--i.e. they are related by one of the relations *causes, is-a-precondition-of* or *is-a-way-to*. Recall too that the operator library typically contains only valid domain information. Thus there are two further conditions that will be satisfied whenever a subgraph joining nodes X and Y is constructed: (iii) X and Y both model valid domain actions or propositions; and (iv) the relationship between X and Y that is encoded in the operator library actually holds between the actions or propositions they model. Since an inferred plan graph is a graph all of whose subgraphs satisfy Conditions (i) through (iv), as well as some additional conditions, the inferred plan itself will be composed only of valid actions and propositions that are included in the operator library and that are actually related to one another in ways represented there. No plan that is invalid, either in virtue of containing some nonrealizable action or proposition, or in virtue of improperly relating two actions and/or propositions, can be inferred; nor can any plan, valid or not, be inferred if it contains actions or propositions not in the operator library, or if it contains actions or propositions related to one another in ways not encoded in the operator library.

One significant extension to the standard methods of plan inference just discussed has been made by Litman [Litman 85], who uses a stack of plans in the inference process. The stack contains not just domain level plans, but also *meta-plans*, which model agents' behavior with respect to plans *per se*. Typical

[12] Actually, there seems to be some tension as to what is really meant by the W operator. Allen states that AW(P) means "A has a goal to achieve P," which seems to imply that P is a single action or property, not a whole plan. Consistent with this, he says that "SBAW(X) $\rightarrow_i$ SBAW(Y)" should be taken to mean that "if S believes A has a goal of X, then S may infer that A has a goal of Y" (p. 120). But he uses these rules to infer not just that A has a goal of Y, i.e., that his plan contains Y, but that it contains X and Y related to one another in some particular way specified by the rule. So the precondition-action rule considered above should probably be written "SBAW(P) $\rightarrow_i$ SBAW(P $\dashrightarrow_{\text{is-a-precondition-of}}$ ACT), if P is a precondition of ACT." Writing the rule this way would clarify his model, but it would not affect the claim that the B and W operators are transparent to the inference rules.

meta-plans are INTRODUCE-PLAN, CONTINUE-PLAN, IDENTIFY-PARAMETER(of a plan), and MODIFY-PLAN. The use of a stack of plans has been especially useful for explaining the inference of interleaved plans, which are often found in extended discourses; in particular, Litman has used her model to analyze interrupting subdialogues, such as requests for clarification. Robert Wilensky [Wilensky 83] also makes use of a form of meta-level plan, to analyze goal interaction. The plans that can be inferred by these systems, however, are still subject to the four constraints listed in the previous paragraph, since within each level of plans, the inference process uses rules like those in Figure 2-5.

2.3. Necessary Assumptions of Existing Models

Having seen how plans are inferred by existing plan inference systems, we can now ask the following question: what sort of assumptions are needed to guarantee that these systems' behavior is correct? Put otherwise, what are the conditions under which these systems will be able to infer an actor's plan correctly?

Before outlining the necessary assumptions, it is important to reiterate one of the design principles that is typically adopted by these systems, to wit, the decision to encode in the operator library only valid domain knowledge. I will call this the *principle of parsimony*. Since the operator library represents the system's knowledge, and the system is, in turn, the inferring agent, the principle of parsimony can be stated as follows:

Principle of Parsimony (PP)

the inferring agent does not have explicit knowledge of any domain information that it knows to be invalid.

The PP should not be confounded with a second assumption, *correct knowledge*, which states that the inferring agent is never mistaken:

Correct-Knowledge Assumption (CKA)

all the domain knowledge that the inferring agent has is valid.

Given that plan inference systems emulate domain experts, the CKA is not wholly unreasonable.[13] The PP, however, is obviously too strong: experts do

[13] But see the discussion below for the consequences of weakening it.

have knowledge of many typical misconceptions that can arise in their domain of expertise, upon which they can draw during the plan inference process. I will return to this point below. However, it is worthwhile first to ask what sort of assumptions are necessary to guarantee the behavior of systems observing the PP.

It turns out that two such assumptions are necessary. The first is the *closed-world assumption*, which, in general, is that anything not inferrable from a system's knowledge is assumed to be false [Reiter 78]. In terms of the problem at hand, it is more convenient to state the contrapositive of the closed-world assumption, i.e., that everything that is true is inferrable from the system's knowledge:

Closed-World Assumption (CWA)
the inferring agent has knowledge of all valid domain information.

That is, the operator library must contain representations of all valid actions in the domain and, furthermore, must contain representation of all propositions and actions that can be related to any encoded action. It may seem confusing to claim that all such information needs to be explicitly encoded in the operator library, when all that the CWA demands is that it be implicitly encoded in--i.e., inferrable from--the system's knowledge. However, recall that what existing plan inference systems infer are possible plans. There are no facilities for inferring new actions or new relations between actions or between actions and propositions. Hence all such information must be encoded explicitly.

Why is the CWA necessary? Without it, there can be cases in which the actor's plan, although valid, contains actions or relations between actions about which the inferring agent does not have knowledge; the system as inferring agent would necessarily fail to infer the plan in such cases. The CWA rules out such a possibility: if we assume that the system has knowledge of all valid domain information, then there are no valid plans that an actor can have that the system cannot, in principle, infer. However, while the CWA is enough to make the claim that the system will behave correctly whenever it is faced with a valid plan, it is still not enough to guarantee that it will always behave correctly. To make this claim, another assumption, which I will call the *valid-plan assumption*, is also necessary:

Valid-Plan Assumption (VPA)
the actor's plan is valid.

When this assumption is made, the system never needs to infer a plan that contains invalid domain information. Thus, since by the CWA the system is guaranteed to perform correctly whenever faced with a valid plan, making the VPA guarantees that the system will always be able to perform correctly, i.e. be able, in principle, to infer the actor's plan.[14]

Actually, the VPA as stated is slightly stronger than necessary. There is, in fact, a class of invalid plans that existing systems can infer. These are plans that are *valid* in the sense that all their components are known to the system and fit together in ways expected by the system, i.e., they satisfy the four requirements given at the end of Section 2.2. Their invalidity is a result of some proposition, upon which the plan's success depends, being false. In principle, existing systems could use Rule 3 of Figure 2-5 to infer a plan subgraph joining some actions α and β, without believing that the plan including that subgraph will succeed. This could occur if the system believes that one of the preconditions in the operator representing β will be false at the time the agent does α. Although such cases can, in principle, be handled, existing systems have not focussed on them. In fact, various control heuristics have biased the inference process against finding just such plans (see, e.g., Allen's Heuristic (H1) [Allen 83a, p. 127]). In Chapters 5 and 6 I will return to this type of invalidity and show where it fits in the classification of invalidities that I provide.

In question-answering, one important corollary of the VPA is the *appropriate-query assumption.* This is the assumption that

Appropriate-Query Assumption (AQA)
> the query being analyzed accurately requests information that the questioner needs to achieve his goal.

If, by the VPA, the questioner's plan is assumed to be valid (i.e., it is assumed that the questioner knows what he needs to do to achieve his goal) then it follows that his query can be assumed to be appropriate, (i.e., to reflect a need for information he truly requires to perform his plan, and consequently, achieve his goal). However, evidence from naturally occurring data--not to mention common sense--shows that the AQA is unwarranted: people seeking advice often ask for

[14]If the CKA is dropped, then it becomes possible that an invalid plan can be inferred if the invalidity happens to correspond directly to an invalid belief of the inferring agent; but in this case the fact that the plan is invalid goes undetected.

information that is not in fact appropriate to their goals. Thus, to construct a cooperative question-answering system, we need to give up the AQA; and because it is a direct corollary of the VPA, we need to give that up too.

This then is the point at which this thesis departs from existing research in plan inference: it suggests a way of "abandoning" the VPA, i.e., of designing a system that can infer plans even when those plans may not be valid. One approach to doing this might be to encode some typical invalid plans--i.e., to give up the PP. Several CAI systems, including [Brown 78, Stevens 79 and, Woolf 83] have done just this, encoding sets of erroneous beliefs that their users are likely to have. Genesereth's MUSER system [Genesereth 79] similarly contains an explicit library of erroneous plan fragments. Although this seems to be a useful strategy, it is necessarily incomplete. It is impossible for any person to have complete knowledge of the potential beliefs of other people, since the range of beliefs is, in principle, infinite. This means that system designers cannot anticipate *a priori* all potential misconceptions the users of their system may have. It also means that the intelligent agent that such systems emulate--the human being--cannot know *a priori* all potential misconceptions that people asking her questions might have. Sometimes she will have to deal with a novel (to her) belief. Fortunately, human beings seem to be quite "robust": good at dealing with novel situations. In fact it has been argued [Dennett 84] that such robustness is a definitive feature of high-level intelligence.

In this research, then, I have maintained the PP and developed an account of robustness in plan inference. Said otherwise, I have developed methods of reasoning from valid information to determine novel invalid plans.[15] In any complete reasoning system, the most reasonable strategy would probably be to encode typical invalid knowledge explicitly, and fall back onto a theory such as the one developed in this thesis whenever some invalidity not encoded there--that is, not anticipated *a priori*--is encountered.

What I will not be doing is suggesting how to abandon the CWA. Of course the CWA is also overly strong; however, once it is abandoned, then

[15]Of course, having such reasoning principles encoded means that, in some sense, the invalid plans are "implicitly" encoded, in the same way that a sentence of some language L is implicitly encoded by a grammar generating L. This is inevitable. What the PP requires is that the invalid information not be explicitly encoded. To continue the analogy with grammar, this is akin to requiring that all sentences of L not be listed explicitly.

whenever the inferring agent infers some novel-to-her plan, she needs to consider whether it is invalid or whether it is simply a valid plan that contains information she has not previously known. A similar consequence results if we abandon the CKA. If we allow the possibility that the inferring agent mistakenly believes that some invalid plans are valid, then, again, whenever she infers a novel plan, she will have to consider whether it is invalid or whether her own beliefs are invalid. Instead, in the model I develop, the inferring agent will always assume that all novel plans are invalid.

Note that abandoning the CWA and the CKA will not affect the way that a plan is inferred or the way in which an inferred plan is evaluated. What will change is the way in which the result of that evaluation is used.

Before concluding this discussion, it may be enlightening to consider things from a slightly different perspective. Analogies can be drawn between the CKA, the VPA, and claims of soundness and between the CWA and a claim of completeness. On this view, the CKA asserts that the inferring agent's knowledge is sound (with respect to "truth" in the "real world"), while the VPA can be seen to assert that the actor's knowledge is sound (again, using the "real world" as a model). The CWA asserts that the inferring agent's encoded knowledge is complete (once more, with respect to the "real world"). Then, since the inferring agent's knowledge is sound and complete, and the actor's is also sound (though not necessarily complete), it follows that the actor's knowledge is a subset of the inferring agent's. Hence, a search through the inferring's agents space of possible plans is guaranteed to find the actor's plan. We know, though, that the actor's knowledge is not necessarily sound, and it is this assumption that is abandoned in this thesis.

2.4. Explicitly Identified Goals

Under the VPA, when a question is asked that explicitly identifies the actor's goal, no plan inference needs to be done. Instead, an answer that directly addresses the identified goal is guaranteed to be appropriate. Plan inference, under the VPA and the corollary AQA, is needed for two purposes:[16]

[16] I am restricting my attention here to cases in which the questioner takes a single conversational turn. In extended discourse, it may also be useful to perform plan inference for the purpose of relating the utterances in the actor's successive turns; see [Litman 85, Carberry 85].

- to determine an unmentioned goal of the actor
- to determine what goal an actor has when his query mentions several acts or states he desires, but does not indicate how these acts and states are meant to be related.

The first of these is the usual purpose to which existing plan inference systems have been put. Using an example from Allen [Allen 83a], if someone asks "When does the train to Windsor leave?", the system can infer that his goal is to board the Windsor train; it then uses this inference to decide that it also should tell him the Windsor train's gate number.

The second of these purposes has not been noted in the literature. Yet there are queries--such as Examples (8) and (9) below--in which Q explicitly mentions his goal, but does not mark it as such.

(8) Q: "I want to talk to Kathy. I need to find out the phone number at the hospital."

(9) Q: "I want to talk to Kathy. I need to find out whether she wants to go to Maryland."

In Example (8), if Q believes that Kathy is at the hospital, then he probably intends to enable talking to Kathy by finding out the phone number in the hospital. On the other hand, if Q believes that Kathy is the only one in the office with a phone book, then he may intend talking to Kathy to enable finding out the phone number at the hospital. Example (9) is similarly ambiguous, its meaning depending in part upon whether Q intends to have his talk on the way to Maryland or to have his talk consist in finding out whether Kathy wants to go to Maryland. The linguistic form of the query does not by itself reveal what Q's goal is. Yet his conversational partner R needs to determine this to provide an appropriate answer, even if the VPA and AQA are made. For even if Q's plan is assumed valid, R still needs to know whether, in Example (8), to tell Q where Kathy is or what the hospital phone number is; in Example (9), she needs to know whether to tell Q again, where Kathy is, or whether Kathy wants to go to Maryland.

When the VPA and the AQA are abandoned, plan inference is still necessary for both these purposes. If Q asks "When does the train to Windsor leave?" when in fact there is no train to Windsor, but there is a train to Detroit and a ferry from there to Windsor, then giving a cooperative answer depends on determining that the questioner's unmentioned goal was to go to Windsor. And if Q asks the query in Example (8) above, then R will need first to determine whether Q intends to call Kathy at the hospital (as opposed to finding out the hospital phone number from Kathy) before she decides whether Q will in fact succeed in this plan, or whether the fact that Kathy has already been discharged will undermine his attempt.

But when the VPA and the AQA are abandoned, plan inference is also necessary to formulate appropriate responses to queries in which the goal is explicitly mentioned and indicated as such. Without inferring the underlying plan, it will be impossible to ensure that the response given does not mislead the questioner.[17] To see this, consider the following two queries:

(10) Q1: "How can I get mail to load only the new messages?
I figure this would speed up entry time into it."

(11) Q2: "How can I get my username changed? I figure this
would speed up entry time into mail."

Assume that in the mail system under discussion, entry time depends upon the number of messages loaded, and not upon username. Assume further that neither of the queried actions are performable, i.e., there is no way to have mail load only new messages, nor is there a way to have one's username changed. It seems that intuitively at least, a coherent plan behind Example (10) can nonetheless be inferred: if the queried action were performable, it would result in fewer messages being loaded to mail, and since entry time depends on the number of messages loaded, this would result in faster entry to mail. Thus an answer "You can't load only new messages, but you can clean out your mail file" is a valid· one. In contrast, there is no reason to believe that changing one's username would lead to faster entry into mail. An answer similar to that suggested for Example (10) would be unsuitable for Example (11), since it would mislead Q2 into believing that entry time does depend upon username. If, at a later date, Q2 learned that it had become possible to change his username, he might do so, intending to speed entry to mail. Similarly Q1 might later try to get mail to load only new messages, should that option become available. Q1, who later has mail load only new messages, will succeed in his goal, while Q2,

[17] And not misleading one's interlocutor seems to be a requirement on cooperative conversation; see [Joshi 82].

who has his username changed, will fail. A more appropriate answer to Example (11), given certain assumptions about the relevance, is something like "Why do you think that will speed entry to mail?" (i.e., "I can't figure out your plan").

These examples should recall the claim made in Chapter 1 that there are different kinds of invalidities that can be detected in plan inference. When the AQA and the VPA are not made--when there exists a possibility of an invalid plan underlying the query--then plan inference is necessary even when the actor is explicit in identifying his goal: it is necessary to determine how the intended actions are meant to contribute to the goal and to determine whether they will indeed do so, and if not, why they will not.

In this thesis, I will deal primarily with queries in which the goal is explicitly mentioned; usually I will also assume that it is explicitly indicated as the goal. The reason for doing this is that inferring an unmentioned goal seems to demand a great deal of world knowledge that circumscribes the set of "likely" goals. When the domain is quite constrained, this information can be easily encoded; thus, in the train-station domain that is so popular in plan inference work, the only likely goals are taken to be boarding a train and meeting a train. Without such knowledge about what set of goals are most likely in any given circumstance, the inference process quickly becomes mired in combinatorial explosion, since any one action can, in principle, be done with the intention of supporting any one of arbitrarily many other actions.

Examination of naturally occurring dialogues suggests that people asking questions are at least implicitly aware of such inherent difficulties in plan inference, and attempt to assist in the process by providing information that they believe relevant to it. The following three examples, taken verbatim from the mail transcripts described previously, illustrate this phenomenon:

(12) Q: " Is there any way to tell a bboard to load only new messages (I would like to speed up entry into infomac) ?"

(13) Q: "How can I scroll up and down in mail? I usually forget what number mail I'm looking at and if the info I need is off the screen, I can't get to it easily."

(14) Q: "[I]s there anyway to dynamically load a file of definitions and commands and have them interpreted by mail? I've wanted to do this so I could have a large file of net addresses and aliases that I would manually load if I wanted to send mail to one of them. I figgure [*sic*] this would decrease the startup time for mail, since I would take all the symbol defs out of my mailinit file."

In each of these queries, Q does explicitly mention his goal. In Example (12), he asserts that he wants to have bboard load only new messages in order to speed up entry time into it. In Example (13), he wants to scroll in mail in order to find message numbers that have moved off the screen. And in Example (14), he wants to load a file of definitions dynamically in order to enable another action--of taking symbols defs out of his mailinit file--that in turn will speed up entry time in to mail. In each case, despite the fact that Q tells R his goal, producing an appropriate response depends upon R's attempting to infer Q's plan.

CHAPTER III
Plans as Mental Phenomena

This chapter distinguishes between two views of plans: the data-structure view and the mental phenomenon view. It then draws on commonsense conceptions to develop a model of plans as mental phenomena. "Having a plan" is analyzed as having a particular configuration of beliefs and intentions. The beliefs and intentions are about the acts that play a role in the agent's plan. Playing a role in a plan is defined in terms of two relations over acts: generation and enablement.

3.1. The Data-Structure View and the Mental Phenomenon View

In his paper on the role of plans in a theory of practical rationality, Michael Bratman observes that there is an ambiguity in speaking of an agent's plan: "On the one hand, [this] could mean an appropriate abstract structure--some sort of partial function from circumstances to actions, perhaps. On the other hand, [it] could mean an appropriate state of mind, one naturally describable in terms of such structures" [Bratman 83, p. 271]. We might call the former sense the *data-structure view of plans*, and the latter the *mental phenomenon view of plans*.

Within AI, the data-structure view of plans is historically prior to the mental phenomenon view. As described in the previous chapter, work on plan synthesis, which preceded work on plan inference, considered plans to be data structures encoding aggregates of actions that, when performed in circumstances satisfying some specified preconditions, would achieve some specified results.[18] When the study of plan inference was begun, the mental phenomenon view of

[18]Statements of this view can be found in a number of places; see, in addition to the works discussed in the preceding chapter, [Nilsson 80, p.282, Fikes 71, p.190, Wilkins 83, p.733, Waldinger 81, p.251, and, CohenP 82, pp. 515-522.]

plans gained prominence since, after all, inferring another agent's plan means figuring out what actions he "has in mind." To make sense of the plan inference problem, plans had to be seen not merely as aggregates of actions, but as aggregates of actions to which an agent stands in some particular relationship--in other words, as mental phenomena.

And so they were, as the discussion of Allen's system in Section 2.2 illustrates. However, in developing an account of plans as mental phenomena, researchers like Allen drew heavily on the existing AI models of plans as data structures. Though "having a plan" was taken to describe a mental attitude, the objects of such an attitude were taken to be describable in terms of the data structures already developed in systems like NOAH. If an agent had some plan, that plan was constrained to be a collection of actions and propositions related by *causes*, *is-a-precondition-of*, and *is-a-way-to*, as defined in Figure 2-5--in other words, related in ways corresponding directly to the relations that can obtain between the actions and propositions represented in a NOAH operator. Further, nothing more was said about the state of "having a plan"; it was not analyzed into component attitudes of, say, believing and intending. For Allen, as well as for most of those whose work he inspired, "the properties of the WANT operator"--which is the the relation that holds between agents and their plans--"[are] completely specified by the . . . plan inference rules" [Allen 83a ,p. 117]. But the plan inference rules apply directly to NOAH-like operators. Thus the data-structure view of plans is not only historically prior to the mental phenomenon view in AI: it is also logically prior.

In this chapter, I will develop an account of plans that is first of all an account of the mental phenomenon of "having a plan." In particular, I will be concerned with describing the set of beliefs and intentions that are requisite to having a plan. Since it will turn out that the variance in beliefs between two agents is what results in one agent considering another's plan to be invalid, a careful account of the relationship between beliefs and plans is essential to a model of reasoning about invalid plans.

In developing the account of plans as mental phenomena, I will draw primarily upon our commonsense conceptions of what it means to have a plan. My discussion in this chapter is largely informal: that is, I here provide an overview in English, and leave until the next chapter the development of a representation that is sufficiently formal to support computation. In Chapter 4, I will also compare the standard NOAH-like models of plans with the model developed here, showing how the latter can subsume and make more precise the former.

3.2. Types of Plans

In the rest of this chapter, I will freely make recourse to our commonsense conceptions: to what we say or think about what it means to "have a plan". Just as we can speak of an agent having a belief that X, and of an agent believing that X, I will speak both of an agent having a plan to X, and of an agent planning to X, using the two locutions interchangeably: in both cases I am adopting a mental phenomenon view of plans. However, I am only concerned with capturing our conceptions about having plans to (do) β, where β describes some action. Thus, for instance, I will be concerned with plans to talk to Kathy, or to annoy one's cousin, or to meet the train from Windsor, or to speed up entry into the mail system. I restrict my attention in this way because it is this type of plan that underlies requests for information and advice.

Sometimes we say that we have a plan to q, where q is some state of affairs rather than an action. I might, for instance have a plan to be finished with my dissertation by next May, or to be the governor of California, or to have a million dollars by the time I am thirty. However, if I have a plan to be finished with my dissertation by next May, then I might be said to have a plan to achieve being finished with my dissertation by next May. Similarly, if I have plan to be the governor of California, then I have a plan to achieve being the governor of California. In general, if one has a plan to q, for arbitrary proposition q, then one can be said to have a plan to achieve q, and this qualifies as the type of plan that I intend to include in the model.[19]

[19]Strictly speaking, some morphological change may be necessary in going from the phrase *plan to q* to *plan to achieve q*. For instance, while I may plan to be the governor of California, I plan to achieve be*ing* the governor of California; similarly, I plan to be finished with my dissertation by next May, but plan to achieve finish*ing* my dissertation by next May. While the details of the morphology are irrelevant to the concerns of this thesis, it is interesting to note that a plan to achieve being α-ed X, though equivalent to a plan to be α-ed X, is not equivalent to a plan to α X. So for example, if I plan to be finished with my dissertation by next May, I also plan to achieve being finished with it by next May, but I do not necessarily plan to finish it by then: I might instead plan to win the Nobel prize, which will result in my committee automatically granting me my degree without my doing any further work on my dissertation.

The actions that one plans to do may be qualified in various ways: I may plan to go swimming, or I may plan to go swimming wearing a red swimsuit, or I may plan to go swimming at night. I may even plan to go swimming wearing a red swimsuit at night.

What types of plans am I then ruling out by the restriction to plans to do an action? First, I am ruling out the type of plan that one has in mind when she speaks of her plan for the day, or for next Friday evening, or for the next five years. "Next Friday evening" is not a proposition, and a plan for next Friday evening is not a plan to achieve next Friday evening. The English language itself gives us a clue that this is a different sort of plan, since it is somewhat more natural to speak of one's *plans* for next Friday evening.

Another type of plan that I am ruling out is what one means when he talks of the plan of the story or the plan for one's lecture. As with "next Friday evening," "the story" and "one's lecture" are not propositions, and it seems inappropriate to equate the plan for one's lecture with the plan to achieve one's lecture. Of course, one can have a plan to give a lecture, but that is an altogether different thing.

3.3. What is a Plan?

Having described the range of plans that will be of concern in this thesis, I will now consider our commonsense conceptions about what such plans consist in--i.e. I will consider what it means to *have a plan to do some action* β.[20] Consider first the plan I have to ask Kathy how she is feeling. I plan to do this, believing that Kathy is at the hospital, by finding out the phone number of the

[20]In the subsequent discussion, I will ignore several important issues of commitment over time, as discussed by Bratman [Bratman 86] and Cohen and Levesque [Cohen 85]. This omission is justified principally because, in the question-answering situations that will be of concern in this thesis, unexpected changes in the world that would force a reconsideration of the actor's intentions can be safely ignored. A refinement to the theory of intention, which could then be incorporated into the theory of plan inference developed here, should capture the requirements of commitment.

hospital, calling there, and then saying to Kathy "How are you doing?".[21] The performance of these acts is meant to "entail"--in a sense of entail yet to be further specified--the performance of my goal act.

One's plans, however, may fail. If, unbeknownst to me, Kathy has already gone home, then my plan will not lead to my goal of asking her how she is feeling. For me to have a plan to do β, that consists in the doing some collection of acts Π, it is not necessary that the performance of Π actually lead to the performance of β. What is necessary is that I *believe* that its performance will do so.[22] This insight is at the core of a view of plans as mental phenomena; on this view plans "exist"--i.e. gain their status as plans--by virtue of the beliefs of the person whose plan they are.

So far, then, I have associated the state of having a plan to do β with a belief that executing some collection of acts Π will lead to doing β. Note that the temporal ordering of the acts is an essential part of the plan: I may well have a plan to prepare onions for a sauce by chopping them and then sauteeing them--believing that by so doing I will perform my goal--and not have a plan that involves sauteeing and then chopping the onions. However, the ordering need not be total: as Earl Sacerdoti, working with a data-structure view, demonstrated [Sacerdoti 77], there are many plans which include acts whose temporal order with respect to one another is irrelevant. For example, I may have a plan to set the table that includes the acts carrying the flatware, plates, and glasses to the table, setting out the flatware, setting out the plates, and setting out the glasses. I may believe that it is essential that the first act be performed prior to the others, but may also believe that the other three acts can be performed in any order with respect to one another, and can even be interleaved. Of course, when I actually execute my plan, the acts I perform will be totally ordered with respect to one another. So there is a sense in which the beliefs that are part of my plan are partial.

---

[21]Throughout the rest of this chapter, I will make the simplifying assumption that when one dials the hospital, one reaches directly the person with whom one wants to speak. If the reader is uncomfortable with this simplification, the action of asking for Kathy can be inserted before the action of saying to Kathy "How are you doing?".

[22]In fact this condition may be slightly too strong: the agent need not be sure that performing his plan will entail performing his goal. In the normal state of affairs, though, he will at least think this likely and will act as if he believed it. See the further discussion below, on pages 43 ff.

There is also another sense in which my beliefs may be partial: they may concern acts only to an arbitrary level of abstraction. The set of acts that I believe will entail my asking Kathy how she is feeling includes the act of finding out the phone number of the hospital. It includes this despite the fact that I may not yet have considered how I will do this--for example, whether I will call information or look in a phone book. Such partiality will be particularly evident in question-answering situations: when an agent asks how to load only the new messages in order to speed up entry time into mail, his plan includes a belief that the former act will entail the latter, though it does not include a belief about what acts will entail the former.[23]

An agent G's belief that performing the acts in Π will entail performing β is not by itself sufficient to guarantee that G has a plan, consisting of doing Π, to do β. To see why not, consider the following scenario. Suppose I decide that while I am finding out the phone number at the hospital (say by looking in the phone book), I might as well at the same time find out the phone number of the bank (perhaps because I know I have to call there later to check on a wire transfer). As before, I believe that finding out and dialing the phone number at the hospital, and then saying certain words, will entail my asking Kathy how she is feeling. I do not believe that this will cease to be true if I also find out the phone number at the bank. Thus, there is a temporally ordered collection of acts Π, which equals {finding out the phone number at the hospital, finding out the phone number of the bank, calling the hospital, saying to Kathy "How are you doing?"}, such that I believe that executing Π will lead to my goal of asking Kathy how she is feeling.[24] However, it seems incorrect to say that my plan to ask Kathy how she is feeling includes my act of finding out the phone number at the bank; instead, this act is part of another plan (to check on my wire transfer) that I intend to interleave with my original plan. For an act to be included in my plan, I must believe that it plays a role in that plan; for the case at hand I do not believe that finding out the phone number of the bank plays any role in my plan to ask Kathy how she is feeling.

---

[23]Bratman [Bratman 86] discusses the significance of partiality of plans in resource-bounded agents like humans and robots.

[24]Notice that Π in this example need only be partially ordered: I may consider it irrelevant whether I first find out the phone number at the hospital and then the phone number at the bank, or vice versa.

Playing a Role in a Plan

What does it mean for an act to play a role in a plan? Consider once more my plan to ask Kathy how she is feeling. Part of this plan, I claimed, involved my calling the hospital. What would we say about my beliefs about these two acts: my asking Kathy how she is feeling and my calling the hospital? We might say that I believe that the latter will enable the former; i.e. that by calling the hospital I will establish a communication channel--a phone link--to Kathy, which will enable my saying something to her. Similarly, we might say that I believe that finding out the phone number at the hospital plays a role in my plan because I believe that doing the former will enable my calling the hospital, which itself plays a role in my plan. In general, then, if an agent believes that doing one act α will enable either his goal or some other act γ that plays a role in his plan, then α may play a role in his plan. In order to strengthen the "may" to "will," we need to consider the agent's intentions. I will discuss this presently.

Next consider the relationship between my acts of saying to Kathy "How are you doing?" and asking her how she is feeling. We would not say that I believe that the first act will enable the second. Instead we could describe my beliefs using the "*by* locution" in English: we could say that I believe that *by* saying "How are you doing?" I will be asking Kathy how she is feeling. Similarly, consider a slightly more detailed analysis of my plan to ask Kathy how she is feeling, which makes explicit the way in which I go about finding out the phone number at the hospital. One way that I might plan to do this is by looking it up in the phone book. We would not say that I believe that my looking up the hospital phone number will *enable* my finding it out: rather we would say that I believe that by looking it up I will find it out. Or I might, instead of looking up the phone number, plan to find it out by getting my office mate to tell it to me, a result of my asking her to tell it to me, which consists in my uttering the question "Do you know the phone number at the hospital?" Once again, we would not say that I believe that my uttering the question "Do you know the phone number at the hospital?" will *enable* my asking my office mate to tell me the phone number at the hospital, nor would we say that my asking her to tell me the phone number will *enable* my getting her to tell it to me. What we would say is that I believe that *by* uttering the question I will be asking her to tell me the phone number; that *by* asking her I will be getting her to tell it to me; and that *by* getting her to tell it to me, I will be finding it out. Notice that the *by*-locution does *not* completely correlate with causation: while it is true that asking to be told the phone number seems to relate causally to getting one's hearer to tell you the phone number, there is no such causal flavor to the relation between uttering the question and asking to be told the phone number.

The claim is that if an agent believes that by doing α he will be doing either his goal act β or some other act γ that plays a role in her plan, then α may play a role in his plan. Of course, to some extent this just begs the question, since I have left vague the conditions under which one act can be said to be done *by* doing another. In Section 4.3 I will show how the concept of *generation* developed by Alvin Goldman [Goldman 70] provides an answer to this. For lack of a more natural term, I will adopt immediately the term *generation* to describe the relation between two acts that we commonly express with the *by* locution. What is important to notice here is that there does seem to be an intuitive difference between the relation between acts that we describe as enablement and the one I am now calling generation. Just as we would not describe the relations between the acts discussed in the previous paragraph as enablement, so we would not, in general, use *by* to describe the relations earlier discussed as examples of enablement. We would not, for example, say that I believe that I can call the hospital by finding out the phone number there.[25]

What does the difference between enablement and generation consist in? Most importantly, it is the case that when one action α generates another action β, then the agent need only do α and β will automatically be done also. However, when α enables β, then the agent needs to do something more than α to guarantee that β will be done. I cannot simply find out the phone number at the hospital and then rationally expect that I will have called the hospital. But if I utter the words "Do you know the phone number at the hospital?" in the appropriate circumstances, then I need do nothing more to have asked my office mate the phone number at the hospital. And having done that, if the circumstances are right (e.g. my office mate knows the hospital phone number and is willing to tell it to me) , then I need do nothing more to have caused my office mate to tell me the hospital phone number.

Of course, if the circumstances are not right, then by asking my office mate for the phone number, I will not have caused her to tell it to me. For instance,

[25]This claim may be less clear for the acts of asking Kathy how she is feeling and calling the hospital: it seems possible to say that I plan to ask Kathy how she is feeling by calling the hospital. The ordinary-language test is a rough one, and occasionally fails to correlate with the phenomenal distinctions I want to draw.

she might not hear me ask her. In that case, I may repeat my question. Now it is important not to confuse things here. It is true that in this case, I *do* "do something more" than my original act of asking for the phone number in order to perform the act of causing my office mate to tell it to me. But here my first act--of asking for the phone number--does *not* generate my act of causing my office mate to tell it to me. My second act--of repeating my request--might, if my office mate hears me this time and responds, but then I have not "done anything more" than this second act, of repeating my question, to cause my office mate to tell it to me.

Analogously, I might dial the phone number and get a busy signal. I then need to do something more to establish a communication channel to Kathy. I might dial the number again, or I might drive over to the hospital. In either case, if I succeed in establishing a communication channel, I succeed in doing so by dialing the second time, or by going to the hospital: it is my act of *re*dialing or my act of going to the hospital, and not my original act of dialing, that generates my act of establishing the communication channel.

Beliefs and Intentions in Plans

So far we have seen that for an agent G to have a plan to do β that consists of doing Π, he must have a certain set of beliefs about the acts that Π comprises. Specifically,

> (1) G must believe that executing the acts in Π, in their (possibly partial) temporal order, will entail his performance of β

and

> (2) G must believe that each act α in Π plays a role in his plan, i.e. either he believes that by doing α he will do β or some other γ that plays a role in his plan (i.e. that α generates β or γ); or he believes that doing α will enable doing β or some other γ that plays a role in his plan.

Condition (1) can be thought of as a sufficiency condition: it guarantees that if G believes that α is part of what he will do to achieve his goal, then α is in the plan. Condition (2) can be thought of as a necessity condition: it guarantees that if α is part of G's plan, he believes it will play a role in achieving his goal.

Although these beliefs are necessary, they are not sufficient to guarantee that doing Π is G's plan to do β. It is also necessary that G have a certain set of intentions with respect to Π.[26] In particular, for my plan to do β to consist in my doing Π, I must intend to execute each of the acts in Π.[27] Let Π once more be {finding out the phone number at the hospital, calling the hospital, saying "How are you doing?"}. I may believe that executing Π would entail my asking Kathy how she is feeling, but if I intend instead to wait until next Thursday and then talk to her face to face, Π is not a plan I have to ask Kathy how she is feeling.

Further, in order for doing Π to count as my plan to β, not only must I intend to execute Π, but I must also intend it *as a way* of doing β; that is, I must intend to do the acts in Π in order to do β. Imagine that I am a teenager whose parents have forbidden me to stay out past midnight. Imagine further that there is some club, "The Mayday Late-Night Club," whose membership is limited to those who show up at the movies at 1 a.m. on May the first. Now let Π be the singleton set {going to the movies at 1 a.m. on May the first}. I may intend to execute Π, and I may believe that doing so will entail both becoming a member of the Mayday Late-Night Club and aggravating my parents. But if I intend my execution of Π as a way to do the former--and *not the latter*--then doing Π will count as a plan I have to join the Late-Night Club, but not as a plan I have to aggravate my parents. Notice that this is true even if I intend to aggravate my parent in some other way, say by getting a Mohawk haircut.

An argument similar to that made in the belief case also applies here. Thus, not only must I intend to execute Π in order to do β, I must also intend each act α in Π to play a role in my doing β; i.e., I must intend each α either to generate or enable β or some other γ that itself plays a role in my plan. Thus we can state three more conditions on G's having a plan to do β that consists in doing Π, namely:

[26]For the purposes of this thesis, I might almost as well have claimed that G has a certain set of *desires* with respect to Π; I will not draw heavily on the distinctions between desires and intentions. However, there are important such distinctions, and despite a number of proposals in the philosophical literature to reduce intentions to beliefs and desires [Audi 73, Beardsley 78, Davidson 80a, Goldman 70], Bratman [Bratman 83] has argued convincingly that such a reduction is impossible. He notes, amongst other things, that where desires, even when coupled with appropriate beliefs, are merely conduct-influencing pro-attitudes, intentions must be seen as conduct-controlling pro-attitudes if we are to make sense of the notion of plans.

[27]Conditional acts present special difficulties that need to be addressed in future research. If an agent intends, say, to take the bus if it rains and to walk otherwise, he intends to take the bus or to walk, but it is not obvious how to derive the latter from the former.

(3) G must intend to execute each act α in Π, in the (possibly partial) temporal order.

(4) G must intend to execute Π as a way of doing β.

and

(5) G must intend each act α in Π to play a role in his plan, i.e. either he must intend by doing α to do β or some other γ that plays a role in his plan; or he must intend by doing α to enable doing β or some other γ that plays a role in his plan.

The close parallel between Conditions (2) and (5) should lead us to ask whether one subsumes the other. The discussion so far has shown that having the beliefs defined in Condition (2) does not entail having the intentions in Condition (5). But does having the intentions in Condition (5) entail the beliefs in Condition (2)?

The question of whether an agent's intention to do α entails a belief that the agent will do α has been debated in the philosophical literature, and no consensus seems to have been reached. Where Grice [Grice 71] for one, answers in the affirmative, Donald Davidson [Davidson 80b] goes to great pains to provide a counterargument.[28] A telling comment on the controversy is provided by Bratman [Bratman 83] who notes that "plans normally support expectations of their successful execution. . ., [although] there may still be cases in which I plan to A but do not believe I will" (p. 286, footnote 4). Within this thesis, it will prove to be sufficient to assume this "normal" state of affairs and accept the view that if one intends to X, one must believe that one will.

[28] Davidson's case rests upon examples such as the following: I might intend to make ten legible copies of what I am writing by pressing hard on carbon paper, without believing with any confidence that I will succeed. Grice maintains that many such examples are actually elliptical versions of conditional intentions. Contra Grice, Davidson argues that such an intention is *not* an elliptical version of a conditional intention to make the ten copies if I can, for since one cannot intend to do what is impossible, intending to do X if one can is equivalent to intending to do X *simpliciter*; nor is it an elliptical version of some more detailed conditional intention to do X if, e.g., the carbon paper is particularly good, my hand muscles are more powerful than I thought, etc. His argument against this is that "there can be no finite list of things we think might prevent us from doing what we intend, or of circumstances that might cause us to stay our hand [Grice 71, p. 94]." This is obviously a description of the notorious "frame problem" that plagues AI.

Given this, we can see that Condition (5) directly entails Condition (2). For if, for each α in Π, G intends α to play a role in his plan, then G also believes that α will play a role in his plan. If G intends to do β or some other γ by doing α, he also believes that he will do β or this other γ by doing α; if he intends to enable β or some other γ by doing α, he also believes that he will enable β or this other γ by doing α.

Similarly, Condition (4) entails Condition (1). For if G intends to do Π as a way of doing β, then G must believe he will do β by doing Π, and this is exactly Condition (1).

To state the commonsense requirements on G's having a plan it is thus sufficient to state the conditions on intending--Conditions (3) through (5). However, while the conditions on belief--Conditions (1) and (2)--are entailed by Conditions (4) and (5) respectively, and are thus redundant, it is worth keeping them in mind, for it will turn out that when an inferring agent deems an actor's plan invalid, it is because she believes it includes invalid beliefs. In fact, it is even worthwhile to make explicit yet another belief that is requisite to having a plan. Condition (3) above asserts that G intends to execute each act α in Π; this then entails that G believes he will do each act α in Π. And this belief, in turn, entails a belief that G can do each act α in Π. Adding this condition to the definition, we can summarize the analysis of "having a plan" as follows:

(P0) An agent G has a plan to do β, that consists in doing some set of acts Π provided that

1. G believes that he can execute each act in Π.
2. G believes that executing the acts in Π will entail the performance of β.
3. G believes that each act in Π plays a role his plan.
4. G intends to execute each act in Π.
5. G intends to execute Π as a way of doing β.
6. G intends each act in Π to play a role in his plan.

CHAPTER IV
Representing Simple Plans

This chapter develops a formal representation for a restricted subset of plans: simple plans. Simple plans are defined to be those in which only generation--and not enablement--is believed to relate the constituent acts. A representation for actions is adopted. Goldman's generation relation is then discussed in some detail, and a representation for it proposed. Next executability is described, and finally a representation of the mental attitudes of belief and intention is adopted. The interaction between belief and intention is considered. All the pieces are then combined into a formal representation of the state of having a simple plan, and several examples of simple plans are given. The chapter concludes by reconsidering the standard model of plans, comparing it with the model developed here.

4.1. Toward a Formal Representation

The purpose of this chapter is to begin the development of a formal model of plans--i.e., a representation that not only captures the commonsense conceptions outlined in the informal definition, (P0), given at the end of the previous chapter, but also is sufficiently well defined to serve as the object of an automated reasoner. More specifically, the representation of plans developed here will need to be manipulated by an automated plan inference system. To develop such a model, I will make a simplification: I will focus developing of a representation of what I will call *simple plans*. An agent has a simple plan if and only if he believes that all the acts in that plan play a role in it by generating another act; i.e., if it includes no acts that he believes are related to one another by enablement. In this chapter, I develop the representation of simple plans, and in the next two chapters, I show how an automated reasoning system can infer and reason about possible invalidities in simple plans.

Formalizing Definition (P0) requires developing formal representations for several different types of things: actions; relations between actions--for the case of simple plans, the *generation* relation; a relation over actions--*executability*; and the mental attitudes of belief and intention. Each of these will be considered in turn in this chapter; in the process, I will build up a representation language accommodating all of them. Then the pieces will be combined into a formal definition of "having a simple plan." Finally, I will compare the model of plans developed here with the standard model discussed in Chapter 2.

4.2. Representing Actions

To begin, we need to observe a difference between types of action, such as picking up an object or calling the hospital, and actions themselves, such as Joan's calling the hospital yesterday afternoon. Actions can be thought of as triples of act-types, agents, and times. Any action may or may not occur; if it does, we need also to separate the action itself (the triple) from its occurrence. To keep straight this three-way distinction I will adopt the following terminology: I will use the term *act-type* to refer to types of actions--though sometimes I will also talk about types of acts, to avoid awkwardness. I will use the terms *action* and *act* interchangeably to refer to triples of act-type, agent, and time. And I will use the term *occurrence* to refer to realization of some action.

Because Goldman's account of the generation relation will be essential to the model of plans developed here, it is worthwhile to discuss the distinction he makes between *act-types* and *act-tokens*. For him, an act-type is "simply an act-property, a property such as mowing one's lawn, running, writing a letter, or giving a lecture. When we ascribe an act to an agent, we say that the agent exemplified the act-property (at a certain time)" (p. 10). An act-token, in contrast, is the exemplification of an act-type, so, for example: "John's moving his hand (at t) is a *token* of the type (or property) of moving one's hand" (p. 11). I am thus using the term act-type in the same way as does Goldman. His act-tokens correspond to what I term occurrences: they are real events in the world. For the most part Goldman is concerned with providing a theory of actual events in the world; his two-way distinction between types and tokens is thus sufficient and he does not need to consider actions (action-triples) in the abstract. (He does, however, run into some difficulties in attempting to account for intentional action, and needs then to resort to speaking of *hypothetical acts*.) In contrast, in developing a theory of plans, we will want to talk about the actions that agents intend to do--and not only are these not real events in the world at the time of the intention, but they may never be realized; their realization may even be impossible.

In what follows, I will not attempt to give a definition that carefully delimits act-types, distinguishing them from states (or state-types) or properties; or that distinguishes them from nonaction states or nonaction properties; or even that distinguishes one act-type from another. In the literature on the philosophy of action, the assumption is often made that there are some things that are commonly perceived as being types of actions; the enterprise then is to construct a definition D such that D is satisfied by all and only those things.[29] The philosopher's very assumption licenses my avoiding the question "What is an action?" I will simply concur with them and assume that any agent is aware of various types of actions, i.e., can say of certain concepts "That is a type of action."[30] This does not mean, however, that all agents will be aware of the same set of act-types. Different agents may be aware of (or know about or "have") different concepts, just as they may have different beliefs. I have a concept of "modem," which my grandmother does not. Similarly, the concept of "compiling a program" is one that I have and she does not. Although it sounds perfectly natural to say that an agent "has" a belief, it may sound a bit more awkward to say that she "has" a concept. However, for uniformity I will adopt this terminology, and will say of an agent that she "has" some concept C.

It is extremely difficult to say exactly what having a concept consists in: this is a central, if not *the* central, question of epistemology. Having a concept is not the same thing as believing in the reality of that concept: I have the concept "unicorn," although I do not believe that unicorns are "real." Furthermore,

[29] Lawrence Davis [Davis 79] provides a good overview of this literature.

[30] To assume that there is a set of act-types without attempting to provide conditions for membership in that set is quite consistent with the usual practice in constructing planning systems [Fikes 71, Sacerdoti 77, Wilkins 83] and in developing logics of action in AI [Rosenschein 81, Kautz 82, Moore 84]. But see Allen [Allen 84], which I will discuss below, and McDermott [McDermott 82], for some first steps toward stating such conditions.

It is clear that further refinements to the theory of properties implicit here would help resolve certain obvious questions about property identity which might naturally arise in the context of theorizing about actions and plans. In particular, questions about the uniform applicability of action properties across various categories of agents, e.g., the sense in which both humans and horses "run", would be most successfully addressed in this way (cf. Putnam [Putnam 79]).

although possible, it seems unlikely that agents ever have concepts without also having some beliefs about those concepts. What is hard to specify are any requirements on those beliefs. Consider, for instance, the case of act-type concepts. One sort of belief that an agent may have about a particular act-type is how to perform acts of it. This is not necessary, though: I have a concept of "developing film," but I do not know how to do it. Another sort of belief that an agent may have about an act-type is what the effects of performing acts of it are, but again, beliefs of this sort are not necessary: I have a concept of "pouring milk into the battery of my car," but I have only very vague beliefs about what the effects of such an act are. In fact it seems possible for an agent to have a particular act-type concept without having either of these kinds of beliefs: my mother, unlike my grandmother, does have a concept of compiling a program, but I suspect that she neither knows how to compile a program nor what the effects of such an act would be. Still, it is not the case that she has *no* other beliefs about this particular act-type: she believes, for instance, that it is something her daughter talks about, and that acts of it are done in the process of developing a program.

While having *some* beliefs about a concept C seems to be necessary to having C, it is extremely to difficult to say more about the nature of such beliefs. Yet having a belief about a concept C entails having C. Thus, when we model agents, we can say that the concepts they have consist of all and only the concepts about which they have beliefs.

Allen's Temporal Logic

To represent act-types, actions, and occurrences, I will make use of a formalism developed by James Allen [Allen 84]. This particular formalism, which is an interval-based temporal logic, differs from other popular formalisms in AI for representing actions, such as the situation calculus [McCarthy 69], in that it treats actions as things that are true--or that occur--over intervals of time. This treatment, as we will see, allows for a natural encoding of the generation relation, an essential feature of which is the simultaneous performance by an agent of two distinct yet closely related actions, each of which may have its own prerequisites and effects. The ability to express within the formalism assertions about arbitrary relations between intervals of time will also prove quite useful in a number of places. As one example, it will enable the natural representation of actions that consist in doing series of other actions, e.g., playing a C scale by playing a C, then playing a D, then playing an E, etc.

Allen's temporal logic is a typed first-order predicate calculus, which will form the core of the representation language used in this thesis. Two important types in the language are time intervals and properties. Terms of the former type denote, not surprisingly, intervals of time, though these may be arbitrarily small, so that what appears to be a time point will be treated as an extremely small interval. Terms of the latter type denote propositions that can hold or not during a time interval.[31] Another type in the language will be act-types, so there will be terms, like *pick-up* or *call*, denoting act-types. I will assume a number of other types, such as agents and objects. Note that there is not a type in the language for actions: actions are represented not with a single term, but with triples of terms of type act-type, agent, and time. Where necessary, I will be explicit about giving type restrictions, but I will often just assume appropriate types.

Of particular importance in this work will be two predicates from Allen's logic: HOLDS and OCCURS. HOLDS is a binary relation over properties and time intervals: HOLDS(p,t) is taken to be true if and only if property p holds throughout time interval t. OCCURS, in Allen's framework, is also a binary relation, over events and time intervals: OCCURS(α, t) is true if and only if event α occurs during time interval t. Events are a type in Allen's language, though, as I will discuss in the next paragraph, they will not be a type in the language I use. The primary difference between properties and events concerns the characteristics of subintervals of the intervals over which they hold or occur. If a property holds over a time interval t, it holds over all subintervals of t. So, if "the sun is shining" holds over the time interval "last Tuesday," it also holds over time intervals "last Tuesday morning," and "last Tuesday afternoon," etc. In contrast, if an event occurs over time interval t, then t is the smallest possible interval over which it could occur. If a robot moves a block from location X to location Y over interval t, it is not true that it moves it from X to Y over any subinterval of t. These properties about subintervals are built in to the

[31]Hence the use of the term *property* is somewhat nonstandard. In this account, *tall* is not a property, but *tall(sylvia)* is.

semantics of HOLDS and OCCURS.[32]

I will modify Allen's formalism slightly by making OCCURS a ternary relation. Where Allen folds the agent into the representation of the event, it will be representationally cleaner, for my purposes, to keep the agent separate from the type of action he performs. OCCURS(α,G,t) will be true if and only if agent G performs α during time interval t. The subinterval properties of OCCURS will be maintained: if OCCURS(α,G,t) is true, then OCCURS(α,G,t') is not true for any subinterval t' of t. Separating act-type terms from agent terms and making OCCURS ternary does result in an inability to deal with agentless events. I will not need this capability within this thesis, though if it were necessary, one could either introduce a dummy agent term or a separate, binary OCCURS predicate used only for modeling agentless events.

Thus act-types are terms in the language, and occurrences are represented as formulas. One common example of an act-type is picking up an object. To represent this in the language, a constant such as *pickup* can be introduced. Then, assuming *john* is a constant of type agent and *t* a constant of type time interval, OCCURS(pickup, john, t) describes an occurrence of a pickup action-- specifically, John's performing a pickup at time t. The three arguments to OCCURS thus describe an action. Other predicates, such as INT (which will represent intention) and GEN (which will represent generation) will also include among their arguments three that together describe an action. Actions may occur, agents may intend actions, actions may generate other actions. Actions have *performing agents* and *performance times*: for the action that is the argument to OCCURS(pickup, john, t), John is the *performing agent*, and t the *performance time.*

Act-Type Constructor Functions

One other feature of the representation of actions I am adopting should be discussed. Recall that act-types are represented by terms in the language. Act-type terms need not be constants (i.e., zero-ary functions); they may also arise via function application. I will refer to functions that construct act-type terms as

[32]Allen, following Mourelatos [Mourelatos 78], distinguishes a third class, processes, which are intermediate between properties and events. Processes may occur over subintervals of their time of occurrence, but need not occur over all subintervals. "Walking" is an example of a process. I will not need to make such a distinction, and will assume that all processes are continuous and, therefore, can count as properties.

act-type constructor functions. Act-type constructor functions can be used to encode any of the various ways in which act-type concepts are composed from other concepts.[33]

One example of an act-type constructor function might be *talk-to*, a function from terms of type agent to terms of type act-type. If *jane* is a term of type agent, which may denote the performing agent but more likely denotes some other agent, then *talk-to(jane)* will be a term of type act-type. Then, just as we can represent occurrences of acts whose types are represented with constants in the language--e.g., with OCCURS(pick-up,john,t)--we can also represent occurrences of acts whose types are represented with more complex terms--e.g., OCCURS(talk-to(jane), john, t). Another example might be *achieve*, a function from terms of type property to terms of type act-type; if *on(table,block)* is a property term, *achieve(on(table,block))* would be an act-type term.

An important class of act-type constructor functions are those that map act-type terms into act-type terms. An example of this is *undo*. Assume that *type(control-Z)* is an act-type term. (It is itself the application of the act-type constructor function *type* to the term *control-Z*.) Then another act-type term in the language would be *undo(type(control-Z))*.

Act-type constructor functions of several arguments are also permissible; I will abbreviate in the usual way a function from items of type X to a function from items of type Y to items of type Z, as a function from items of type X cross items of type Y to items of type Z. Hence, *but* might map from act-types cross properties to act-types: *but(move(X), ~deleted(X))* would represent something like the act-type of moving, say a mail message, to another file but not causing it to be deleted from the original file.

Act-type constructor functions will play an important role in explaining how people can reason about types of acts they have not encountered before. Imagine an agent Q talks about planning to "gloth" where "gloth" is an act-type concept he has but not one another agent R has. Since having a concept C was equated above with having some beliefs about C, this means that Q has beliefs about the concept "gloth," but R does not. In this case, R will have a very hard

[33] A precise account of what can count as an act-type constructor function thus must await a precise account of what can count as an act-type.

time understanding Q. However, if Q talks about "going to Szeged," then if Q and R both have the concept represented by the act-type constructor function "go-to," R will understand a great deal about Q's plans, even if she has never before heard of Szeged. Or to take a more extreme example, imagine that Q talks about planning to "gloth fewer spurvs." If R has the concept represented by the act-type constructor function do-to-fewer then she may understand a lot about Q's plan. For instance, she may realize that Q plans to "gloth only the spurvs marked 'T'" as a way of "glothing fewer spurvs" without knowing either what it means to gloth or what a spurv is.[34]

4.3. The Generation Relation

Recall that in Chapter 3, *generation* was described as one of the two relations that hold together plans. The generation relation was defined by Alvin Goldman in his book *A Theory of Human Action* [Goldman 70], which is an attempt to refute the *identity thesis* of philosophical action theory. This thesis, put forth by Donald Davidson [Davidson 80a], G.E.M. Anscombe [Anscombe 58] and others, maintains that very often there are widely diverging descriptions of the same act, so that, for example, if John gives his chess opponent a heart attack by checkmating him by moving his queen to king-knight-seven by moving his hand, then John has performed *one* act, of which four distinct descriptions have been given. Goldman sees several problematic consequences of the identity thesis. For instance, if John's giving his chess opponent a heart attack is the same act as John's moving his hand, then the conditions that make each possible ought to be identical. But his opponent's having a weak heart seems to be a factor in making possible John's giving him a heart attack, but not a factor in making possible John's moving his hand. Goldman thus defines *generation* as an alternative to identity in explaining the relation between the acts in examples like this one.

In describing the generation relation, I will follow closely Goldman's presentation: first I will discuss some of its properties; next describe various types of generation; and finally present Goldman's complete definition. Concurrent with this discussion, I will show how to encode the generation relation in the language of the interval-based temporal logic.

[34]Notice that in this example, R's understanding is helped by the fact that she recognizes that "gloth" functions syntactically as an verb. Without this clue, R might mistake "gloth" for an adverb, taking the construction to be parallel to "many fewer spurvs".

Properties of the Generation Relation

When one considers pairs of acts whose relationship to one another is intuitively supposed to be explained as generation, several properties of the generation relation emerge. First, generation is not a relation between act-types *per se*. John checkmates his opponent by moving his queen to king-knight-seven at some particular occasion: moving one's queen to king-knight-seven certainly does not always entail checkmating one's opponent. Or to give another example, Helen's flipping the light-switch may generate her turning on the light in her bedroom on Tuesday morning, but her husband's flipping the same switch on Tuesday afternoon may not generate his turning on the light if, for instance, the bulb has burned out in the interim. Generation is a relation between actions--triples of act-type, agent, and time--which further depends upon properties that hold during the performance time.

Some detail hidden in this last statement should be made explicit. Whether one act generates another depends in part upon the context of their potential occurrence. The action <flip-switch, Helen, Tuesday 8 a.m.> only generates the action <turn-on-light, Helen, Tuesday 8 a.m.> if, amongst other things, the light bulb in Helen's lamp is working on Tuesday at 8 a.m. Thus when we reason about whether generation holds between two unrealized acts, our reasoning will depend upon whether we believe that certain properties will hold at the performance time of the acts.

In Goldman's definition, generation is a relation over act-tokens, which, as we have already seen, correspond to occurrences of actions in the theory I am developing. Because he was primarily concerned with providing a theory of acts that *have* occurred, defining generation in this way was sufficient. However, in developing of a theory of plans, we will need to be concerned with the generation relation holding over acts that have not yet occurred, and which, in fact, may never occur. I will thus treat generation not as a relation over occurrences, but as a relation over acts--albeit acts envisioned as having certain contexts of potential occurrence, in light of the discussion in the previous paragraph. In fact, Goldman makes a similar move when he considers intentional action: he then speaks of generation holding over "hypothetical" acts (pp. 56-63). So Goldman first defines generation as a relation over act-tokens, which are realized actions--i.e., actions that have occurred and, therefore, are necessarily in the past--and then also allows it to apply to hypothetical actions, which are as yet unrealized actions with performance times in the future. It is simpler to consider generation all along as a relation over acts.

Returning to the properties of generation, we can next observe some of its algebraic properties. It is asymmetric: though John checkmated his opponent by moving his queen to king-knight-seven, he did not move his queen to king-knight-seven by checkmating him. It is irreflexive: John did not checkmate his opponent by checkmating him[35]. And it is transitive: if John gave his opponent a heart-attack by checkmating him by moving his queen to king-knight-seven, then he gave his opponent a heart-attack by moving his queen to king-knight-seven. In short, the generation relation is a partial ordering over certain sets of actions.

Third, the performing agent of two actions related by generation must be the same. Somewhat less obviously, their times of performance must also be identical. As Goldman explains,[36]

> ...neither one of a pair of generational acts is *subsequent* to the other. Let us say that G's doing β is *subsequent* to G's doing α if and only if it is correct to say that G did α *"and then"* (or *"and later"*) β. If G shaves at 8:00 o'clock and eats breakfast at 8:15, then his eating breakfast is subsequent to his shaving. . . On the other hand, if G checkmates his opponent by moving his queen to king-knight-seven, his checkmating his opponent is not subsequent to his moving his queen to king-knight-seven. It would be incorrect to say that G moved his queen to king-knight-seven "and then" checkmated his opponent, for that would wrongly suggest that it took an additional move to perform the checkmate. Similarly, if G turns on the light by flipping the switch, his turning on the light is not subsequent to his flipping the switch. His turning on the light is not subsequent to his flipping the switch even if the light does not go on until a few seconds after the switch is flipped. (Imagine a delaying mechanism in the switch.) Although the light does not go on until a few seconds later, it would still be incorrect to say that he flipped the switch and then turned on

[35]There may be a sensible reading of "John checkmated his opponent by checkmating him," but it is not analogous to "John checkmated his opponent by moving his queen to king-knight-seven." One way in which the two differ is that the latter, but not the former, provides an explanation of how John did the checkmating. See Goldman, p. 50.

[36]I have renamed some of Goldman's variables to make the quotation consistent with respect to the variables used elsewhere in this thesis. I will continue the practice of renaming his variables where necessary without further mention.

> the light. Likewise, although the opponent's heart attack begins several moments after the checkmate, it would be inappropriate to say that G checkmated his opponent "and then" gave him a heart attack. (p.21)

I will introduce the four-place relation symbol GEN to denote generation. $GEN(\alpha,\beta,G,t)$ will be true if and only if the act of G's performing α at time t generates the act of G's performing β at time t. When $GEN(\alpha,\beta,G,t)$ is true, I will say that $<\alpha,G,t>$ and $<\beta,G,t>$ form a *generational pair*. Below, I will say a good bit more about when two acts do in fact form a generational pair, i.e., about when $GEN(\alpha,\beta,G,t)$ is true for particular α, β, G and t.

Note that the GEN relation implicitly captures the fact that the performing agent must be the same for both members of the generational pair, as must the performance time. The algebraic properties of generation can be captured in the following axioms:[37]

(G1) $GEN(\alpha,\beta,G,t) \rightarrow \neg GEN(\beta,\alpha,G,t)$
(Asymmetry)

(G2) $\neg GEN(\alpha,\alpha,G,t)$[38]
(Irreflexivity)

(G3) $GEN(\alpha,\beta,G,t) \wedge GEN(\beta,\gamma,G,t) \rightarrow GEN(\alpha,\gamma,G,t)$
(Transitivity)

By no means, however, do (G1) through (G3) define generation. For one thing, they do not encode sufficient conditions, only necessary ones. They fail to encode, for instance, the essential fact that if G's performing α at t generates his performing β, the former results automatically in the latter. Below, I will give Goldman's complete definition of generation. First, however, I will briefly describe two distinct types of generation.

[37]Throughout, all variables should be taken to be universally quantified with widest possible scope, unless otherwise noted.

[38](G2) is a consequence of (G1), and is listed separately just for emphasis.

Types of Generation

While Goldman distinguishes between four types of generation--*causal*, *conventional*, *simple*, and *augmentation*--it is convenient to follow the suggestion of Beardsley [Beardsley 78, p. 165] and collapse the last three of these into a single category he calls *sortal* generation.

Let me once more use Goldman's own words to describe his theory. He says that "[a]ct-token α of agent G causally generates act-token β of agent G only if (a) α causes some [event] E and (b) β consists in G's causing E" (p.23). So, if G's flipping the switch causes the light to go on, then G's flipping the switch causally generates G's turning on the light since the latter can be analyzed as causing the light to go on. Similarly, if G's closing the door results in a fly being kept outside, then his closing the door causally generates his preventing a fly from entering the house, since the latter can be analyzed as causing a fly to be unable to enter the house.

In contrast, in *sortal generation*, there is no event E related to the generated occurrence that is *caused* by the generating occurrence. Instead, performing the generated act simply consists in performing the generating act. We have already seen one example of sortal generation: John's checkmating his opponent by moving his queen to king-knight-seven. There is no event that is caused by his moving his queen such that John's checkmating his opponent consists in that event. Other examples of sortal generation given by Goldman include

- G's jumping 6 feet 3 inches sortally (for Goldman, simply) generates G's outjumping George (where George has just jumped 6 feet);

- G's coming home after midnight sortally (for Goldman, simply) generates G's breaking his promise (where G has promised to be home by midnight);

- G's extending his arm out the car window sortally (for Goldman, conventionally) generates G's signaling for a turn (where G is driving his car and approaching a corner).

Again, it is essential to keep in mind that what is being mentioned in these examples are acts, not act-types. It is not the case that whenever an agent comes home after midnight he will, by so doing, break his promise: rather, certain agents coming home after midnight on certain occasions will, in so doing, break their promises. But whenever an act of α generates an act of β, there are regularities that can be observed to hold of the act-types α and β themselves. Specifically, whenever acts of α and β form a generational pair, there are certain conditions C such that any time an act of α occurs while C holds, the simultaneous occurrence of an act of β will be guaranteed. For example, whenever an act of α = "coming home after midnight," occurs when the performing agent has promised to be home at midnight, a simultaneous act of β = "breaking one's promise," performed by the same agent, is guaranteed. More generally, the occurrence of any action α will guarantee the simultaneous occurrence of "breaking one's promise" if at the performance time it is true that the performing agent has promised to do some γ such that doing α, in conjunction with certain facts of the context in which α occurs, entails not doing γ. Similarly, any occurrence of the action α = "jump six feet three inches" at a time at which George has just jumped six feet will be accompanied by an occurrence of the action β = "outjump George"; this can also be generalized in the obvious way. The existence of such regularities is essential to the complete definition of the generation relation, to which we now turn.

The Definition of Generation

Goldman's complete definition of generation follows[39]:

Act-token α level-generates act-token β if and only if

1. α and β are distinct act-tokens of the same agent that are not on the same level;

2. neither α nor β is subsequent to the other; neither α nor β is a temporal part of the other; and α and β are not co-temporal;

3. there is a set of conditions C such that

   a. the conjunction of α and C entails β, but neither α nor C alone entails β;

[39]Goldman's use of the term *level*-generation derives from the way in which he diagrams related acts. In this thesis, I use the shorter term *generation*. I will discuss the remaining undefined technical terms in his definition--*same level, subsequent, temporal part, and co-temporal*--later in this section.

b. if the agent had not done α, then he would not have done β

c. if C had not obtained, then even though G did α, he would not have done β. (p. 43)

While Goldman is defining the generation relation between act-tokens, his definition rests on an implicit relation that holds between act-types. The critical clause in his definition is Clause (3), which defines the requirements on what I will call the *generation-enabling conditions* C. These conditions are the regularities of context noted previously. What Clause (3a) encodes is the fact that not *all* occurrences of α entail simultaneous occurrences of β; and, similarly, not *all* instances of C (i.e., time intervals in which C holds) entail simultaneous occurrences of β. Clause (3a) thus precludes either α or C from being identical with β. While the "α" and "β" in the header to, and Clauses (1) and (2) of, Goldman's definition refer to act-tokens (and can, more generally, refer to actions), in Clause (3a) they refer to act-types. Their status in Clauses (3b) and (3c) is less clear, but I will suggest below that these two clauses are, in any event, less important.

So, when an act of G's doing α at time t generates an act of his doing β at t time, there are certain conditions C such that any time there occurs an act of α while C holds, there will also occur a simultaneous act of β. The regularity of the generation-enabling conditions C is what enables us to reason about whether by doing some action we will do another action, and consequently what enables us to construct and reason about (simple) plans. Such regularities will be encoded in the representation language in axioms of the following form, where α and β are specific act-type terms (i.e., not variables), and C is a specific proposition term (again, not a variable):

$$\begin{aligned}
&\forall G \forall t_1 [[\mathrm{HOLDS}(C,t_1) \wedge \mathrm{OCCURS}(\alpha,G,t_1)] \rightarrow \mathrm{OCCURS}(\beta,G,t_1)] \wedge \\
&\exists G \exists t_2 [\mathrm{OCCURS}(\alpha,G,t_2) \wedge \neg \mathrm{OCCURS}(\beta,G,t_2)] \wedge \\
&\exists G \exists t_3 [\mathrm{HOLDS}(C,t_3) \wedge \neg \mathrm{OCCURS}(\beta,G,t_3)]
\end{aligned}$$

Axioms of this form will be so important that it will be useful to have a means by which to refer to them readily. I will thus introduce the following abbreviatory device: I will say that act-type α *conditionally generates* act-type β under conditions C, and will write CGEN(α,β,C), whenever there is an axiom of this special form relating α, β, and C. I will also refer to an axiom of this form as a *cgen-axiom*. Thus the CGEN predicate is defined as follows:[40]

Definition
(C1) $\mathrm{CGEN}(\alpha, \beta, C) \equiv$
(i) $[\forall G_1 \forall t_1[[\mathrm{HOLDS}(C,t_1) \wedge \mathrm{OCCURS}(\alpha,G_1,t_1)] \rightarrow \mathrm{OCCURS}(\beta,G_1,t_1)] \wedge$
(ii) $\exists G_2 \exists t_2[\mathrm{OCCURS}(\alpha,G_2,t_2) \wedge \neg \mathrm{OCCURS}(\beta,G_2,t_2)] \wedge$
(iii) $\exists G_3 \exists t_3[\mathrm{HOLDS}(C,t_3) \wedge \neg \mathrm{OCCURS}(\beta,G_3,t_3)]]$

Consider some specific cgen-axiom, i.e., assume that α' and β' are particular action terms and C' is a particular proposition such that CGEN(α',β',C'). Let us consider each conjunct of the axiom. The first says what we would expect, namely that any time an act of α' occurs when C' holds, an act of β' will also occur. The second conjunct says that an occurrence of an act of α' is not by itself sufficient to guarantee an occurrence of an act of β'; the third says that neither are the generation-enabling conditions sufficient to guarantee an occurrence of an act of β'. Note that a cgen-axiom is not just a biconditional of the form:

$$\mathrm{HOLDS}(C',t) \wedge \mathrm{OCCURS}(\alpha',G,t) \equiv \mathrm{OCCURS}(\beta',G,t)$$

because in fact it is not necessary, when α' conditionally generates β' under C', for OCCURS(β',G,t) to entail either OCCURS(α',G,t) or HOLDS(C',t). There may be other act-types beside α' that conditionally generate β': i.e., there may be more than one way to do β'. And if, say, γ' is an alternative to α' as a way of doing β', the conditions under which an act of γ' generates an act of β' need not be C'.

Bratman [Bratman 78] has observed that some restriction must be placed on the generation-enabling conditions C lest Goldman's Condition (3) collapse into triviality: for instance, C cannot just be 'A $\rightarrow$ B'. However, C can be a sentence that expresses temporal information. So, for example, the generation-enabling conditions for the pair of act-types "come home after midnight," "break one's promise" express the fact that the performing agent, prior to the time of performance, made a promise of a certain kind.

[40]Throughout the text, I will explicitly mark axioms and definitions as such. All other labeled formulas, such as (G1) through (G3), are theorems. The proof of (G1) through (G3) will be given later in this section. Appendix A lists all the axioms and theorems presented in the text.

GEN can now be defined in terms of CGEN as follows:

Definition
(G4) $GEN(\alpha,\beta,G,t) \equiv \exists C(CGEN(\alpha,\beta,C) \wedge HOLDS(C,t))$

which can be expanded as:

Definition
(G4--expanded version)

$GEN(\alpha,\beta,G,t) \equiv$
(i) $\exists C[\ \forall G_1 \forall t_1((HOLDS(C,t_1) \wedge OCCURS(\alpha,G_1,t_1)) \rightarrow OCCURS(\beta,G_1,t_1)) \wedge$
(ii) $\exists G_2 \exists t_2(OCCURS(\alpha,G_2,t_2) \wedge \neg OCCURS(\beta,G_2,t_2)) \wedge$
(iii) $\exists G_3 \exists t_3(HOLDS(C,t_3) \wedge \neg OCCURS(\beta,G_3,t_3)) \wedge$
(iv) $HOLDS(C,t)]$

What Definition (G4) encodes is that fact that two acts will form a generational pair if and only if there is some generation-enabling condition relating the act-types they involve, and further, that condition holds at their performance time. Even if the condition expresses temporal information, that information must be evaluated, and indeed initially stated, with respect to the intended performance time of the generational pair. It is true now that G will break his promise by coming home after midnight if prior to coming home he makes the necessary promise, even if he has not now made the promise. Note too that Definition (G4) does not require that the acts in the generational pair actually have to have occurred, or will occur. That is, from Definition (G4) we can derive:

$$GEN(\alpha,\beta,G,t) \rightarrow [OCCURS(\alpha,G,t) \rightarrow OCCURS(\beta,G,t)]$$

but *not*

$$GEN(\alpha,\beta,G,t) \rightarrow [OCCURS(\alpha,G,t) \wedge OCCURS(\beta,G,t)].$$

To see why this is as it should be consider the following. It may well be true that if you walked over to the light switch right now and flipped it, you would, by doing that, turn off the light. If so, then GEN(flip-switch, turn-off-light, you, now) is true when evaluated in "the real world." But it does not follow from this fact that you *will* flip the switch or turn off the light; OCCURS(flip-switch, you, now) and OCCURS(turn-off-light, you, now) are not consequences of the instance of the generation relation.

Although generation is asymmetric, so that two acts cannot generate one another, there is no injunction against the same two act-types conditionally generating one another. That is,

$$\mathrm{CGEN}(\alpha,\beta,\mathrm{C}) \wedge \mathrm{CGEN}(\beta,\alpha,\mathrm{D})$$

is not inconsistent unless C equals D. (If C were to equal D, then the derived generation relation that held between acts involving α and β would fail to be asymmetric.) So, for instance, the act-type flipping the switch conditionally generates the act-type turning on the light under certain conditions, e.g., the power's being on, the bulb's working, and so on. But it is also possible for the act-type turning on the light to conditionally generate flipping the switch under other conditions, such as there being some kind of reverse wiring between the light and the switch. The fact that $\mathrm{CGEN}(\alpha,\beta,\mathrm{C}) \wedge \mathrm{CGEN}(\beta,\alpha,\mathrm{D})$ is not inherently inconsistent is what will allow the proper treatment of ambiguous cases, such as those discussed in Section 2.4, in which each member a pair of acts might be construed to generate the other. Although the examples there--for instance of finding out the phone number in order to talk to Kathy, or of talking to Kathy in order to find out the phone number--involve plans with enabling actions, one can construct similar examples with simple plans, such as one involving flipping the switch and turning on the light.

A Comparison of the Definitions

We can now reconsider Goldman's definition and show how it is captured by Definition (G4). I will consider each clause of Goldman's definition in turn. First, his Clause (1) contains three requirements: that the performing agents of the acts in the generational pair be the same; that the acts be distinct; and that they not be "on the same level." As already mentioned, the first requirement is implicitly encoded in the GEN operator itself. A straightforward argument can be given that the second requirement is entailed by Definition (G4): Assume that $\alpha = \beta$ and that $\mathrm{GEN}(\alpha,\beta,\mathrm{G},\mathrm{t})$. Then by Clause (ii) of Definition (G4) there is some agent G' and some interval t' such that $\mathrm{OCCURS}(\alpha,\mathrm{G'},\mathrm{t'}) \wedge \neg \mathrm{OCCURS}(\beta,\mathrm{G'},\mathrm{t'})$. But since by assumption $\alpha=\beta$, this means that there is a G' and a t' such that $\mathrm{OCCURS}(\alpha,\mathrm{G'},\mathrm{t'}) \wedge \neg \mathrm{OCCURS}(\alpha,\mathrm{G'},\mathrm{t'})$, an obvious contradiction. Put simply, if α is identical with β, then of course an occurrence of α entails an occurrence of β.

The third requirement of Goldman's Clause (1), that the acts in the generational pair not "be on the same level" is intended to rule out acts that differ merely in containing different individual concepts of a single object--e.g.

"hitting the tallest man in the room" and "hitting the mayor," where the mayor is the tallest man in the room. I have not explicitly ruled this case out in my definition, for doing so in a principled fashion requires a solid theory of linguistic reference, which is outside the scope of this work. However, Goldman himself notes (p. 14) that it is not essential to his theory that such pairs even be considered as distinct act-tokens; and if we do take them to be identical, they are ruled out by the preceding argument that the members of the generation pair are necessarily distinct.

As for Clause (2) of Goldman's definition, it also contains three requirements. The first of these, nonsubsequence, is implicitly encoded in the definition of GEN, by having the two acts have the same performance time. (Recall from the quotation, p. 54, that this is the meaning of nonsubsequence.) The second requirement precludes one member of the generational pair from being a temporal part of the other; thus an act of playing a C at the piano should not generate an act of playing a C scale at the piano, for the former is a temporal part of the latter. This restriction is also ruled out by the fact that the performance times of the two acts in the generational pair in Definition (G4) are identical. If one act is a temporal part of another, then necessarily the two have different performance times; in particular, the former must have a performance time that is shorter than, and included in, the performance time of the act of which it is part.

There is thus an essential difference between examples like the preceding, playing a C and playing a C scale, in which one act is truly a *temporal* part of the other, and examples like playing a C and playing a C chord, in which the two acts have identical times of performance, despite the fact that in some sense the former is part of the latter. An act of playing a C *can* generate an act of playing a C chord; the requisite generation-enabling condition is that the agent is also playing an E and a G.

Returning to the comparison of Goldman's definition with Definition (G4), the next thing to consider is the third part of his Clause (2), namely, the requirement that the acts forming the generational pair not be co-temporal. This requirement is directly entailed by Clause (iii) of Definition (G4). Informally, an act of α is cotemporal with an act of β if, were the acts to occur we would say of the performing agent G that he "did α (at t) while also doing β"; as, for instance, in "wiggling one's toes while also strumming the banjo", but not "moving one's knight to king-knight-seven while also checkmating one's opponent" (Goldman,

p. 22). Letting α be wiggling one's toes, and β be strumming the guitar, there does exist a C such that whenever C obtains, an occurrence of α will guarantee a simultaneous occurrence of β with the same performing agent, the C in question being "the performing agent strummed the guitar." This particular C is the only C that will satisfy Clause (i) of Definition (G4); and this C is β itself. But then it is ruled out by Definition (G4), as can be shown with an argument similar to that given to show that the distinctness of the act-tokens is ensured by Definition (G4): if C is itself β, then by itself it entails β and, therefore, violates Clause (iii) of Definition (G4).

The final clause of Goldman's definition needs to be considered subclause by subclause. Clause (3a) was what motivated the formulation of the CGEN relation, in terms of which GEN was defined; so Clause (3a) is directly encoded in Definition (G4). What about Clause (3b), which asserts the counterfactual that if G had not done α, he would not have done β? Goldman's support for Clause (3b) rests on a strong assumption that the agent also did not do any functional equivalent of α. That is, to use Goldman's example, when we say that "if G had not stuck his arm out the car window, he would not have signaled for a turn," we assume that G would not have, say, turned on his car blinker. This assumption, claims Goldman, is the natural consequence of the requirement imposed in counterfactual reasoning that "everything else (but the counterfactual) is held constant." In fact, it is not obvious that this is a natural consequence. If, in the actual case, G did signal by sticking his arm out the car window, we can assume that he intended to signal. Then in reasoning counterfactually about what would have been the case had he not stuck his arm out the window, we might want to hold constant his intention to signal, and indeed his signaling itself. It is not obvious whether in reasoning counterfactually we should assume that if G had not stuck his arm out the window, he would have had different intentions--in particular, he would not have had an intention to signal--or whether we should assume that if G had not stuck his arm out the window, he would have done different actions--in particular, he would have done something else to satisfy his intention to signal. In one case we ascribe to G different intentions in order to hold constant his bodily actions; in the other we ascribe to him different bodily actions in order to hold constant his intentions.

It is thus questionable whether Clause (3b) really should be included in the definition, for it is not obvious that it does describe a property of acts that intuitively form generational pairs. Given the difficulties in reasoning about and encoding counterfactuals, I have not included the requirements of Clause (3b) in Definition (G4).

What about Goldman's Clause (3c)? He gives a compelling reason in support of it: it is included to capture the asymmetry of the generation relation. His argument that it fulfills this function follows:

> [(3c)] succeeds in identifying the true direction of level-generation. For when one tries to apply this counterfactual in the wrong direction, one gets a false statement, or at any rate one which is not clearly true. Suppose, for example, that [for the case of signaling for a turn and extending one's arm out the window], we choose our C in such a way that when it is conjoined with a statement asserting that G signaled for a turn, it implies that G extended his arm out the car window. Our C would be: there is a rule saying that one signals for a turn if and only if one extends one's arm out the car window. Can we now say that if this C had not obtained, then even though G signaled for a turn, he would not have extended his arm out the car window? This counterfactual seems to be false. The question of whether or not G extended his arm out the car window (or *would* have extended his arm out the car window) is independent of the existence of such a rule. Even if this rule had not obtained, G's performance of the act of extending his arm out the car window would have been unaffected. Performing an act of signaling for a turn *does* depend on the existence of such a rule; but performing an act of extending one's arm out the car window does *not* depend on the existence of such a rule. Even if it were false that the *only* way to signal for a turn was to extend one's arm, and even if it were true that G signaled for a turn, it might still have been true that G extended his arm (pp. 42-43).

This argument bears careful consideration. Observe that the C that Goldman has in mind cannot really be the biconditional rule "one signals for a turn if and only if one extends one's arm out the window." If it were, his argument would be seriously flawed, since it relies on distinguishing between the possibility of G's extending his arm in the counterfactual world in which he does not signal, and the impossibility of G's signaling in the counterfactual world in which he does not extend his arm. Yet if these two acts were biconditionally related, then G would always do both, or do neither.

So C cannot be the rule which I would encode as:

(a) $\text{OCCURS(signal,G,t)} \equiv \text{OCCURS(extend-arm,G,t)}$.

There are two possible ways of weakening this rule; let us consider each in turn.

First, it might be that what is intuitively the generating act is sufficient but not necessary for the generated act. That is, it might be that extending one's arm really does imply signaling, but not vice versa: there may be more than one way to signal. In this case, C would be encoded as

(b) $[\text{OCCURS(extend-arm,G,t)} \rightarrow \text{OCCURS(signal,G,t)}] \wedge$
$[\text{OCCURS(signal,G,t)} \rightarrow$
$\text{OCCURS(extend-arm,G,t)} \vee \text{OCCURS(use-blinker,G,t)} \vee \ldots]$

A generation-enabling condition such as this poses no special problem, however. The asymmetry of generation in pairs of acts for which Rule (b) is the generation-enabling condition is guaranteed by Clause (3a) of Goldman's definition, and equivalently, by Clause (i) of Definition (G4) as follows. We attempt to see whether generation is satisfied in the "wrong" direction--i.e., we attempt to prove GEN(signal, extend-arm, G,t). In this attempt, we need to ask whether every occurrence of signaling for a turn when Rule (b) holds entails an occurrence of extending one's arm out the window. But we find that it does not: with Rule (b) holding, acts of signaling may occur when acts of, say, blinker-use, but not arm-extending, occur.

What then if we weaken (a) in the other direction, so C asserts that what is intuitively the generated act is sufficient, but not necessary for the generating act; i.e., for the current example signaling entails extending one's arm but not the converse:

(c) $[\text{OCCURS(signal,G,t)} \rightarrow \text{OCCURS(extend-arm,G,t)}] \wedge$
$[\text{OCCURS(extend-arm,G,t)} \rightarrow$
$\text{OCCURS(signal,G,t)} \vee \text{OCCURS(stretch,G,t)} \vee \ldots]$

This seems to be the case that Goldman has in mind: his analysis of the counterfactuals is consistent with it. A case like this still provides no problems as long as there are additional conditions necessary for an instance of extending one's arm to count as signaling. For instance, extending one's arm might count as signaling only if done when the agent is in a vehicle approaching a corner. Or, to take another example, consider acts of removing an olive-sized piece of dough and doing the last step in preparing challah for baking.[41] The only way to do the last step in preparing challah for baking is to remove an olive-sized piece of dough. However, for the occurrence of an act of removing an olive-sized piece of dough to entail the occurrence of an act of doing the last step in preparing challah for baking, additional conditions must also hold: e.g., the dough must be kosher, and the baker must have the intention of making challah.

When there are extra conditions like these, then asymmetry of generation is again guaranteed, but this time it is guaranteed by Clause (ii) of Definition (G4), rather than by Clause (i). To see this, consider again what happens if we attempt to see whether generation in the "wrong" direction is satisfied. Although Clause (i) is satisfied when Rule (c) holds--occurrences of signaling and finishing-challah-preparation do entail occurrences of arm-extending and dough-removing respectively--Clause (ii) is not satisfied. There can be no occurrences of signaling without arm-extending, nor of finishing-challah-preparation without dough-removing.

The only remaining case is the true biconditional, i.e., a rule such as (a). Such a rule cannot be the generation-enabling condition for any pair of actions, since no occurrences of such actions could satisfy satisfy Clause (3a) of Goldman's definition, or, equivalently, Clause (ii) of Definition (G4). This is because if occurrences of some α always entail occurrences of some β, and vice versa, then there will never occur one without the other. When a biconditional rule such as (a) holds, the actions it relates are, in fact, identical. This is not an unhappy result: Goldman's objections to equating acts such as G's signaling for a turn and G's extending one's arm out the window (at some time t) do not seem to apply to acts involving act-types that always co-occur.

The comparison of Goldman's definition and Definition (G4) is now complete: I have argued that Definition (G4) captures the requirements of his generation relation.[42] Further, the algebraic properties of generation noted earlier in the chapter--asymmetry, irreflexivity and transitivity--can be deduced from Definition (G4). The immediately preceding several paragraphs show that (G1)--the requirement of asymmetry--is a consequence of Definition (G4). (G2)

[41] I thank Barbara Grosz for this example.

[42] Except for his Clause (3b), whose status, in any case, is in doubt.

corresponds to Goldman's requirement that the occurrences in the generation pair be distinct, something I have also already shown to be a consequence of Definition (G4). The proof of (G3) is direct, by forming a new generation-enabling condition from the conjunction of the generation-enabling conditions of the antecedent generational pairs. That is, if GEN(α,β,G,t), there is some C1 such that CGEN(α,β,C1) and HOLDS(C1,t); and if GEN(β,γ,G,t), there is some C2 such that CGEN(β,γ,C2) and HOLDS(C2,t). Let D be equal to the conjunction of C1 and C2. Then it is staightforward to prove that GEN(α,γ,G,t) by setting the C in Definition (G4) to D.

4.4. The Executability Relation

Recall that when an agent has a simple plan that consists of doing some set of acts, he not only believes that each act he intends as part of the plan generates another act he intends as part of the plan (or generates the goal)--he also believes that he can execute each act. The former belief is captured in Clauses (2) and (3) of Definition (P0) on page 44 (restricted to simple plans); the latter in Clause (1). Having shown, in the last section, how to represent generation, we need now to turn our attention to representing executability.

In modeling executability, I will appeal to another concept from the philosophy of action, namely *basic actions* [Danto 63, Danto 65, Goldman 70]. Basic actions are, essentially, actions that can be performed at will; typical examples in the literature are an agent's raising his arm or turning his head. But there is a second important condition on basic actions: they are not performed by doing some other action.[43] This latter condition rules out actions like an agent's imitating a bird, which he may be able to execute at will, but only by doing something else like moving his arms in a particular way. Within the philosophical literature, the set of basic actions is roughly equal to the set of

[43]For Goldman, tokens of a basic act-type are not generated by other act-tokens; for Danto, they are not caused by another simultaneous act-token.

bodily actions.[44]

Intuitively, when we model some domain of action and consider executability in that domain, we do not reason "all the way down" to the level of bodily actions. Instead we decide, perhaps arbitrarily, on some set of actions that we assume are executable at will, and do not consider how actions in that set are actually performed. So, for instance, within the domain of computer mail, we might assume that all actions that consist in typing a command are executable at will. We can stipulate a set of basic act-types for the domain, such that every action we consider basic contains a domain-basic act-type (as well as a performing agent and a performance time). In so stipulating, we must be careful that none of the other act-types that we then reason about are ways of doing one of the basic act-types; hence, for the computer mail example, pushing down the 'd' key cannot be one of the domain act-types we consider.

Figure 4-1 illustrates the situation. The large circle represents all the types of actions there are in the universe. It is divided into act-types relevant to some domain and act-types irrelevant to it. The former is then further subdivided into three parts: (1) the stipulated set of basic act-types; (2) the act-types that conditionally generate basic act-types; and (3) the act-types that are conditionally generated by basic act-types.[45] Having so divided the universe of act-types, only those in the unshaded part of the diagram can be considered in reasoning about the domain.

Suppose we have stipulated some set of act-types as basic in a domain. Does this mean that all agents can perform at will, at any time, any member of the set? It does not. Agents have repertoires of basic act-types, and not all agents have the same repertoire. Within the philosophical literature, an example

---

[44]The story I have just told is oversimplified. Problematic cases arise, in which an action satisfies only one of the two conditions mentioned. Davidson [Davidson 80c] provides as an example the case of an agent who can tie his shoelaces by moving his fingers in a particular way, but is unable to move his fingers in the characteristic way at will when the laces are not there. Fortunately, it seems possible to avoid cases like this by carefully choosing the level of basic acts (see the following discussion).

[45]Since, as we saw earlier, it is possible for act-type α to conditionally generate act-type β and vice versa, provided the generation-enabling conditions in each case are different, there may be some overlap between categories.

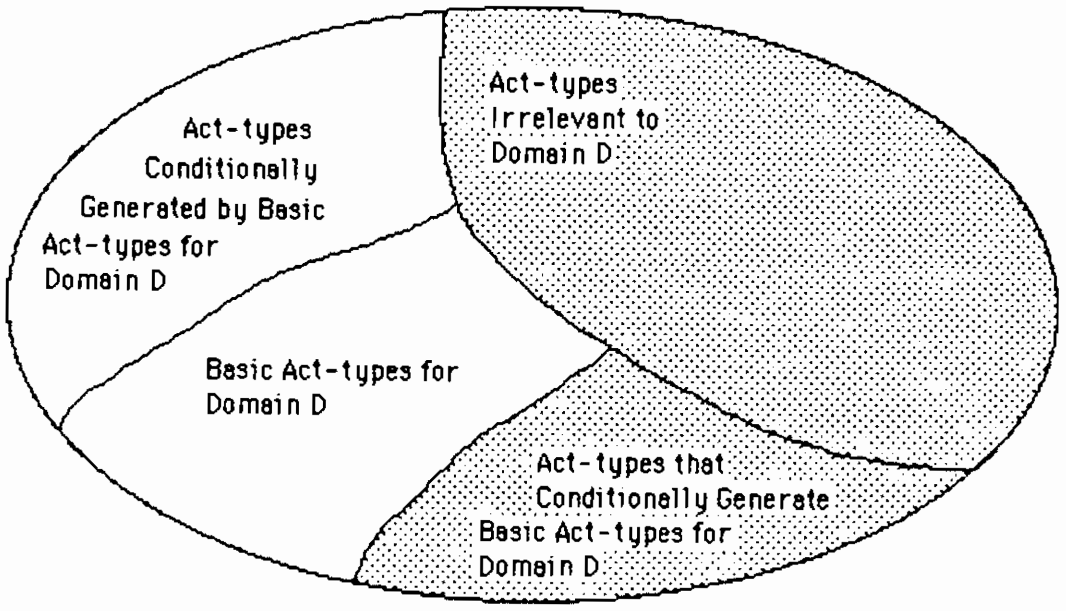

Act-types Conditionally Generated by Basic Act-types for Domain D
Act-types Irrelevant to Domain D
Basic Act-types for Domain D
Act-types that Conditionally Generate Basic Act-types for Domain D

Figure 4-1: Partitioning the Set of All Act-Types

commonly used to demonstrate this point is that of paralysis. Although the set of bodily act-types often forms a reasonable set of basic act-types--in fact, for most philosophers, it *is* the set of basic act-types--an agent who is paralyzed from the waist down will be unable to perform at will some members of the set: for example, moving one's leg. Further, a single agent may have different repertoires of basic act-types at different times: an able-bodied agent might become paralyzed, and thereby loose part of his repertoire of basic act-types. Thus the action consisting of G's doing α at time t is basic if and only if α is a domain-basic act-type that is in G's repertoire at time t.

One additional concept is needed to define executability. Even if the action of G's doing α at t is basic, G may not be able to do α at t, if he is not in the *standard conditions* with respect to α [Goldman 70, pp. 64-65]. An agent is in the standard conditions with respect to act-type α at time t if there are no external forces that prevent him from doing α at t. (In contrast, when α, though a member of the set of basic act-types for a domain, is not in an agent's repertoire at some particular time, it is because of internal conditions.) So, if an agent is not paralyzed, moving his arm will be basic for him, given that all bodily act-types are domain-basic; however, if his arm is tied to his body, he will not be in the standard conditions with respect to the act-type in question, and so will not be able to raise his arm.

Similar examples can be constructed when the set of basic act-types for a domain is stipulated. For instance, within the computer-mail domain, in which the stipulated set of domain-basic act-types comprises all those act-types which can be described as typing some command, an agent might be unable to type the command **DEL** . if the 'd' key on his keyboard is broken. In this case, he is not in the standard conditions with respect to the act-type.

In sum, within a domain we can stipulate a set of basic act-types. At certain times, certain agents will have within their own repertoires some subset of these basic act-types; when G has in his repertoire at t the domain-basic act-type α, then $<\alpha,G,t>$ will be a basic action. If $<\alpha,G,t>$ is a basic action, and G is in the standard conditions with respect to α at time t, then $<\alpha,G,t>$ will also be executable. To model this, I will introduce three new predicates into the representation language: BASIC, SC, and EXEC. BASIC(α,G,t) will be true whenever α is a domain-basic act-type which is in G's repertoire at time t; i.e., whenever $<\alpha,G,t>$ is a basic action. SC(α,G,t) will be true whenever agent G is in the standard conditions for α at time t; and EXEC(α,G,t) will be true whenever G can execute α at time t.

We can immediately see the following:

Axiom
(E1a) $\mathrm{BASIC}(\alpha,G,t) \wedge \mathrm{SC}(\alpha,G,t) \rightarrow \mathrm{EXEC}(\alpha,G,t)$

That is, if α is basic for G at time t, and G is in the standard conditions with respect to α at time t, then G can execute α at time t (equivalently, $<\alpha,G,t>$ is *executable*).

Axiom (E1a) provides a definition of executability basic actions. But what about nonbasic actions? A nonbasic action--i.e., an action consisting of G's doing α at t where α is not a domain-basic act-type--is executable only if it can be generated by--some series of basic actions that are themselves executable. It is important to allow *series* of basic actions; otherwise we could not represent the executability of, say, playing a C scale, which is generated by the series of acts playing a C, playing a D, and so on. If we let a semicolon (;) denote temporal concatenation of acts,[46] we can represent the executability of nonbasic acts as follows:

Axiom

$$\text{(E1b)}\ \exists \alpha_1 \ldots \exists \alpha_n\ [(\mathrm{BASIC}(\alpha_1,G,t) \wedge \ldots \mathrm{BASIC}(\alpha_n,G,t) \wedge \mathrm{SC}(\alpha_1,G,t) \wedge \ldots \wedge \mathrm{SC}(\alpha_n,G,t) \wedge \mathrm{GEN}(\alpha_1;\ldots;\alpha_n,\beta,G,t))] \rightarrow \mathrm{EXEC}(\beta,G,t)$$

In fact, the antecedent of Axiom (E1b) can be weakened. Each act-type α_i need only be basic for G during the subinterval of t during which it is being performed; and similarly, G need only be in the standard conditions with respect to each α_i for its subinterval of performance. To see why this is so, consider an example from the domain of computer mail. This time, however, let the set of domain-basic act-types be all act-types that consist in typing some character, rather than those that consist in typing some command. Then typing type **DEL .** is a nonbasic act-type. G's typing **DEL .** at time t may be generated by his typing **D**, typing **E**, typing **L**, typing <space>, and then typing **.**, each during a subinterval of t. Assume that all these act-types are in G's repertoire throughout the time interval t. But now also assume that the 'd' key on G's keyboard breaks during this interval. As long as it is not broken during the time in which he attempts to type the **D**--i.e., as long as he is in the standard conditions with respect to typing **D** during the subinterval of t in which he performs this basic action--his nonbasic act of typing **DEL .** is executable.

Despite such cases, I will, for the sake of simplicity, henceforth ignore the possibility that one of the basic actions in a generating series may fail to be basic during some subinterval; and similarly for the requirement on standard conditions. The described weakening of the antecedent of Axiom (E1b) could be readily incorporated into an extensions of this theory, however.

---

[46]That is, using Allen's temporal operators:

$$\mathrm{OCCURS}(\alpha;\beta,G,t) \equiv \exists t_1 \exists t_2\, [\mathrm{OCCURS}(\alpha,G,t_1) \wedge \mathrm{OCCURS}(\beta,G,t_2) \wedge \mathrm{MEETS}(t_1,t_2) \wedge \mathrm{STARTS}(t_1,t) \wedge \mathrm{FINISHES}(t_2,t)]$$

We can now combine Axioms (E1a) and (E1b) into a definition of executability:

<u>Definition</u>
(E1) $\mathrm{EXEC}(\beta,G,t) \equiv$
(i) $[\mathrm{BASIC}(\beta,G,t) \wedge \mathrm{SC}(\beta,G,t)] \vee$
(ii) $\exists \alpha_1 \ldots \exists \alpha_n [\mathrm{BASIC}(\alpha_1,G,t) \wedge \ldots \wedge (\mathrm{BASIC}(\alpha_n,G,t) \wedge$
$\mathrm{SC}(\alpha_1,G,t) \wedge \ldots \wedge \mathrm{SC}(\alpha_n,G,t) \wedge$
$\mathrm{GEN}(\alpha_1;\ldots;\alpha_n,\beta,G,t)]$

It may be extremely difficult to prove that some act of G's doing β at t is executable when β is not basic. This is because, to do so, one not only has to find a series of act-types that conditionally generate β under some enabling condition C, but one also has to determine that the the condition C will hold during t. This means that one may have to reason about whether the agent can bring about the enabling condition. And this in turn requires a careful theory that distinguishes between events that are in an agent's control, and events that are not. Richard Pelavin [Pelavin 86] is working on a general solution to this problem.

Before leaving the topic of executability, there is an interesting relation between executability and generation that should be observed. If an act of α is executable, and it generates an another act, of β, then the act of β is also executable. That is,

(E2) $\mathrm{EXEC}(\alpha,G,t) \wedge \mathrm{GEN}(\alpha,\beta,G,t) \rightarrow \mathrm{EXEC}(\beta,G,t)$

The proof of (E2) is quite direct. If G's doing α at t is basic, then since it is executable, G is also in the standard conditions with respect to α at time t; hence, since G's act of α generates his act of β at time t, the latter is executable. If instead G's doing α at t is not basic, it must be generated by some action consisting of G's doing the sequence of $\gamma_1;\ldots;\gamma_n$ at t, where each of the members of this sequence is a basic action, and where G is in the standard conditions with respect to each γ. But then, since by (G3) generation is transitive, an act of $\gamma_1;\ldots;\gamma_n$ also generates an act of β; hence, the latter is executable.

4.5. Representing Beliefs and Intentions

Let us very briefly take stock of what the representation language looks like so far. It is a typed, first-order predicate calculus, including terms of type time interval, property, act-type, agent and object. Seven predicate symbols have been introduced--OCCURS, HOLDS, CGEN, GEN, EXEC, BASIC, AND SC--along several with axioms and theorems defining interdependencies between them. Yet to be represented are beliefs and intentions. To model these, I will need to complicate the language, introducing two modal operators: BEL and INT.

Beliefs

To represent belief I will make use of the well-known framework for doing so that derives from the work of Jaakko Hintikka [Hintikka 62]. The advantages of Hintikka's formalism consist largely in the fact that it is widely used in AI research and is quite well understood.[47] Hintikka writes $B_G P$ to denote the fact that agent G believes the proposition P; I will modify this slightly and write BEL(G,p,t), to denote the fact that G believes property p during time interval t. Then, following Hintikka, I will adopt the following axiom:

Axiom
(B1) $BEL(G,p,t) \land BEL(G,p \rightarrow q,t) \rightarrow BEL(G,q,t)$

That is, if, during time t, G believes some proposition p, and also believes that p implies q, then he believes q. In adopting this axiom, I am confronted with the well-known problem of consequential closure, but for for the time being I will accept that.[48]

In introducing the modal BEL operator, a new rule of inference must be adopted. Whereas when we were using a first-order predicate calculus, the single rule of modus ponens was sufficient

Inference Rule
(R1) From $\vdash p$ and $\vdash (p \rightarrow q)$ infer $\vdash q$

we now need a second rule:

[47] Halpern and Moses [Halpern 85] provide a good overview of Hintikka's model as well as alternative models of belief and knowledge.

[48] See Konolige [Konolige 84] for an analysis of belief that avoids consequential closure.

Inference Rule
(R2) From $\vdash p$ infer $\vdash$ BEL(G,p,t) for all G and all t.

The literature includes much discussion about what axioms about belief should be adopted in addition to Axiom (B1).[49] In particular, there is much debate about the so-called introspection axioms. One of these encodes the assertion that agents believe that they believe what they believe:

Axiom
(B2) $\mathrm{BEL}(G,p,t) \rightarrow \mathrm{BEL}(G, \mathrm{BEL}(G,p,t), t)$

Another encodes the assertion that agents believe that they do not believe what they do not believe:

Axiom
(B3) $\neg \mathrm{BEL}(G,p,t) \rightarrow \mathrm{BEL}(G, \neg \mathrm{BEL}(G,p,t), t)$

A decision for or against such axioms will not immediately affect the theory developed in this thesis. Consequently, I will somewhat arbitrarily accept these axioms, along with (B4), which guarantees that agents do not have inconsistent beliefs:

Axiom
(B4) $\neg \mathrm{BEL}(G,\mathit{false},t)$

What is essential is that the so-called knowledge axiom not be introduced, i.e., it is *not* true that

$$\mathrm{BEL}(G,p,t) \rightarrow \mathrm{HOLDS}(p,t)$$

since agents may have invalid beliefs.

Intentions

To model intention, I will introduce a four-place modal operator INT, where $\mathrm{INT}(G,\alpha,t_2,t_1)$ denotes agent G's intention at time t_1 to perform action α

[49]For discussion see Lenzen [Lenzen 78]and Israel [Israel 85].

at time t_2. So agents are modeled here as intending to do actions.[50]

There is an important difference in the types of the arguments to BEL and INT: agents believe propositions but they intend to do acts. When we need to capture an intuition that an agent intends some proposition p, we can say, along the lines discussed in Chapter 3 that he intends to achieve(p). This means making use of the *achieve* act-type constructor function described in Section 4.2. Agents may also intend to do what are intuitively generational pairs: G may intend to do β by doing α at time t_2. We cannot write GEN(α,β,G,t) as the second argument to an INT predicate, because doing so would violate a type constraint: this argument must be an act-type term. Hence, I will introduce a special act-type constructor function, *by*, which maps from cross products of act-types to act-types. To denote the fact that G intends at time t_1 to do β by doing α at time t_2, (or, to do α in order to do β at time t_2), I will write INT(G,by(α,β),t_2,t_1).

As with belief, there are a number of different views that one might take about the nature of intention, and consequently, a number of different sets of axioms that might be considered to characterize intention. Alternative axiomatizations of intention have not been as widely discussed in the literature as have been axiomatizations of belief; fortunately, as with the belief case, the theory of plan inference developed in this thesis can be developed without firm commitment to one or another axiomatization. Of course, future extensions to and generalizations of the model will rely more and more on the details of any particular axiomatization of intention adopted.

The axioms that I will discuss are reminiscent of Axiom (B1). Axiom (B1) represents reasoning from beliefs about propositions, to beliefs about further

[50]The performance time of an intended act may be underspecified. We might, for instance, want to represent the fact that Walt intends now to go to the supermarket sometime next week. To express this, we might attempt to write something like the following:

$$\text{INT(Walt,go-to(supermarket),t,now)} \wedge \text{DURING(t,next-week)}$$

However, what this formula really represents is that Walt intends to go to the supermarket at some particular time t, which happens to be within the next week. The representation of intentions to do actions that are "vaguely" specified in one or more ways is an interesting problem for future research.

propositions that are themselves believed to be entailed by the former. I will here describe axioms that represent reasoning from intentions to do actions, to intentions to do further actions that are themselves believed to be generated by the former. Analogues of the introspection axioms, (B2) and (B3), do not seem to arise in intention: intending to do α does not imply intending to intend to do α.

One axiom that we certainly do *not* want is the following:

$$INT(G,\alpha,t_2,t_1) \wedge GEN(\alpha,\beta,G,t_2) \rightarrow INT(G,\beta,t_2,t_1).$$

Agents do not intend all the acts that are generated by their intended acts; in particular, they do not intend acts that they do not even realize are so generated. Nor do we want the axiom:

$$INT(G,\beta,t_2,t_1) \wedge GEN(\alpha,\beta,G,t_2) \rightarrow INT(G,\alpha,t_2,t_1)$$

Not only might G not realize that α generates β, but even if he does, he might intend to generate it in some other way.

The more interesting axioms involve those acts that the agent does believe are generated by his intended acts. There are at least two such axioms worth considering:

Axiom

(I1) $INT(G,\alpha,t_2,t_1) \wedge BEL(G,GEN(\alpha,\beta,G,t_2),t_1) \rightarrow$
$INT(G,\beta,t_2,t_1)$

(I2) $INT(G,\alpha,t_2,t_1) \wedge BEL(G,GEN(\alpha,\beta,G,t_2),t_1) \rightarrow$
$INT(G,by(\alpha,\beta),t_2,t_1)$

Axioms (I1) and (I2) both express claims about what further intentions an agent must have when he has an intention to do α, and believes that this act will generate his doing β. Axiom (I1) states he must also intend to do β, and Axiom (I2) claims he must also intend to do by(α,β). It is possible to accept either of Axioms (I1) or (I2) independently, to accept neither of them, or to accept them both.

Bratman [Bratman 86, especially Chap. 10] presents a detailed theory that rejects both Axioms (I1) and (I2). Part of his argument against these axioms rests on examples about what can happen when G discovers that the belief included in the antecedents of Axioms (I1) and (I2) is wrong. Suppose, for instance, that I intend to do my laundry and expect that by so doing I will pollute the local water system (say, because my detergent has a high phosphate content). After doing my laundry, however, I discover that I was mistaken--my detergent is actually phosphate-free. It is not necessarily the case that at this point, I go back and attempt to pollute the local water system. Thus, since having an intention to do α, on Bratman's analysis, leads to endeavoring to carry out that intention, he concludes that it was not necessarily the case that I ever intended to pollute the water system; *a fortiori*, it was not necessarily the case that I ever intended to pollute the water system by doing my laundry (or to do my laundry in order to pollute the water system).

Roderick Chisholm [Chisholm 76] seems to accept something close to Axiom (I2) without accepting Axiom (I1). His "principle of the diffusiveness of intention" (p. 75) is quite close to Axiom (I2)[51]; his "principle of the nondivisiveness of intention" (p. 74) is quite close to a rejection of Axiom (I1). Chisholm's views are discussed in more detail in [Bratman 86], Chapter 10.

One might also accept axiom Axiom (I1) without Axiom (I2). On this account, if an agent has an intention to do α, and a belief that his act of α will generate an act of β, he must intend to do β. However, he may not intend to do α in order to do β, i.e. by(α,β). Such a view is at least plausible. The type of example that would support it is the following. Suppose that G intends to cheat on his income tax, believing that in so doing he will be committing a crime. Since we hold G responsible for his criminal act, it may not be unreasonable to say that he intends to commit the crime. However, there is less pressure to say that he intends to cheat on his income tax in order to commit a crime.

We might also consider an axiom like the following:

Axiom

(I3) $\mathrm{INT}(G, by(\alpha,\beta), t_2, t_1) \rightarrow$
$\mathrm{INT}(G,\alpha,t_2,t_1) \wedge$
$\mathrm{INT}(G,\beta,t_2,t_1) \wedge$
$\mathrm{BEL}(G,\mathrm{GEN}(\alpha,\beta,G,t_2),t_1)$

[51]One difference, whose consequences should be explored, is that Chisholm's diffusiveness principle concludes that G intends α-*and*-β, rather than β-by-α.

That is, if G intends at time t_1 to do β by doing α at time t_2, he must intend both to do α at time t_2 and to do β at time t_2, and must believe that these two acts will form a generational pair. Note that if we accept Axiom (I3), we cannot also accept Axiom (I2) without accepting Axiom (I1).

As mentioned above, I will not attempt here to argue seriously for any particular set of axioms characterizing intention. Instead I will accept without further discussion Axioms (I1) and (I3), but not Axiom (I2). Refinements to the theory of plan inference developed here, however, may depend on a more careful assessment of the axiomatization of intention.

4.6. Combining the Pieces

We can now combine the various pieces of the representation that have been built up in this chapter, in order to encode the definition of having a simple plan[52]:

Definition

(P1) $\text{SIMPLE-PLAN}(G,\alpha_n,[\alpha_1, \ldots,\alpha_{n-1}],t_2,t_1) \equiv$

(i) $\text{BEL}(G,\text{EXEC}(\alpha_i,G,t_2),t_1)$ for $i=1,\ldots,n-1 \wedge$

(ii) $\text{BEL}(G,\text{GEN}(\alpha_i,\alpha_{i+1},G,t_2),t_1)$ for $i=1,\ldots,n-1 \wedge$

(iii) $\text{INT}(G,\alpha_i,t_2,t_1)$ for $i=1,\ldots,n-1 \wedge$

(iv) $\text{INT}(G,by(\alpha_i,\alpha_{i+1}),t_2,t_1)$ for $i=1,\ldots,n-1$

The formula $\text{SIMPLE-PLAN}(G,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],t_2,t_1)$ should be read "agent G has a simple plan at time t_1 to do α_n at time t_2, that consists of doing the set of acts $\{\alpha_1, \ldots, \alpha_{n-1}\}$ at time t_2. Note that these are simultaneous acts. Clause (i) of Definition (P1) captures Clause (1) of Definition (P0). Clause (ii) of Definition (P1) captures both clauses (2) and (3) of Definition (P0): when i takes the value n-1, Clause (ii) of Definition (P1) captures the requirement, stated in Clause (2) of Definition (P0), that G believes his acts will entail his goal; when i takes values between 1 and n-2, it captures the requirement of Clause (3) of Definition (P0), that G believes each of his acts plays a role in his plan. Similarly, Clause

[52]The definition makes use of a list of action instances. I have adopted the Prolog notation of writing a list as $[X_1,X_2,X_3]$, where that abbreviates $cons(X_1, cons(X_2, cons(X_3, [])))$, cons being a binary function symbol.

(iii) of Definition (P1) captures Clause (4) of Definition (P0), and Clause (iv) of Definition (P1) captures Clauses (5) and (6) of Definition (P0).

```
become a member of the Mayday Late-Night Club
(become-member)
                                   ^
                                   |
                                   |
satisfy entry requirement for Mayday Late-Night Club
(satisfy-rqts)
                                   ^
                                   |
                                   |
go to the movies at 1:00 a.m. on May the first
(goto-movies-late)
```

Figure 4-2: G's Plan to Join the Mayday Late-Night Club

An example will illustrate what is entailed in having a simple plan. Consider again the example, discussed in Chapter 3, of an agent who wants to join the Mayday Late-Night Club. His plan is diagrammed in Figure 4-2; shown are the types of the acts he intends, along with abbreviations that we can use to refer to them. Each act is drawn below the act it is meant to generate, with an arrow pointing to that act. Assume that G has this plan at time t_1; the performance time of his intended acts is t_2 (which is 1:00 a.m. on May first). Then by Definition (P1) G must have all of the following beliefs and intentions:

1a. BEL(G, EXEC(goto-movies-late,G,t_2), t_1)

1b. BEL(G, EXEC(satisfy-rqts,G,t_2), t_1)

2a. BEL(G, GEN(goto-movies-late,satisfy-rqts,G,t_2), t_1)

2b. BEL(G, GEN(satisfy-rqts,become-member,G,t_2), t_1)

3a. INT(G, goto-movies-late, t_2, t_1)

3b. INT(G, satisfy-rqts, t_2, t_1)

4a. INT(G, by(goto-movies,satisfy-rqts),t_2,t_1)

4b. INT(G, by(satisfy-rqts,become-member),t_2,t_1)

The number of each belief and intention corresponds to the clause in (P1) that it satisfies. There is an additional belief and an additional intention that can be inferred.

First, although Definition (P1) only directly stipulates that G believes that each of the constituent actions in his plan is executable, it entails that he also believes that his goal act is executable. To see this, first observe that if the act of G's satisfying the entry requirements (satisfy-rqts) at t_2 is executable and generates G's becoming a member of the club (become-member) at t_2, then the latter act is also executable: this is a direct instantiation of Theorem (E2):

(i) $\text{EXEC}(\text{satisfy-rqts},G,t_2) \wedge \text{GEN}(\text{satisfy-rqts},\text{become-member},G,t_2) \rightarrow \text{EXEC}(\text{become-member},G,t_2)$

But then by Inference Rule (R2) we can deduce that G believes this fact, i.e.:

(ii) $\text{BEL}(G, [\text{EXEC}(\text{satisfy-rqts},G,t_2) \wedge \text{GEN}(\text{satisfy-rqts},\text{become-member},G,t_2) \rightarrow \text{EXEC}(\text{become-member},G,t_2)],t_1)$

Items (1b) and (2b) above assert that G believes that his satisfying the requirements at t_2 is executable, and believes that this act generates his becoming a member at t_2. But these are precisely the antecedents of G's belief in (ii). Hence, by Axiom (B1), which asserts that agents believe what they believe to be the consequences of their beliefs, we can deduce that G believes that his becoming a member at t_2 is an executable action:

(iii) $\text{BEL}(G, \text{EXEC}(\text{become-member},G,t_2), t_1)$

We can also deduce that G intends his goal action, even though this is not explicitly required by Definition (P1). To do this, we make use of Axiom (I3), which asserts that whenever an agent intends to do some α in order to do some β, he must, amongst other things, intend to do β. Then since (4b) asserts that G intends to satisfy the entry requirements in order to become a member (that is, he intends an act of the constructed type by(satisfy-rqts, become-member)), G must also intend to become a member:

(iv) $\text{INT}(G, \text{become-member},t_2, t_1)$

An alternative proof of (iv) can be given by instantiating Axiom (I1) with the sentences in (2b) and (3b): whichever set of axioms we choose to represent intention, we will want this deduction to be sound.

Note that even when we accept Axiom (I1), and thereby claim that agents intend all the expected effects of their intended acts, side-effects are still not part of the agent's plan. In the example under discussion, G might believe that by going to the movies late at night he will annoy his parents, i.e.,

(v) BEL(G, GEN(goto-movies-late,annoy-parents,G,t_2), t_1)

By (3a) we know that G intends to go to the movies late. Then since G so intends, and believes (as represented in (v)) that an effect of this is annoying his parents, by Axiom (I1) it follows that he will also intend to annoy his parents:

(vi) INT(G, annoy-parents, t_2, t_1)

However, annoying his parents is still not part of G's plan, because he does not believe that it generates his goal act, either directly or transitively: he is missing a belief about the act annoy-parents that would satisfy Clause (ii) of Definition (P1).[53]

Four more examples of simple plans are diagrammed in Figures 4-3 through 4-6. As in Figure 4-2, I have shown in these diagrams only the types of the acts G intends: intended performance time can be assumed to be t_2, and the time that the intention is held, t_1. The reader can determine directly from Definition (P1) the set of beliefs and intentions that G must have if he has any of these plans.

[53]The treatment I give to side-effects in plans is quite similar to that Goldman gives in his account of intentional action; see Goldman, pp. 49-63.

```
get one's driver's license
                ^
                |
                |
convince one's examiner to give one a license
                ^
                |
                |
convince one's examiner that one is a competent driver
                ^
                |
                |
signal for a turn
                ^
                |
                |
extend one's arm out the car window
```

Figure 4-3: G's Plan to Get His Driver's License (adapted from Goldman, pp. 69-60)

```
cause one's cousin to leave
                ^
                |
                |
annoy one's cousin
                ^
                |
                |
watch "Cheers"
```

Figure 4-4: G's Plan to Get His Cousin to Leave

```
speed up entry into mail
                 ^
                 |
                 |
cause fewer messages to be loaded
                 ^
                 |
                 |
cause only new messages to be loaded
```

Figure 4-5: G's Plan to Speed up Entry into the Mail System

```
get one's officemate to tell one the hospital's phone number
                             ^
                             |
                             |
ask one's officemate to what the hospital's phone number is
                             ^
                             |
                             |
utter the words "Do you know the hospital's phone number?"
```

Figure 4-6: G's Plan to Find out the Phone Number of the Hospital

4.7. Reconsidering the Standard Model

We now have sufficient representational tools to reconsider the standard, hierarchical model of plans most commonly adopted in AI systems and to compare it with the model developed in this chapter and the preceding one. The most obvious difference between the two models, which I discussed at length in Chapter 3, is that the former views plans primarily as data structures, while the latter views them as mental phenomena. But a question still arises as to the correspondence between the relations between acts used in the data-structure view--*causes, is-a-precondition-of*, and *is-a-way-to*--and those used in the mental-phenomenon view--*generation* and *enablement*.

The relations used in the data-structure view involve some redundancy, as illustrated by the following two operators:

```
Header: flip_switch              Header: turn_on_light

 Effect:  light_on                 Body: flip_switch
```

These operators encode the same information, provided we equate the act of turning on the light with the act of achieving that the light is on. In general, whenever some act α *causes* some proposition p, it is also true that α *is-a-way-to* achieve(p). It will thus be sufficient to restrict our attention to the two relations *is-a-precondition-of* and *is-a-way-to*.

Consider then the following generic operator, in which α *is-a-way-to* β, and P *is-a-precondition-of* β:

```
Header: β
Preconditions: P
Body: α
```

Is there a way to map the relations expressed in it into the two relations *generation* and *enablement*? In fact there are a number of plausible mappings. Figure 4-7 shows some of these.

Sentence (1) in the figure encodes the interpretation, apparently implicit in certain operators, that the preconditions P are sufficient for the performance of the header β by performance of the body α, but without necessarily being sufficient for the performance of α itself (and consequently, without necessarily being sufficient for the performance of β itself). This is the interpretation that most straightforwardly translates into the relations between act-types used in this thesis: under this interpretation, the entire generic operator is associated with the sentence CGEN(α,β,C). However, it is not always the interpretation implicit in existing planning work. Sentences (2a) and (2b) encode the interpretation of the preconditions P as necessary for the occurrence of β: these two sentences differ from one another in the time interval during which P is meant to hold. Consider a typical planning operator in which a proposition representing that the power is on is included as a precondition for an operator with a header representing the act of turning on the light. It is not always obvious whether the former is intended to be a necessary, or merely sufficient, condition for performance of the latter. (It might be merely sufficient, if, say, the light is attached to an emergency generator.) The preconditions in an action operator

```
Given   Header: β
        Precondition: P
        Body: α

Does this mean:

1) CGEN(α,β,P), i.e.,
        OCCURS(α,G,t) ∧  HOLDS(P,t)  →   OCCURS(β,G,t) ∧ ...

2a)OCCURS(β,G,t)  →   HOLDS(P,t)
2b)OCCURS(β,G,t)  →   ∃ t0[MEETS(t0,t) ∧  HOLDS(P, t0)]

3a)OCCURS(α,G,t)  →   HOLDS(P,t)
3b)OCCURS(α,G,t)  →   ∃ t0[MEETS(t0,t) ∧  HOLDS(P,t0)]

4a)HOLDS(P,t)  →   ∀ G[EXEC(α,G,t)]
4b)HOLDS(P,t)  →   ∀ t1[MEETS(t,t1)  →   EXEC(α,G,t1)]
        where some further restriction on the length of t1
        is also given

5a)HOLDS(P,t)  →   ∀ G[EXEC(β,G,t)]
5b)HOLDS(P,t)  →   ∀ t1[MEETS(t,t1)  →   EXEC(β,G,t1)]
        where, again, some further restriction on the length
        of t1 is also given

or some combination of these, e.g., (4a) + (1)?
```

Figure 4-7: Interpretations of the Standard Relations

are sometimes also meant to be related to the performance of the act-type in the body of the operator--either to be necessary for it, as encoded in (3a) and (3b), or to be sufficient for it, as encoded in (4a) and (4b). One illustration of the preconditions being related to the body act-type α instead of the header act-type β would be an operator with a header representing turning on the light, body representing flipping the switch, and precondition representing both the power being on and the agent standing near the switch. The agent may well be able to turn on the light without standing near the switch, say by throwing something at it. Standing near the switch is meant, in an example like this, to be necessary for the body action to occur. Combinations of the sentences shown in the figure are also possible: the interpretation underlying Sacerdoti's work seems to be a combination of (1) and (4a).

Unfortunately, much of the existing planning literature has been vague about the intended interpretation of action operators, and has used them at different times to mean different things. It is because of the resulting ambiguity of interpretation that I have avoided using the relations *causes*, *is-a-precondition-of*, and *effects* in this work, and have instead made use of *generates* and *enables*. No doubt there are translations of the former into the latter, but it appears that no one translation will account for all existing uses of them.

CHAPTER V
Invalidities in Simple Plans

Given the analysis of having a plan as having a particular configuration of beliefs and intentions, this chapter associates having an invalid plan with having such a set of beliefs and intentions where some of the beliefs are incorrect. It argues that in modeling plan inference, what are important are judgements of invalidities, which are analyzed as discrepancies between the beliefs that the inferring agent ascribes to the actor, when she believes he has some plan, and beliefs she herself holds. The types of discrepancies that can arise under the given analysis are defined, and are shown to correspond to regularities observable in cooperative responses to questions. Specifically, plans may be judged to contain unexecutable acts, or to be ill-formed. Additionally, queries may be judged incoherent: this observation leads to a reanalysis of what must be inferred in cooperative conversation. The chapter concludes by comparing this analysis with some earlier work in AI on cooperative question-answering.

5.1. Invalidity Judgements

In the previous two chapters, I developed an account of plans as mental phenomena, analyzing the state of having a plan as that of having a particular configuration of beliefs and intentions. The analysis was presented informally in Definition (P0) (on page 44), and somewhat more formally, for the case of simple plans, in Definition (P1) (on page 78). Given these definitions, we can state what it means for an agent to have an invalid simple plan: namely, G has an invalid simple plan if and only if he has the set of beliefs and intentions listed in Definition (P1), where one or more of those beliefs is incorrect, and, consequently, one or more of the intentions is unrealizable.

The correctness of the actor's beliefs thus determines the validity of his plan: if all the beliefs that are part of his plan are correct, then all the intentions in it are realizable, and the plan is valid. Validity in this absolute sense, however, is not what is of primary concern in modeling plan inference. What is important there is rather the inferring agent's *judgement* of whether the actor's plan is valid. This judgement depends on whether the inferring agent believes to be correct the beliefs that she ascribes to the actor. So, an inferring agent's judgement of validity or invalidity differs from absolute validity or invalidity in two ways: first, the beliefs being judged are those that the inferring agent ascribes to the actor by virtue of believing he has some plan--they may not be beliefs the actor indeed has; and second, the judgement of these beliefs is from the perspective of the inferring agent--she will judge to be incorrect beliefs that are incompatible with her own beliefs.

In fact, if we make the Correct-Knowledge Assumption (CKA) and the Closed-World Assumption (CWA) discussed in Section 2.3, the inferring agent's beliefs will be sound and complete, and she will judge a plan to be invalid, and invalid in a particular way, only if it is in fact invalid in that way. In general though these assumptions are too strong: people may have erroneous beliefs and, as a result, may err in their judgements of other people's beliefs and plans. Hence an inferring agent may consider invalid a plan that in fact would succeed, and vice versa, and two agents may differ in their beliefs about whether any particular plan is invalid.

To summarize: When an actor has a plan, that plan will be valid according to whether its constituent beliefs are correct. When an agent infers an actor's plan, she will judge it to be valid according to whether she believes correct the beliefs she ascribes to the actor as part of his plan. When there are discrepancies between those beliefs and beliefs she herself holds, she will judge the plan to be invalid. Henceforth, when I speak of a plan as valid or invalid, I will mean one that is valid or invalid from the perspective of some other agent. Similarly, when I speak of a plan being invalid in some particular way, e.g., being ill-formed[54], I will mean that there is a discrepancy that can be categorized as an ill-formedness between the inferring agent's beliefs and those she ascribes to the actor.

In this chapter, I will examine the types of discrepancies that can arise between the beliefs of the inferring agent and the beliefs she ascribes to the actor. Specifically, discrepancies may arise between one or more of the beliefs,

[54]See Section 5.3.

corresponding to Clauses (i) and (ii) of Definition (P1), that the inferring agent ascribes to the actor, when she believes he has some simple plan, and beliefs that she herself holds. I will demonstrate that these belief discrepancies explain certain regularities that can be observed in cooperative responses to questions. I will then discuss a regularity that cannot be explained by the model so far developed; this will lead to a reanalysis of what must be inferred in cooperative conversation.

Before beginning this discussion, though, I need to make one point about the relationship between the types of invalidities I will define, and the types of responses found in naturally occurring discourse. I take the intuitively perceived regularities in, and differences between, responses found in naturally occurring discourse to be evidence for structural regularities in, and differences between, the plans that are inferred to underlie queries. In other words, I am willing to make assertions of the following form: if a response to some query exhibits a certain feature (say *F*), one can account for that response by claiming that the plan inferred to underlie the query is invalid in some particular way (say, it has an invalidity of type *I*). However, I will not make converse assertions: I do not claim that whenever the plan inferred to underlie a query is invalid in some particular way (*I*), a cooperative response to it must exhibit a particular feature (*F*). There are simply too many other factors that can affect response generation, most importantly factors of relevance and salience.

So, for instance, I will account for Example (1), first discussed in Chapter 1, by saying that the plan inferred to underlie the query there is ill-formed. R responds in that example by informing Q that some act he intends will not facilitate his goal; such responses, I will claim, result from inferred ill-formed plans. However, I will not claim that every time an ill-formed plan is inferred to underlie some query, the inferring agent will necessarily inform the actor that an act he intends will not facilitate his goal. In the example under discussion, R might decide that it is irrelevant to Q to know where Kathy is--say, if R believes that Q will never again call her--and so might simply tell him her home phone number without further comment. Examples like this are common in technical domains: computer experts providing advice often refrain from including in that advice all the details of their judgements of the advice-seeker's plan, in order not to confuse him with excessive information.

It is of course possible to engineer a computer system that consistently generates responses that exhibit some certain feature whenever it infers a plan that is invalid in a particular way. SPIRIT does just this. Indeed, such a system is likely to perform substantially better than one that does not take any account of the inferred plan in generating a response. But building such a system is not equivalent to claiming that it properly and completely models human conversational ability (or even human conversational ability within the confines of answering queries about how to do some action), and it is this latter claim that I am avoiding. I maintain that the plan inferred to underlie a query, along with the type of any invalidities noted in that plan, are but two of several factors that contribute to the determination of a cooperative response.

In fact, I would speculate that an account of appropriate responses should be a consequence of a full-blown account of rational, cooperative behavior. In principle, one should be able to show from first principles of cooperativity (which themselves may be derived from first principles of rationality) that, when an agent is asked a query, and she infers an underlying plan that is invalid in some particular way, then given certain facts about the context in which the query was asked, the most cooperative response is precisely that which can be observed in natural discourse. In certain "typical" contexts this would be the response exhibiting the particular features that motivated the initial analysis. Successfully demonstrating this result would lend more support to the analysis of plan invalidities than does stipulating that particular invalidities correlate with particular responses. The approach being suggested is essentially that taken by Philip Cohen and Hector Levesque in their work on speech-act theory [Cohen 80, Cohen 85]. However, such an endeavor is extremely difficult, since it relies upon a much better understanding of the fundamental nature of cooperativity and rationality than we currently have. To develop this understanding, we may need to do some bootstrapping. That is, we may first need studies, like the present one, that explicate the phenomena that a theory of cooperativity should accommodate.[55]

[55]Similarly, Cohen and Levesque's work probably could not proceed if the notions of speech acts that they are trying to accommodate had not already been well considered. And bootstrapping is not a terrible thing; as Goldman says of it, it is "an inevitable, and hence not very objectionable, feature of most philosophical theorizing" [Goldman 70,p. 19].

5.2. Unexecutable Acts

Recall Definition (P1). Its Clause (i) guaranteed that when an agent has some simple plan, he believes that all the acts he intends as part of it are executable. The first way in which a plan may be judged invalid is for the inferring agent to believe that one of those acts is, in fact, unexecutable.

Examples of Plans with Unexecutable Acts

Let us consider some examples. Recall the plan diagrammed in Figure 4-5. Suppose that an agent R has inferred this to be the plan that another agent Q has. Then R must believe that Q believes, *inter alia*, that he can execute, at time t_2, all three of the acts shown in the figure. We can express these beliefs as follows:

(i) BEL(R, BEL(Q, EXEC(Q, do-to(load(x),new-messages), t_2), t_1), t_1)
(ii) BEL(R, BEL(Q, EXEC(Q, do-to-fewer(load(x),all-messages), t_2), t_1), t_1)
(iii) BEL(R, BEL(Q, EXEC(Q, speed-up(enter-mail), t_2), t_1), t_1)

Further discussion of the act-type terms in this example can be found in Section 7.5, where the implementation of this example in SPIRIT is described. For now it suffices to observe that the act-type term in (i)--do-to(load(x), new-messages)--denotes loading only the new messages; the term in (ii), loading fewer messages; and the term in (iii), speeding up entry into mail. Then (i) represents the fact that R believes at time t_1 that Q believes that at time t_1 that he can cause only the new messages to be loaded at time t_2. Similarly, (ii) represents R's belief that Q believes that he can cause fewer message to be loaded, and (iii), R's belief that Q believes he can speed up entry time into mail.

Now R might herself believe that one or more of these acts is unexecutable. That is, any (or all) of (iv) through (vi) might be true.

(iv) BEL(R, $\neg$ EXEC(Q, do-to(load(x),new-messages), t_2), t_1)
(v) BEL(R, $\neg$ EXEC(Q, do-to-fewer(load(x),all-messages), t_2), t_1)
(vi) BEL(R, $\neg$ EXEC(Q, speed-up(enter-mail), t_2), t_1)

Each of (iv), (v), and (vi) is independent of the other two. So, for instance, only (iv) might be true: R might believe that there is no way to load only new messages into the mail system, but believe that you can load, say, only the flagged messages, and thereby load fewer messages than would normally be loaded. If R further believes that entry time into the mail system is proportional to the number of messages loaded, then she will, in this case, also believe that by loading fewer messages, Q can speed up entry time into mail, i.e., that Q's speeding up entry time into mail is executable.

Or only (v) might be true: R might believe that new messages are all that are ever loaded into mail. Thus R will believe that Q's loading only new messages is an executable act, but that loading fewer messages than normal is not. And R might believe that there is no way to do that. But she might further believe that Q's speeding up entry time into mail is executable, say if there is a quick-load switch that Q can set.

Or only (vi) might be true: R might believe that there is a way to load only the new messages, and that by doing that, Q will be loading fewer messages. But R might further believe that entry time into the mail system is constant, and there is no way for Q to speed it up.

As an example of (iv) and (v) but not (vi) being true, R might believe that there is no way to load only the new messages, and in fact, no way to load fewer messages: all the messages in the mail file must always be loaded. But R might still believe that there is a way to speed up entry into mail, say by setting a quick-load switch.

I leave it to the reader to construct examples of the other combinations--(iv) and (vi); (v) and (vi); and (iv), (v) and (vi). The point of this discussion is to demonstrate that when an inferring agent believes an actor has some plan, it is in principle possible for her to believe that any of the intended acts it includes are unexecutable.

Bear in mind that an agent R need not consciously think of an inferred plan as including intentions to perform unexecutable acts; in fact, it is unlikely that most people have ever before thought of the term *unexecutable* in precisely the sense I am using it here.[56] Rather *unexecutability*, (and, later, *ill-formedness* and *incoherence*) are names that I am giving to certain configurations of beliefs that an agent might have, just as *plan* was defined to be a particular configuration of beliefs and intentions. The point of naming such a

[56]A similar claim about a plan being *ill-formed* is even stronger.

configuration of beliefs is to capture a regularity that is suggested by the analysis of discourse. The particular configuration of beliefs described here as "believing that some actor has a plan that includes an intention to perform an unexecutable act" correlates with the strategy of telling the actor, in response to his query, that some act cannot be done.

As an example, imagine Q asking the query which we first saw in Section 2.4:

> (15) Q: "How can I get mail to load only the new messages? I figure this would speed up entry time into it."

Depending on which of (iv), (v), and (vi) above are true, we might see responses of the following kinds. Example (16) illustrates a response in which only (iv) is true; Example (17), one in which only (v) is true; Example (18), one in which only (vi) is true; and Example (19), one in which (iv) and (v), but not (vi), are true. In each case, I have italicized the portion of the response that conveys the information that an act (or acts) in the inferred plan cannot be done.

> (16) R: "*There is no way for you to load only the new messages.* But you can load only the flagged messages by setting the flagged-msgs-only switch in your init file, and this will result in speeding up entry time into mail."

> (17) R: "New messages are all that are ever loaded; *there is no way for you to load fewer messages.* You could speed up entry time by setting the quick-load switch in your init file though."

> (18) R: "Well, you can invoke mail by typing 'mail /load:new', and then only the new messages will be loaded. *But entry time into mail is constant*, I'm afraid."

> (19) R: "*There is no way for you to load only the new messages; all your messages are always loaded.* But you can speed up entry time by setting the quick-load switch in your init file.

In Example (16), R conveys the fact that Q's loading only new messages is unexecutable; in Example (17), that Q's loading fewer messages than normal is unexecutable; and in Example (18), that Q's speeding up entry time into mail is unexecutable. The examples should demonstrate that such information may be conveyed implicitly: in Example (18), for instance, R does not explicitly assert that there is no way to speed up entry time into mail, though this fact can be deduced from her assertion that entry time is constant. Similarly, in Example (19), while R explicitly asserts that Q's loading only new messages is unexecutable, she conveys implicitly the information that his loading fewer messages than normal is also unexecutable.

Similar examples can be constructed for the other combinations of unexecutable acts in the inferred plan. I stress again that I am *not* claiming that whenever an agent infers a plan that she believes contains an intention to perform an unexecutable act, she will necessarily convey that information to the actor whose plan it is. I am not even claiming that she needs to do so to be cooperative. Rather, I claim that such information is often conveyed, and I am here providing an account of its source.

Weakening the Definition

Note that (iv) through (vi) express facts about R's belief that some act is not executable. This is a very strong belief; it commits R to a further belief that there is no sequence of acts that (1) will generate the act in question; (2) are all basic for Q; and (3) are such that Q is in the standard conditions with respect to each of them. This, after all, is how executability is defined in Definition (E1).[57] We might want to account for responses like those in Examples (16) through (19) on the basis of a slightly weaker belief. It seems reasonable to suppose that R will inform Q that some act in his inferred plan cannot be done when she does not know of a way to do it. R need not assent to an absolute belief that no sequence of basic acts will generate the act in question; all that is required is that she be unable to prove the existence of some generating sequence. To model this, we can equate the judgement of an act as unexecutable with the absence of a belief that it is executable, rather than with a belief that it is not executable:

Definition
(J1) $BEL(G, UNEX(\alpha,H,t_2), t_1) \equiv \neg BEL(G, EXEC(\alpha,H,t_2), t_1)$

[57] Alternatively, R could believe that the act in question is of a domain-basic type, but that at its intended performance time Q is not in the standard conditions with respect to it, or it is not in Q's repertoire. I will assume, however, that agents do not ask how to perform basic acts.

Then we associate the response strategy of informing the actor that some act in his plan cannot be done with a judgement of unexecutability of that act.

Now BEL(G, $\neg$ EXEC(α,H,t_2), t_1) implies $\neg$ BEL(G, EXEC(α,H,t_2), t_1), since, by Axiom (B4), agents do not have inconsistent beliefs. Thus a belief that an act is not executable, as in (iv) through (vi), entails a belief that it is unexecutable, i.e.,

(J2) BEL(G, $\neg$ EXEC(α,H,t_2), t_1) $\rightarrow$ BEL(G, UNEX(α,H,t_2), t_1).

The converse, however, is not true: an agent may believe that an act is unexecutable without having the belief that it is not executable.

In effect what I have done here is to adopt notion of negation as failure. I claim that in reasoning about the plan that underlies some query, an inferring agent R may, after some reasonable amount of effort, simply abandon her attempt to find a generating sequence of acts, and therein automatically come to be in the state that I describe as believing some act to be unexecutable. I have not described how R searches for the generating sequence; I take this to be the central problem of AI planning theory, which might be solved by any of the typical methods--backwards chaining, regression, bidirectional search, canned solutions, etc.--or by some yet to be developed alternative. However, there is still a further complication here.

In attempting to find a sequence of basic acts that generates some act of β, R needs to do four things. She needs to find a sequence of act-types α_1;. . .; α_n that conditionally generates β; she needs to determine whether these act-types are in Q's repertoire at the intended performance time of his plan; she needs to determine whether Q will be in the standard conditions with respect to each of them at the intended performance time of his plan; and she needs to determine whether the condition C under which they conditionally generate β will indeed hold at intended performance time. But it is extremely difficult to state precisely how this last task can be done. This is because, if a condition C will not otherwise hold, Q may himself be able to bring C about. But, as mentioned in Section 4.4, to model the process by which R determines whether Q can bring C about, a careful theory is required that distinguishes between events that are within the agent's control and those that are not. Richard Pelavin is developing such a theory [Pelavin 86]. Further, to use such a theory in cooperative communication, an account of *reasonableness* is needed. R must determine whether it is reasonable for the agent to attempt to bring about some inferred generation-enabling condition in order to execute the act in his plan. It is not reasonable, for instance, for the agent Q in Example (1) to attempt to get Kathy back to the hospital, despite the fact that it may be within his power to do so. I take the question of what it is reasonable to attempt to plan for to be a general, and very difficult, problem for AI planning systems.

Naturally Occurring Examples

It is worthwhile to consider briefly a few naturally occurring dialogues in which information about unexecutable acts is conveyed. The dialogues in this section are all taken, verbatim, from the mail transcripts discussed in Chapter 1. The queries in the first three examples were all mentioned in Section 2.4. I have again italicized the relevant portion of each response. Natural examples are very rich sources of information, and are open to considerable analysis. I attempt here to give just a very informal demonstration of the phenomenon of conveying information about unexecutable acts.

The first example, (20), will be quite familiar, and needs no further discussion:

(20) Q: "Is there any way to tell a bboard to load only new messages (I would like to speed up entry into infomac)?"
R:[58] "*As far as loading only new messages, Mail cannot do this* because it parses the entire message file whenever you start up. . . I will clean out the infomac bboard (moving old messages to an archive file) so that it will load faster."

In Example (21), R informs Q that an act he explicitly asks about is unexecutable:

[58] In the actual transcript, Q sent two questions to R, and R returned a message containing answers to both. The answer quoted in Example (20) is the second one; that is why it begins with "As far as ..."

(21) Q: "How can I scroll up and down in mail? I usually forget what number mail I'm looking at and if the info I need is off the screen, I can't get to it easily."
R: ". . .*Mail can't do anything about scrolling for you.* However, the last message you were working with (for example, reading) is called the "current message". Mail lets you refer to the current message without knowing its number. In place of a number, you type a period ("."). . ."

The third example, (22), is also an obvious case of R informing Q that an act about which he asks is unexecutable. Note that in this case R probably believes that other acts in Q's plan, such as decreasing the startup time by taking the symbol defs out the mailinit file, are also unexecutable, but R does not explicitly convey this information.

(22) Q: "[I]s there anyway to dynamically load a file of definitions and commands and have them interpreted by mail? I've wanted to do this so I could have a large file of net addresses and aliases that I would manually load if I wanted to send mail to one of them. I figgure *(sic)* this would decrease the startup time for mail, since I would take all the symbols out of my mailinit file."
R: "*There is currently no way to dynamically load a file of definitions.* . ."

In the next example, (23), R informs Q that an act about which she asks--telling mail to deliver a message after some particular time and/or date--is unexecutable. However, she believes that Q's inferred goal of actually having a message delivered after some particular time and/or date--*is* executable, and she in fact tells Q how to do that. This example also provides support for the weakened definition of unexecutability. R's response is that she "can't really see any way to do" the act in question, which at least suggests a weaker belief than absolute certainty that there is no way to do it.

(23) Q: "Is there a way to tell mail that I'd like to deliver a message after some particular time and/or date? I couldn't see anything in the documentation that hinted at this, maybe I just missed it."
R: "*That's an easy one--no. I can't really see any easy way to do this either (and guarantee it).* If you really wanted to do that you could submit a batch job to run after a certain time and have it send the message."

And one final example, (24), is similar to the last one in that R informs Q that his plan contains an unexecutable act, but that his goal act is executable.

(24) Q: ". . .Is there some way to mark messages other than to actually look at them or print them? Thanks!"
R: "I don't quite know what you mean by 'mark a message'? Do you mean, make mail think you've read the message?"
Q: "Yes, I meant to be able to mark a message as having been read without needing to actual *(sic)* do so."
R: "*There's no neat way to mark a message "read" without reading it.* It is possible to delete a message without reading it. Also, I don't know if you're aware of this but once you get into Mail, whether or not you read the messages it tells you are new you'll never be notified about them again outside of Mail (like when you log in). If you do want to delete messages without reading them you probably would want to put the following line into you "mailinit" file: >>>set no warn-on-deletion. This tells Mail not to ask for confirmation before deleting a new message. . .

5.3. Ill-Formed Plans

In the previous section, I identified the first way in which a plan can be judged invalid: there can be a discrepancy between the inferring agent's beliefs and a belief about executability, corresponding to Clause (i) of Definition (P1), that she ascribes to the actor. In this section, I will consider the second way in which a plan can be judged invalid: the inferring agent may believe false some belief about generation, corresponding to Clause (ii) of Definition (P1), that she ascribes to the actor. The inferring agent will in this case believe that the actor believes that two of his intended acts will form a generational pair, while the inferring agent herself does not believe that this will be so. In this case, I will say that the inferring agent believes that the plan is *ill-formed.* And to reiterate I point I made previously: to say that an agent believes a plan is ill-formed is to describe a configuration of beliefs that agent has.

Examples of Ill-Formed Plans

Recall the by-now familiar case of the actor Q who wants to find out the phone number of the hospital so he can call Kathy. Figure 5-1 shows the relevant portion of the plan that Q may be inferred to have. Q intends to call the hospital, and by doing this act to generate an act of establishing a communication channel to Kathy.

$$\begin{tikzcd}
\texttt{establish-channel(Kathy)} \\
\texttt{call(hospital)} \arrow[u]
\end{tikzcd}$$

Figure 5-1: Q's Plan to Establish a Communication Channel to Kathy

The inferring agent R may believe that each of the intended acts in this plan is executable: she may believe that Q can call the hospital, and that he can establish a communication channel to Kathy. But if R believes that Kathy has already gone home from the hospital, she will not believe that Q will do the latter by doing the former: i.e., she will not believe that the former act will generate the latter. This belief is then at odds with the belief that R must ascribe to Q, in accordance with Clause (ii) of Definition (P1), when she believes that he has the simple plan whose acts are diagrammed in Figure 5-1. In the representation language, the beliefs that are in conflict are:

(vii) BEL(R, BEL(Q, GEN(call(hospital), establish-channel(Kathy), Q, t_2), t_1), t_1)
(viii) BEL(R, $\neg$ (GEN(call(hospital), establish-channel(Kathy), Q, t_2), t_1)

The configuration of beliefs I describe as "believing that some agent has a plan that is ill-formed" correlates with the response strategy of telling an agent that the acts he intends will not lead to his goal act. Aravind Joshi, Bonnie Webber and Ralph Weischedel, in their paper on conversational strategies in question-answering [Joshi 84], classify similar behavior under the rubric of a response to "an unavailing act," but there are significant differences between their treatment and my own, which I will discuss in Section 5.5. Often information about ill-formedness may be conveyed implicitly: rather than merely asserting that an intended act will not lead to the expected goal, R may state *why* it will not do so (and from that, expect Q to infer *that* it will not do so). In the example under consideration, then, one cooperative response--in fact, the one given in Example (1)--might be:

> (25) R: "*She's already been discharged.* Her home number is 555-1238.

Again, the italicized portion of the response is what conveys information about ill-formedness. Here R explains why the intended act of calling the hosptial will not generate the goal act.

Just as a plan may be judged to include several intentions to perform unexecutable acts, a plan may be judged to be ill-formed in several ways. Consider once more the plan diagrammed in Figure 4-5, inferred to underlie query in Example (15). When R believes that this is Q's plan, not only are sentences (i) through (iii), discussed in the previous section, true, but so are the following two sentences:

```
(ix)  BEL(R, BEL(Q, GEN( do-to(load(x),new-messages),
                         do-to-fewer(load(x),all-messages),
                         Q,t_2),
                    t_1), t_1)

(x)   BEL(R, BEL(Q, GEN( do-to-fewer(load(x),all-messages),
                         speed-up(enter-mail),
                         Q,t_2),
                    t_1), t_1)
```

Propositions (ix) and (x) are guaranteed by Clause (ii) of Definition (P1). But R might have beliefs incompatible with both of these, i.e.,

```
(xi)  BEL(R, ¬ GEN(do-to(load(x),new-messages),
                   do-to-fewer(load(x),all-messages),
                   Q,t_2),
```

t_1)

(xii) BEL(R, $\neg$ GEN(do-to-fewer(load(x),all-messages),
speed-up(enter-mail),
Q,t_2),
t_1)

Either of (xi) or (xii) is sufficient to describe R as believing Q's plan Π to be ill-formed (provided R also has the beliefs necessary to describe her as believing that Q has the plan diagrammed in Figure 4-5). Example (26) shows the sort of response we might see when both (xi) and (xii) are true:

(26) R: "Well, new messages are all that are ever loaded, *so by doing that you won't load any fewer messages.* And anyway, in this mail system, unlike MM, *entry time does not depend upon the number of messages loaded.*"

Here R explicitly conveys the information that loading only new message will not generate loading fewer messages, and she implicitly conveys the information that loading fewer messages will not speed up entry time into mail.

We can now define ill-formedness. The definition will be intentionally weak in exactly the same manner as was the definition of unexecutability. Instead of requiring that the inferring agent believe that the beliefs about generational pairs she ascribes to the actor are false, she need only lack a belief that they are true:

Definition
(J3) BEL(R, ILL-FORMED(Q, α_n, [α_1, . . ., α_{n-1}], t_2, t_1), t_1)
$\equiv$
BEL(R, SIMPLE-PLAN(Q, α_n, [α_1, . . .,α_{n-1}], t_2, t_1), t_1) $\wedge$
[$\neg$ BEL(R, GEN(α_i,α_{i+1},Q,t_2), t_1)] for some i=0,...,n-1

Definition (J3) states that we can describe R as believing at t_1 that Q's simple plan (at t_1 to do β by doing the α_i's at t_2) is ill-formed if and only if R believes that Q has the simple plan, and that one of the generational pairs Q intends as part of it will not actually form a generational pair. Note that where unexecutability applies to acts, ill-formedness applies to plans.

Unexecutability and Ill-Formedness

It is important to observe the interaction between unexecutability and ill-formedness. The question of whether or not a plan is well-formed is (nearly completely) independent of the question of the executability of the intended acts it includes. Plans can contain intentions to perform unexecutable acts but still be well-formed; or they can be ill-formed but contain only intentions to perform executable acts; or there can be arbitrary combinations of unexecutable acts and ill-formedness. To demonstrate this, I will focus on the executability of the goal act and of the act that is explicitly queried in a request for information. Seven of the eight possible combinations of a plan's being well-formed or ill-formed, having an executable or unexecutable queried act, and having an unexecutable or unexecutable goal act, are logically consistent; I will provide examples of all seven possibilities.

To begin, consider Examples (27) and (28) (which are variations of Examples (3) through (6) discussed in Chapter 1). These dialogues illustrate queries with inferred underlying plans that are ill-formed; in them, R believes that setting the permission of some file will not lead to preventing Tom, the system manager, from accessing that file.

(27) Q: "I want to prevent Tom from reading my file. How can I set the permissions on it to faculty-read only?"
R: "Well, the command is
SET PROTECTION = (Faculty:Read)
but it won't keep Tom out. He's the system manager so he can always access files."

(28) Q: "I want to prevent Tom from reading my file. How can I set the permissions on it to faculty-read only?"
R: "Sorry, there's no way for you to do that, and even if you could it wouldn't keep Tom out. He's the system manager so he can always access files."

In both cases, not only is the inferred plan ill-formed, but also the goal act, of preventing Tom from reading Q's files, is unexecutable. But where the queried act in Example (27), of setting the permissions to group read only, is judged to be executable, the queried act in Example (28), of setting the permissions to faculty read only, is not. So, ill-formedness is independent of the executability of the queried action.

Example (1), repeated below as Example (29), demonstrates that ill-formedness is also independent of the executability of the goal action: the query it contains has an inferred underlying plan which is ill-formed, but which has a goal act, of talking with Kathy (or, in the simple case, of establishing a communication channel with Kathy), which is executable. This contrasts with Examples (27) and (28), just above, which have ill-formed plans in which the goal action is unexecutable.

(29) Q: "I want to talk to Kathy. Do you know the phone number at the hospital?"
R: "She's already been discharged. Her home number is 555-8321."

So ill-formedness is independent both of the executability of the queried act and of the goal act. Also, in ill-formed plans, executability of the queried and of the goal act are independent of one another: in Example (1) both are executable, in Example (27) only the queried act is executable, in Example (30), below, only the goal act is executable, and in Example (28) neither is executable.

(30) Q: "How can I get mail to load only the new messages? I figure this would speed up entry into it."
R: "There is no way for you to do that, and even if you could, it wouldn't speed up entry time because entry time doesn't depend on the number of messages loaded. What you can do to make it faster is to set the quick-load switch in your init file."

What about well-formedness? Certainly there are well-formed plans with executable queried acts and executable goal acts; these are the valid plans. Example (31) illustrates a response to a query that is inferred to have an underlying valid plan.

(31) Q: "I want to talk to Kathy. Do you know the phone number at the hospital?"
R: "It's 555-8321."

The query-response pair formed by Examples (15) and (16), repeated here in Example (32), provides an example of well-formedness where the goal, but not the queried, act is executable. And Example 33 is an example of a well-formed plan in which the neither the queried nor the goal act is executable:

(32) Q: "How can I get mail to load only the new messages? I figure this would speed up entry into it."
R: "There is no way for you to load only the new messages. But you can load only the flagged messages by setting the flagged-msgs-only switch in your init file, and this will result in speeding up entry time into mail."

(33) Q: "I'd like to take advantage of the discount offer you sent me, and subscribe to your magazine. How can I specify that I don't want the magazines to start arriving until October, since I'm going to be moving before then?"
R: "Sorry, you can't do that. To get the discount rate you have to take the subscription before August 31st, and once you take it, the magazines will start arriving automatically within 3 weeks."

There is, in fact, only one exception to the claim that well-formedness, executability of the queried act, and executability of the goal act are mutually independent. It is logically impossible to have a well-formed plan in which the queried act is executable, while the goal act is not. It should be intuitively clear why this case cannot occur: if R believes that the queried act can be done, and further, R believes that the queried act will generate the goal act, then R must believe that the goal act can be done by doing the queried act. The relation between executability and generation in question here is exactly that encoded in Theorem (E2).

Figure 5-2 summarizes the possible combinations of the features of well-formedness and executability of the queried and goal acts, and for each logically consistent combination, gives the number of an example dialogue that illustrates it.

Naturally Occurring Examples

I will briefly present a few naturally occurring examples of dialogues in which information is conveyed about acts not fitting together in the intended manner. These examples are again taken verbatim from transcripts, but this time they come from several different sources.

Example 34 is the original motivation for the calling-Kathy example already discussed several times. The question and response were actually a pair of messages sent through an electronic mail system.

formedness	queried action executable	goal action executable	example #
well	yes	yes	31
well	yes	no	*inconsistent*
well	no	yes	32
well	no	no	33
ill	yes	yes	29
ill	yes	no	27
ill	no	yes	30
ill	no	no	28

Figure 5-2: Combinations of Types of Invalidities

(34) Q: "I called her [Kathy] 10 times at least. The phone at the hospital is always busy! Do I have the right number? Maybe you can give it [to] me again."
R: "I don't know why the number at the hospital is always busy, but in any case, *Kathy's at home now.* Her number there is [(xxx)xxx-xxxx].

As we have seen, R conveys to Q in this response the reason that she believes Q's plan to be ill-formed: Kathy is at home; hence, calling the hospital will not have its intended effect.

Another example comes from the train-station examples collected by Horrigan [Horrigan 77]:

(35) Q: "Track eleven?"
R: "Track eleven?"
Q: "Yeah."
R: "Ah, you work around here?"
Q: "No."
R: "What do you want to go to track eleven for?"
Q: "There's an employment office there. CP.[59]"
R: "*CP Employment is behind gate nine.* Back there."
Q: "The gate nine."
R: "Behind gate nine."

An intuitive explanation of this dialogue is that R infers that Q's intended act of going to track eleven will not generate his goal of going to the employment office. (In fact, at first it seems that R cannot figure out any reasonable goal that Q may have that would be served by his going to track eleven). Despite the belief that the plan is ill-formed, R might well believe that its intended acts--of going to track eleven, and of going to the CP employment office--are both executable.

A third example comes from a set of transcripts collected at Columbia University in which students and faculty advisors meeting to discuss course registration were recorded[60]:

(36) Q: "...I'll take Fundamental Algorithms next semester, so ... I thought I'll take Operating Systems or some other course this semester."
R: "You can do that *except Fundamental Algorithms is a 3000 level prerequisite so usually it makes sense to take this course before the electives.*"

Intuitively, the inferring agent here is informing the actor that while his queried act is executable--he can wait and take Fundamental Algorithms next semester--doing so will probably not lead to his inferred goal of completing his coursework in an optimal manner. Of course, more detailed analysis of this example depends on a theory of optimality.

[59] Canadian Pacific

[60] My thanks to Kathy McKeown for a copy of the transcripts.

5.4. Incoherent Queries

So far, I have described the sort of invalidities that an inferring agent R may find in an actor's Q's plan: she may judge it to be ill-formed, and she may judge it to contain one or more intentions to perform an unexecutable act. When R judges a plan to be invalid in one of these ways, she has intuitively "made sense" of the plan, and understands the source of the invalidities. As just one example, when R judges the plan diagrammed in Figure 5-1 to be ill-formed, she determines that Q may mistakenly believe that Kathy is at the hospital, and consequently, that by calling there Q can establish a communication channel to her.

There are also cases in which R may not even be able to "make sense" of Q's plan. As a somewhat whimsical example, imagine Q saying:

(37) Q: "I want to talk to Kathy, so I need to find out how to stand on my head."

In many contexts, a perfectly reasonable response to this query is "Huh?". Q's query is *incoherent*: R cannot understand why Q believes that finding out how to stand on his head (or standing on his head) will lead to talking with Kathy. One can, of course, construct scenarios in which Q's query makes perfect sense: Kathy might, for example, be currently hanging by her feet in gravity boots. The point here is not to imagine such circumstances in which Q's query would be coherent, but instead to realize that there are many circumstances in which it would not.

Unfortunately, the model as developed so far does not distinguish between a query of this type and one in which the inferred underlying plan is ill-formed. The reason is that, given a reasonable account of semantic interpretation, it is transparent from the query in Example (37) that Q intends to talk to Kathy, intends to find out how to stand on his head, and intends his doing the latter to play a role in his doing the former. Further, as consequences of these intentions Q believes that he can talk to Kathy, believes that he can find out how to stand on his head, and believes that his doing the latter will play a role in his doing the former.[61] But these beliefs and intentions are precisely what are required by

[61] Recall the discussion in Section 3.3 about assuming that an intention to α entails a belief that the agent will do α.

Definition (P0) to have a plan; and if R could determine that the intended role of the supporting act--of standing on his head--was generation, then these beliefs and intentions would also be exactly what is required by Definition (P1). Consequently, after hearing Example (37), R can in fact infer a plan underlying Q's query, namely the obvious one: to find out how to stand on his head in order to talk to Kathy. Then since R does not herself believe that the former act will lead to the latter, on the analysis so far given, we would regard R as judging Q's plan to be ill-formed. Unfortunately, this is not the desired analysis: the model should instead capture the fact that R cannot make sense of Q's query here--that it is *incoherent.*

Let us consider another example. Recall again the query from Example (28), repeated here as Example (38):

> (38) Q: "I want to prevent Tom from reading my file. How can I set the permissions on it to faculty-read only?"

In interpreting this query, R may come to have a number of beliefs about Q's beliefs and intentions. Specifically, all of the following may be true:

(xiii) $BEL(R, BEL(Q, EXEC(set\text{-}permissions(file1,read,faculty),Q,t_2), t_1),t_1)$

(xiv) $BEL(R, BEL(Q, EXEC(prevent(file1,read,Tom),Q,t_2),t_1),t_1)$

(xv) $BEL(R, BEL(Q, GEN(\ set\text{-}permissions(file1,read,faculty), prevent(file1,read,Tom), Q, t_2), t_1), t_1)$

(xvi) $BEL(R,INT(Q,set\text{-}permissions(file1,read,faculty),t_2,t_1),t_1)$

(xvii) $BEL(R,INT(Q,prevent(file1,read,Tom),t_2,t_1),t_t)$

(xviii) $BEL(R,INT(Q,by(\ set\text{-}permissions(file1,read,faculty), prevent(file1,read,Tom), t_2,t_1), t_1)$

But (xiii) through (xviii) are sufficient for (xix):

(xix) $BEL(R, SIMPLE\text{-}PLAN(Q, prevent(file,read,tom), [set\text{-}permissions(file1,read,faculty)], t_2, t_1), t_1)$

This much is not surprising. In effect Q has stated in his query what his plan is--to prevent Tom from reading the file by setting the permission on it to faculty-read only--so of course R should be able to infer just that. Now suppose

R believes that the system manager can override file permissions, that Tom is the system manager, but also that Q does not know that Tom is the system manager. In that case, R can judge the plan to be ill-formed: there is a discrepancy between the belief about generation she ascribes to Q as expressed in (xv), and her own beliefs. In this case, a response such as that given in Example (28) would be appropriate.

But what if R instead believes that it is mutually believed by Q and R that Tom is a faculty member? In that case, while (xiii) through(xix) may still be true, and while there may still be a discrepancy between R's own beliefs and the belief expressed in (xv), we do not want to say that this case is indistinguishable from the previous one. In the previous case, R understood the source of Q's erroneous belief: she realized that Q did not know that Tom was the system manager and, therefore, thought that by setting permissions to exclude him, he could prevent Tom from reading the file. In the current case, R cannot really understand Q's plan: she cannot determine why Q believes that he will prevent Tom from reading the file by setting the permissions on it to faculty-read only, given that Q believes that Tom is a faculty member. This current case is like the case in Example (37): Q's query is incoherent to R.

To capture the difference between ill-formedness and incoherence, I will claim that when an agent R is asked a question by an actor Q, R needs to attempt to ascribe to Q more than just a set of beliefs and intentions satisfying Definition (P1). Specifically, for each belief satisfying Clause (ii) of Definition (P1), R must also ascribe to Q another belief that *explains* it.[62] So, for instance, in Example (1), R can ascribe to Q the beliefs that (1) by calling a location, one can establish a communication channel to an agent who is at that location, and (2) Kathy is at the hospital. The conjunction of these two beliefs explains Q's belief that by calling the hospital he will establish a communication channel to Kathy. Analogously, in Example (28) R can ascribe to Q the beliefs that (1) by setting the permissions on a file to restrict access to a particular group, one prevents from so accessing the file everyone whose is neither a member of that group nor the system manager and (2) Tom is neither a member of the faculty

[62]It may sometimes be the case that R also needs to find a belief explaining Q's belief about executability. This would explain R's saying to a headless robot "Huh, how can you think you can stand on your head? You don't have one." (I thank Bonnie Webber for this example.) Study of naturally occurring dialogues, however, has not revealed widespread need for including this additional type of explanatory belief in the analysis. It is an open question why this is so.

nor the system manager. The conjunction of these two beliefs explains Q's belief that by setting the permissions to faculty-read only he can prevent Tom from reading the file.

In contrast, in Example (37), R has no basis, independent of the semantics of Q's query itself, for ascribing to Q beliefs that will explain why he thinks that standing on his head will lead to talking with Kathy. And in the version of Example (38) in which R believes that Q believes that Tom is a faculty member, R has no basis for ascribing to Q a belief that explains Q's belief that setting the permissions to faculty-read only will prevent Tom from reading the file.

R does not necessarily believe the explanatory beliefs herself; rather, she believes that Q believes them. She must also believe of each explanatory belief that, if it were true, then the intended generational pair it explains in Q's plan would in fact form a generational pair. Then the question arises: what form do explanatory beliefs have? Certainly they will satisfy their requirements and provide the necessary explanation if they are of the following form: act-type α conditionally generates act-type β under condition C, and condition C will hold at the performance time of the inferred plan, i.e.,

$$\text{CGEN}(\alpha,\beta,\text{C}) \wedge \text{HOLDS}(\text{C},t_2)$$

where t_2 is the performance time of the plan. As we will see, we can get quite far with explanatory beliefs of this form, though it is possible that we may want later to add other forms of explanatory beliefs as well.

The relevant explanatory belief for Example (1) then is:

(xx) $\text{BEL}(\text{R}, \text{BEL}(\text{Q},\text{CGEN}(\text{call}(\text{X}), \text{establish-channel}(\text{Y}), \text{at}(\text{X},\text{Y})) \wedge \text{HOLDS}(\text{at}(\text{hospital},\text{Kathy}), t_2), t_1), t_1)$

The belief that R ascribes to Q in (vii) is explained by (xx): by Definition (G4) the former is a consequence of the latter.

Similarly, the explanatory belief for Example (28) is:

(xxi) $\text{BEL}(\text{R}, \text{BEL}(\text{Q},\text{CGEN}(\text{set-permissions}(\text{X},\text{P},\text{Y}), \text{prevent}(\text{X},\text{P},\text{Z}), \neg\,\text{member}(\text{Z},\text{Y}) \wedge \neg\,\text{system-manager}(\text{Z})) \wedge \text{HOLDS}(\neg\,\text{member}(\text{Tom},\text{faculty}) \wedge \neg\,\text{system-mgr}(\text{Tom}),t_2), t_1), t_1)$

Explanatory Plans

I will introduce a new theoretical construct into the model: *explanatory plans*, or *eplans*. Saying that an agent R believes another agent Q has some eplan is again shorthand for describing a set of beliefs possessed by R. Specifically:

Definition

(P2) $BEL(R, EPLAN(Q,\alpha_n,[\alpha_1, \ldots,\alpha_{n-1}],$
$[\rho_1, \ldots,\rho_{n-1}],t_2,t_1),t_1) \equiv$

(i) $BEL(R,BEL(Q,EXEC(\alpha_i,Q,t_2),t_1), t_1)$ for $i=1,\ldots,n-1 \wedge$

(ii) $BEL(R,BEL(Q,GEN(\alpha_i,\alpha_{i+1},Q,t_2),t_1),t_1)$ for $i=1,\ldots,n-1 \wedge$

(iii) $BEL(R,INT(Q,\alpha_i,t_2,t_1),t_1)$ for $i=1,\ldots,n-1 \wedge$

(iv) $BEL(R,INT(Q,by(\alpha_i,\alpha_{i+1}),t_2,t_1),t_1)$ for $i=1,\ldots,n-1 \wedge$

(v) $BEL(R, BEL(Q, \rho_i, t_1), t_1)$ for $i=1,\ldots,n-1$
where each ρ_i is $CGEN(\alpha_i,\alpha_{i+1},C_i) \wedge HOLDS(C_i,t_2)$

Clauses (i) through (iv) of Definition (P2) are identical to Clauses (i) through (iv) of Definition (P1); Clause (v) adds the newly required explanatory beliefs. Given Definition (P2), I will claim that one can model the job of the inferring agent in cooperative question-answering as an attempt to find an eplan that underlies the actor's query. In the next chapter, I will describe how this inference process itself can be modeled. For now, let us consider how the claim that eplans must be inferred affects the account of invalidity developed so far in this chapter.

The account of executability given in Section 5.2 does not change. Eplans, like plans, include beliefs that the intended acts they include are executable. The agent who believes that an actor has some eplan can still believe that one or more of these acts is instead unexecutable.

Similarly, the account of ill-formedness given in Section 5.3 can be maintained when we model agents as inferring eplans. Eplans, like plans, include beliefs that the intended acts form generational pairs, and the inferring agent may still believe that one or more of these beliefs is erroneous. So we can modify Definition (J3) fairly directly to refer to an eplan instead of a plan:

Definition

(J3--modified version)

$BEL(R, ILL\text{-}FORMED(Q, \alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1), t_1)$

$\equiv$

$$BEL(R, EPLAN(Q, \beta,[\alpha_1,...,\alpha_{n-1}],[\rho_1, ..., \rho_{n-1}], t_2, t_1), t_1) \wedge$$
$$[\neg BEL(R, GEN(\alpha_i,\alpha_{i+1},Q,t_2), t_1)] \text{ for some } i=0,...,n-1$$

Note that when R believes that one of Q's beliefs about generation, corresponding to Clause (ii) of Definition (P2), is incorrect, she will also believe that the associated explanatory belief corresponding to Clause (v) is incorrect. That is, if for some α_i and α_{i+1}

(xxii) $BEL(R, BEL(Q, GEN(\alpha_i,\alpha_{i+1},Q,t_2),t_1),t_1)$

is true and is part of R's belief that Q has some eplan, then it is also true that

(xxiii) $\exists\, C_i[BEL(R, BEL(Q, CGEN(\alpha_i,\alpha_{i+1},C_i) \wedge\ HOLDS(C_i,t_2), t_1),t_1)]$

But then, if R believes Q's plan is ill-formed because the belief in (xxii) is incorrect, i.e.,

(xxiv) $\neg BEL(R,GEN(\alpha_i,\alpha_{i+1},Q,t_2),t_1)$

it must also be true that R does not believe the explanatory belief in (xxiii), i.e.,

(xxv) $\neg BEL(R, CGEN(\alpha_i,\alpha_{i+1},C_i) \wedge\ HOLDS(C_i,t_2), t_1)$

When R disbelieves the explanatory belief, she may disbelieve the conditional generation relation, or she may disbelieve the fact that the generation-enabling conditions will hold at performance time, or both. In all three cases, I will classify R's belief as a judgement of ill-formedness, since in all cases it correlates with a response strategy of informing Q that his intended acts will not lead to his goal. Of course, if R chooses to convey to Q why his acts will not lead to his goal, she will convey different information depending on which part of the explanatory belief she disbelieves.

The following three possible responses to the query in Example (27) illustrate the contrast:

(39) R: "Well, the command is
SET PROTECTION =(Faculty: Read)
but Tom's the system manager."

(40) R: "Well, the command is
SET PROTECTION = (Faculty:Read)
but the system manager can override file protections."

(41) R: "Well, the command is **SET PROTECTION = (Faculty:Read)**
but Tom's the system manager, so he can override file protections."

In all three cases, sentences (xiii) through (xviii) may be true: these satisfy Clauses (i) through (iv) of Definition (P2). However there are different beliefs that R might ascribe to Q to satisfy Clause (v) of Definition (P2). Specifically, R may believe that Q believes, incorrectly, that Tom is not the system manager:

(xxvi) $\text{BEL}(R, \text{BEL}(Q, \text{CGEN}(\text{set-permission}(X,P,Y),\text{prevent}(X,P,Z), \neg\ \text{member}(Z,Y) \wedge \neg\ \text{system-mgr}(Z)) \wedge \text{HOLDS}(\neg\ \text{member}(\text{Tom}, \text{faculty}) \wedge \neg\ \text{system-mgr}(\text{Tom}), t_2)\ t_1),t_1)$

Sentence (xxvi) expresses R's belief that Q has the correct beliefs about the conditions under which setting the permissions on a file prevents someone from reading that file, but that Q is mistaken in thinking that Tom is not the system manager. When (xxvi) accurately represents R's belief a response such as Example (39) is appropriate. Here, the belief discrepancy concerns the generation-enabling condition in the explanatory belief: R believes it will not hold at the performance time of Q's plan, but believes that Q believes that it will hold.

Alternatively, there may be a discrepancy about the conditional generation relation itself. R may believe that Q mistakenly believes that whenever you set the permissions on a file to restrict a type of access to a certain group, you prevent everyone who is not a member of that group from so accessing the file:

(xxvii) $\text{BEL}(R, \text{BEL}(Q, \text{CGEN}(\text{set-permissions}(X,P,Y), \text{prevent}(X,P,Z), \neg\ \text{member}(Z,Y)) \wedge \text{HOLDS}(\neg\ \text{member}(\text{Tom},\text{faculty}), t_2),t_1),t_1)$

R herself thinks that this conditional generation relation is incorrect: she believes that when you set the permissions on a file, you restrict everyone who is neither a member of the group permitted access, nor the system manager. In this case, Example (40) might be an appropriate response.

Finally, R might believe that Q has both of the mistaken beliefs expressed in (xxvi) and (xxviii), and thus might generate a response like Example (41). In the next chapter, I will say more about how R can come to have the sort of beliefs represented in (xxvi) and (xxvii).

We now have an account of executability and formedness in eplans. All that remains is incoherence, and an account of that is quite straightforward. An inferring agent will judge an actor's query to be incoherent if she cannot infer an eplan that underlies it. Again we will want the weak definition: the presumption will be that a query is incoherent unless R can successfully find an eplan that may explain it[63] :

Definition

(J4) $\mathrm{BEL}(R, \mathrm{INCOHERENT}(Q, \theta), t_1) \equiv$

$$\neg \exists \alpha_1 \ldots \exists \alpha_n \exists \rho_1 \ldots \exists \rho_{n-1} [\mathrm{BEL}(R, \mathrm{EPLAN}(Q, \alpha_n, [\alpha_1, \ldots, \alpha_{n-1}], [\rho_1, \ldots, \rho_{n-1}], t_2, t_1), t_1) \wedge \mathrm{BEL}(R, \mathrm{UNDERLYING\text{-}EPLAN}(\theta, Q, \alpha_n, [\alpha_1, \ldots, \alpha_{n-1}], [\rho_1, \ldots, \rho_{n-1}], t_2, t_1), t_1)]$$

In the next chapter, I will define the UNDERLYING-EPLAN predicate, which associates queries and eplans, and will show how R can come to believe that an inferred eplan underlies some query. "Believing that a query is incoherent" correlates with the strategy of asking the actor, in response to his query, why he believes his intended acts will lead to his goal.

Coherence, then, is a property of queries, where being well- or ill-formed is a property of plans, and executability a property of acts. Figure 5-3 summarizes the object of each of the types of invalidities I have defined.

Naturally Occurring Example

A discourse fragment that can be analyzed in terms of R judging Q's query to be incoherent can be found in the first part of Example (35) repeated here for convenience.

(42) Q: "Track eleven?"

. . .

R: "Ah, you work around here?"

Q: "No."

R: "*What do you want to go to track eleven for?*"

[63] So inferring agents are harsher than the American system of justice.

Validity Property	Object
well-formed/ill-formed	plans
executable/unexecutable	actions
coherent/incoherent	queries

Figure 5-3: Validity Properties and their Objects

In this part of the dialogue, R seems to be unable to determine what Q's plan consists in. She cannot infer any reasonable goal that Q's getting to track 11 would support. As a result she requests further information to help her decide.

5.5. An Alternative Taxonomy

Throughout this chapter I have suggested a correlation between the types of invalidities found in plans inferred to underlie a given query, and the content of an appropriate response to that query. In their paper "Living Up to Expectations," Joshi, Webber and Weischedel [Joshi 84] present a taxonomy of response strategies that they pair with particular types of beliefs that an agent may have when she formulates a response to a question. Although the analysis I presented throughout this chapter is related to that given by Joshi, Webber and Weischedel (hereafter JWW), there are significant differences between the two approaches. In this section I will provide a brief comparison of them.

JWW begin by assuming that the intended goal of a given query can be inferred. They then identify eight situations in which the inferring agent (R) can "anticipate the possibility of the user/questioner (Q) drawing false conclusions from its response and hence alter it so as to prevent this happening" (p. 169). The eight situations they describe are listed in Figure 5-4. With each situation they associate what they believe to be an appropriate, nonmisleading response. Finally, they propose a case-analysis algorithm for the generation of appropriate responses to queries, in which a single situation is selected by R, and a response suited to that situation is then produced.

1. Failure of the enabling conditions: A Way
2. Failure of the enabling conditions: No Way
3. A Nonproductive Act
4. A Better Way
5. The Only Way
6. Something turning up: Unlikely Event
7. Something turning up: Likely Event
8. Something turning up: Event followed by action

Figure 5-4: Joshi, Webber and Weischedel's Situations

Each of JWW's categories corresponds to a particular set of beliefs that R might have. The categories that JWW identify and the invalidities that I have identified cover many, but not all, of the same conversational situations. Their Categories (1) through (3) correspond to configurations of invalidities I identify, though my treatment is somewhat different. Their Categories (4) through (8) are not covered in my account. Finally the invalidity I describe as "a query's being incoherent" is not accounted for in JWW's theory.

Underlying the different treatment of Categories (1) through (3) in the two theories is an emphasis on a holistic approach, in JWW's work, as a opposed to a modular approach in my own. JWW's Category (1)--"Failure of the enabling conditions: A Way"--corresponds to the configuration of beliefs that I have identified as believing that a plan is well-formed, but has an unexecutable queried act, though an executable goal act. Category (2)--"Failure of the enabling conditions: No Way"--corresponds to believing that a plan is well-formed, but has both unexecutable queried and unexecutable goal acts. And Category (3)--"A Nonproductive Act"--corresponds to believing that a plan is ill-formed and has an unexecutable queried act.

As an example of the difference between JWW's treatment and my own, note that I have argued that the formedness of a plan is independent of the executability of the plan's goal act. JWW do suggest that within their Category (3)--which corresponds to ill-formedness--the goal act may or may not be executable. However, because they use a case-analysis algorithm that maps directly from a particular category describing R's beliefs to an associated appropriate response, the only way for them to make the executability of the goal act independent of the formedness of the plan (or, in their terms, of the productivity of the proposed act) is to split the third category into two other categories.

None of Categories (4) through (8) are situations I have discussed in this thesis. Categories (4) and (5) involve meta-planning evaluations--of an act being the "best" way to perform another act, and of an act being the "only" way to perform another act--and these evaluations are beyond the scope of my thesis. The proper treatment of Categories (6) through (8) requires a well-founded theory of the likelihood of events occurring; otherwise these situations cannot be distinguished from those situations in which R judges the goal act to be unexecutable. Such a theory requires significantly more reasoning capabilities than are currently available in AI systems.

A final difference between the two approaches involves the type of information that each views as available to R in formulating her response. On JWW's account, R only has information about which of the eight categories obtains, information that translates, in my model, to which type(s) of invalidities are perceived. However, in the model I have presented, R has more information than this: she also has information about what the source of the invalidity is. For instance, when she judges the inferred plan to be ill-formed, she will necessarily also have beliefs about what beliefs of Q's explain that ill-formedness. Such information, I claim, will greatly inform R's response; even then, issues of relevance and salience will be needed to determine it fully.

To summarize, the principle differences between JWW's approach and my own involve (1) a holistic account of invalidities in JWW's approach vs. a modular account in my own; (2) the identification of several categories of plan invalidities in JWW's approach that are not accounted for in my own; (3) the identification of a way in which a query itself can be invalid--incoherence--in my approach that is not present in JWW's; and (4) a claim in my account that the respondent needs more information than just the type of the invalidities to generate an appropriate response. Despite these differences, there is an agreement in both works that providing cooperative responses to queries depends upon reasoning about invalidities that may be present in the plans underlying those queries.

CHAPTER VI
Inferring Simple Plans

In the previous chapter the notion of explanatory plans was introduced, and the claim made that one step in formulating a cooperative response to a question can be seen as an attempt to infer an eplan that underlies the question. This chapter begins by describing the relation between that process and the remaining components of the question-answering process as a whole. Next, it considers the encoding of English queries to serve as input to the plan inference mechanism. A model of the plan inference process itself is then developed. Specific plan inference rules are presented, and their use in inferring invalid, as well as valid, plans is described. Included in the chapter are several detailed examples of the entire plan inference process.

6.1. The Role of Plan Inference in Question-Answering

In this chapter, I will turn to the question of plan inference and will show how an agent can find a plan that underlies an actor's query, without assuming that it is valid. But plan inference is only one step in answering a question. Before developing a model of the plan inference process, I will briefly describe the question-answering process as a whole, showing how plan inference is related to its remaining components. Up until now I have frequently alluded to a model of cooperative question-answering and to the role of plan inference in it; in this section I will provide a more explicit depiction.

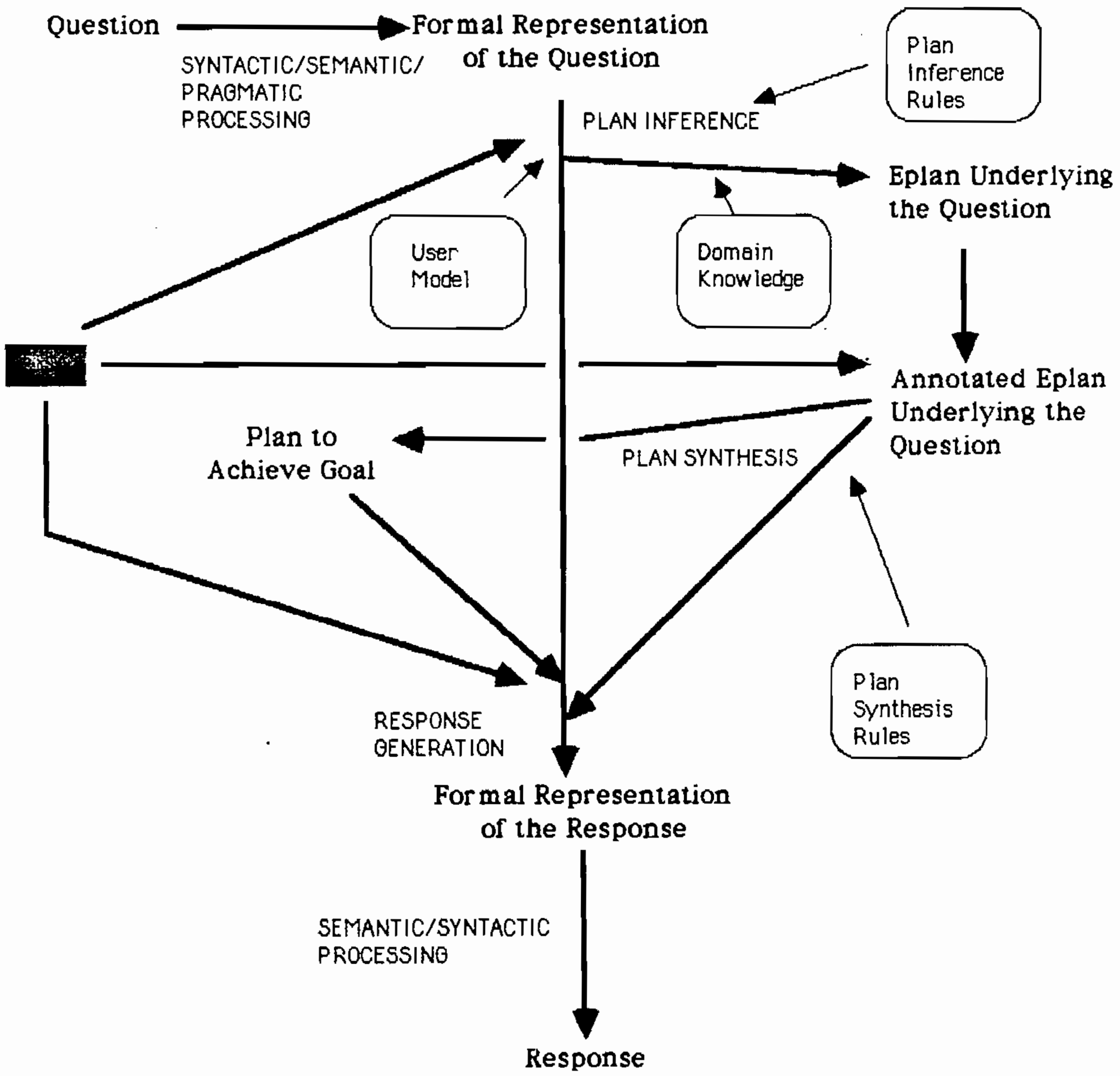
Figure 6-1 diagrams the question-answering process, which begins, naturally enough, with the question itself. I will suppose that any question posed in a natural language can be transformed to some formal representation prior to the process of domain-plan inference. The plan inference process as I model it operates on this formal representation of the question, making use of domain knowledge as well as knowledge of the plan inference process itself. The latter, which is encoded in the model in plan inference rules, includes knowledge about plans what are--what sort of beliefs and intentions they entail--and about how the beliefs that are included in another agent's plans are likely to diverge from one's own beliefs.

The plan inference process may be unsuccessful: the inferring agent, whom I will again call R, may fail to find an eplan that she believes underlies Q's question. According to the analysis given in Chapter 5, this failure corresponds to R's judging the query to be incoherent. When a query is so judged, further analysis is unlikely. It is not, however, impossible. If Q asks how to stand on his head in order to talk to Kathy, and R judges this query to be incoherent, she may, nonetheless, proceed to the plan synthesis stage to determine a plan for Q's standing on his head, and may convey the details of such a plan in her response. Telling Q, in these circumstances, how to stand on his head is unusual, but neither impossible nor necessarily uncooperative. So, R might say something like "I don't know why you need to stand on your head to talk to Kathy, but here's how to do it." Alternatively, the fact of a query's incoherence can serve directly as input to the response generation process, potentially resulting in R's asking for further information to assist her in another attempt at plan inference.

If plan inference, on the other hand, is successful, the result is an eplan that R believes underlies Q's query. Recall what this means: for R to believe that Q has some eplan is for R to have a particular configuration of beliefs about Q's beliefs and intentions. (For R further to believe that the eplan underlies Q's query, R must have additional beliefs about the relation between the eplan and the query. See Section 6.3.) In particular, R may have some beliefs of the form "Q believes X," where R herself does not believe X. Hence the next stage in the question-answering process is to evaluate the inferred eplan, to determine whether or not it is valid--i.e., well-formed and including only executable acts.

Although in Figure 6-1 the two processes--of inferring a plan and of determining whether it is well-formed--are shown to be sequential, they may in fact be interleaved. As R infers each piece of a potential eplan, she may note whether or not that piece is well-formed. In other words, each time R ascribes a potential belief to Q, she can note whether or not she herself has that belief. Then, when she finally settles on an eplan that she believes underlies Q's query she will already have determined whether or not it is well-formed. Alternatively, if it is expensive to compute whether a plan is well-formed, R may defer doing so until she has a complete eplan that she believes underlies Q's query. In either case, the result of the plan inference and evaluation processes can be viewed as

Figure 6-1: The Process of Answering a Question


Question
Formal Representation of the Question
SYNTACTIC/SEMANTIC/ PRAGMATIC PROCESSING
Plan Inference Rules
PLAN INFERENCE
Eplan Underlying the Question
User Model
Domain Knowledge
Annotated Eplan Underlying the Question
Plan to Achieve Goal
PLAN SYNTHESIS
Plan Synthesis Rules
RESPONSE GENERATION
Formal Representation of the Response
SEMANTIC/SYNTACTIC PROCESSING
Response

an annotated eplan in which the annotations mark any ill-formed portions of the eplan, along with the source of their ill-formedness. So for instance, in the familiar example of an agent's intending to call Kathy at the hospital, the inferred eplan will have annotations to the effect that the act of calling the hospital will fail to generate the act of establishing a communication channel to Kathy, and that the reason for this is that Kathy is no longer at the hospital. The annotated eplan is input to the response generation process.

In general, the evaluation of a plan to determine whether or not it is well-formed must be followed by a process of plan synthesis. Having figured out what Q intends to do, R must then figure out how Q can do it. It is during plan synthesis that R determines whether Q's plan contains any unexecutable acts since the attempt to find a way to execute an act issues in a decision as to whether that act is executable. An exception to the need for plan synthesis arises when Q's question merely asks whether his plan is well-formed--e.g., "Will typing **DEL** . cause the current message to be deleted?"--and his plan *is* so judged. Even then, plan synthesis may be desirable to determine whether Q's plan is optimal.

A word of caution is in order here. To say that R synthesizes a "plan" for Q to execute some act is to speak somewhat loosely, given the mental phenomenon view of plans, for what R synthesizes is not a set of actual beliefs and intentions, but rather a set of potential beliefs and intentions such that if Q adopts them, R believes that Q will thereby have a valid plan to execute the act.

Consider first plan synthesis when R has judged the eplan to be well-formed. In this case, if R can synthesize a (simple) plan for Q's queried act, she will have implicitly synthesized a plan for his goal act as well. If R judges the queried act to be unexecutable, she may attempt to synthesize an alternative plan for the goal act.

If instead R has judged the eplan to be ill-formed she may or may not attempt to determine a way for Q to execute his queried act. If she does, and succeeds, the synthesized plan will not be one that R believes will lead to Q's goal act: this is the significance of a judgement that the plan is ill-formed. Hence R may also attempt to synthesize a plan to execute the goal.

The alternative processes of plan synthesis are depicted in Figure 6-2. The triple $<\alpha$,Q,t$>$ represents the act that Q asks how to do, while $<\beta$,Q,t$>$

represents his goal act: either it is a goal he explicitly mentions in his query, or some other act that R determines to be a plausible goal. The arrow marked (1) denotes the inferred eplan, which may be judged to be well-formed or not. Note that to infer the plan denoted by (1), R reasons about what she believes Q believes; to determine whether it is well-formed or not, she compares those beliefs with her own beliefs. The arrow marked (2) denotes a plan to execute the queried act: the triple $<\gamma,Q,t>$ represents some act (or temporal concatenation of acts) that R believes both to be basic and to form a generational pair with the queried act. To infer the plan represented by (2), R must make use of her own beliefs about the domain. If (1) has been judged well-formed, then R will believe that $<\gamma,Q,t>$ also generates the goal act. To infer the plan denoted by the arrow labeled (3), R must again use her own beliefs about the domain: $<\gamma',Q,t>$ represents some act (or temporal concatenation of acts) that R believes is basic and generates the goal act. The plan denoted by (3) may be equivalent to the combination of (1) and (2) if (1) has been judged to be well-formed.

$$\begin{tikzcd}
<\beta,Q,t> & \\
<\alpha,Q,t> \arrow[u, "(1)"] & \\
<\gamma,Q,t> \arrow[u, "(2)"] & <\gamma',Q,t> \arrow[uul, "(3)"']
\end{tikzcd}$$

Figure 6-2: Plans in Question-Answering

Of course, the plan synthesis process may fail: R may be unable to find a way for Q to execute either α or β (or both). On the analysis given in the preceding chapter, such a failure corresponds to R's judging the plan to include unexecutable acts; this judgement should serve as input to the response generation process. Otherwise, if R succeeds in finding ways for Q to execute α and/or β, the plans she synthesizes will be input to response generation.

Response generation, the last stage in the question-answering process, thus depends upon a number of different things. Its inputs may include a judgement of the query's coherence, an eplan inferred to underlie the query, a judgement as to whether that eplan is well-formed and, if not, what the source of the ill-formedness is, judgements as to whether the intended acts it contains are executable, and synthesized plans for executing some or all of those intended acts. As discussed in Chapter 5, factors like relevance and salience also affect response generation; these are shown as a black box input to response generation in Figure 6-1.

6.2. Representing Queries

The plan inference process as I describe it operates on formal representations of information that can be extracted from English queries. In terms of any computer implementation of this model, this means that I assume the existence of a syntactic parser, semantic analyzer, and basic pragmatic component to provide logical forms which can then be translated in a principled way into the input representation for the system.

For the purposes of model of plan inference as developed in this chapter and implemented in SPIRIT, I assume that the queries to be analyzed can be represented by sentences of the form:

$$\text{QUERY}(Q,\theta,\alpha,\beta,t_2,t_0)$$

This should be read as asserting that Q's query θ at time t_0 asks how to do α in order to do β at performance time t_2 (where the performance time may be highly underspecified). So, to translate a query to this form, it is necessary to decide what the queried act is, and what the goal act is.

Adopting this input representation relies on the assumption discussed in Section 2.4 that all queries include explicitly indicated goals. That is, I will assume for now that every query to be analyzed can be paraphrased as "I need to know (how to do) α so that I can do β." Recall that I argued that, even when a goal is explicitly indicated in a query, plan inference is essential to determine whether there are any invalidities in the underlying plan, and if so, what their source is. Since the primary goal of this thesis is to demonstrate how plan inference can be used to reason about invalid plans, the restriction to queries with explicitly indicated goals is not unjustified. In Chapter 8 I will briefly discuss how the plan inference process can deal with queries in which the goal is not explicitly indicated. I will also there consider what additional information may be conveyed in queries, and how such information should be handled in future extentions to the model of plan inference developed here.

A few examples taken verbatim from the mail transcripts will illustrate the mapping I am assuming from surface form to logical representation. The first example is familiar:

> (43) "Is there any way to tell a bboard to load only new messages (I would like to speed up entry into infomac)?"

From Example (43) should be extracted the information that the queried act is Q's telling a bboard to load only the new messages and that the goal act is his speeding up entry time into infomac.

In the next example, what needs to be extracted is

> (44) "How can I scroll up and down in mail? I usually forget what number I'm looking at and if the info I need is off the screen, I can't get to it easily."

that the queried act is Q's scrolling in mail, and the goal act is his finding out the current message number. Determining that this is the goal act relies on an analysis of a statement of the form "I can't do β"; "I can't α"; see [Wilensky 84, Sidner 83].[64]

A third example is similar:

[64] It may be that these analyses require theories of communicative plans and communicative-plan analysis, but I will not pursue this issue here.

(45) "Is there any way to append a mail message to the end of an existing file...? Sometimes I like to store away a series of messages in a single file, and it would be convenient to be able to do this from within mail. . ."

Here what needs to be extracted is that the queried act is Q's appending a mail message to the end of an existing file, and that the goal act is his storing away a series of messages in a single file from within mail.

"How" versus "What," "When," And "Where" Questions

The discussion above has been based upon queries that are roughly of the form "How can I do α? I want to do β." or "Is there some way to do α? I want to do β." A word is in order about the representation of queries of the form "What is the value of X? I want to do β," "Where is X? I want to do β," etc. So, for instance, consider example Example (46), taken from Horrigan's train station transcripts [Horrigan 77].

(46) "Going to Stratford, which gate would it be?"

One interesting fact about Example (46) is that it demonstrates that even in a very simple domain such as the train-station information booth there occur queries in which the goal act is explicitly mentioned. In this example, I claim that the information that needs to be extracted for input to plan inference is that the queried act is Q's finding out the gate number of the train to Stratford, while the goal act is his going to Stratford.[65]

My treatment of this example may seem inconsistent with my treatment of the "How" examples above. With questions of the form "What is the value of X/Where is X", I take the queried act to be finding out (the value of) X. In contrast, with questions of the form "How can I α?" I assume that α is part of the questioner's plan, and take his queried act to be α itself, rather than finding out how to α.

The lack of parallelism is actually just a short cut. We could well take the queried act in the case of "how"-questions to be finding out how to α. However, in each case, this act is intended to be performed in order to enable the performance of α itself, which in turn is performed as part of a plan to do β.

[65]Note that the plan underlying this query is not a simple plan.

Thus, I will consider there to be a theorem, as it were, relating every act of finding out how to α to an act of α. A similar theorem does not exist for questions of the form "What is the value of X?" because in general finding out the value of some single parameter can enable many different acts. Hence, the queried act in these cases must be taken to be finding out the value of X, and the decision as to what act this is meant to support, which in turn supports the goal act, must be left to plan inference process.

6.3. The Plan Inference Process

We are finally in a position to consider in detail the process of plan inference. Recall that so far I have claimed that the job of the inferring agent is to find an eplan that she believes underlies Q's query. In other words, we can model the plan inference process as an attempt to prove, given some query Θ, a theorem of the form:

$$\exists\, \alpha_1 \ldots \exists\, \alpha_{n-1} \exists\, \beta \exists\, \rho_1 \ldots \exists\, \rho_{n-1} [\text{BEL}(\text{R}, \text{UNDERLYING-EPLAN}(\Theta, \text{Q}, \beta, [\alpha, \alpha_1, \ldots, \alpha_{n-1}], [\rho_1, \ldots, \rho_{n-1}], t_2, t_1),\ t_1)]$$

Intuitively, when Q's query Θ asks how to do α in order to do β--i.e., when it can be represented as QUERY$(\text{Q},\Theta,\alpha,\beta,t_2,t_0)$--R will believe that some eplan that she can ascribe to Q underlies Θ if it links an intention to do α at t_2 with an intention to do β at t_2. As we have already seen, for R to believe that Q has some eplan is for her to have a particular configuration of beliefs about Q's beliefs and intentions. The question then is: how does R come to have these beliefs? How does she ascribe beliefs and intentions to Q?

Given a query Θ that can be encoded as QUERY$(\text{Q},\Theta,\alpha,\beta,t_2,t_0)$ there are two intentions that R can directly ascribe to Q: (1) the intention to do α (at t_2) and (2) the intention to do β (at t_2). R's job then is to infer further intentions that Q might have, as well as beliefs that support those further intentions.

To do this, R can reason in the following way. She believes that Q intends to do some act α. Let us further suppose that she has reason to believe that it is plausible that Q believes that act-type α conditionally generates some other act-type γ, under condition C, and that she has no reason to suppose that Q believes that C will not hold at the intended performance time of his plan. Then R can decide that it is plausible for Q to believe that by his act of α, he will do γ, and further, that it is plausible for him to intend to do α in order to do γ, and to intend to do γ. The plausibility of Q's having these intentions depends upon the plausibility of his having the aforementioned beliefs, so R will attribute to Q as a bundle the plausible intentions and supporting beliefs. For this reason, the plan inference rules will relate entire eplans to entire eplans. The process of belief and intention ascription can be iterated: believing that it is plausible that Q intends to do γ, R can then reason about what Q might believe he can do by this act. R can also reason about what Q might plausibly intend to do in order to do his goal act β, and can then iterate as well in the "backward" direction.

At this point, R is merely reasoning about clusters of beliefs and intentions that it is *plausible* for Q to have. Further, when she decides that it is plausible for Q to believe that by doing α he will do some γ, the plausibility of that belief must be established on the basis of something other than Q's query itself. For instance, while it is true that if Q says "I want to speed up entry into mail so I need to change my password," R can figure out that Q believes that by changing his password he will speed up entry into mail, this belief cannot be attributed to Q in the process described in the previous paragraph, unless R can also find an independent explanatory belief that supports it.

So in attempting to find a set of beliefs and intentions--an eplan--that underlies Q's query, R will determine that certain (explanatory) beliefs are plausible for Q to have, independent of the query itself. On the other hand, the fact that ascribing a certain belief to Q helps make sense of his query does increase the likelihood that Q holds the belief in question. That is, we can imagine that R is able to infer a number of beliefs that it is plausible for her to suppose Q has. Some of these beliefs will support further beliefs that particular actions generate other actions. If some subset of this latter group of beliefs can be used collectively to explain why Q might think that his queried act supports his goal act, then this fact makes it likely that Q does indeed have both this subset of beliefs and the explanatory beliefs that support it.

Thus the two processes--of identifying the beliefs that may explain some plan and of identifying the plan itself--are synergetic. In attempting to infer Q's plan, R attempts to identify plausible beliefs of Q's that may explain his plan; in identifying sufficiently many of these beliefs to account for the entire query, R adds weight to the likelihood that she has correctly identified the explanatory beliefs. As an illustration, suppose that Q says "I want to talk to Kathy, so I need to call the hospital." R believes that when an agent calls a location, he can thereby establish a communication channel to another agent who is at that location. This belief can be represented as a conditional generation between calling a location X and establishing a communication channel to Y under the condition that Y is at X. Now, for arbitrary proposition P, when R believes that P, she may also believe that it is plausible for Q to believe that P.[66] So in the current example, R has reason to believe that it is plausible that Q believes that the conditional generation relation just mentioned holds. Note that R's inference that this is a plausible belief of Q's does not depend upon his query, but rather upon a belief that R herself has about the domain. Ascribing to Q the plausible belief in question, however, can explain a further plausible belief of Q's, that by calling the hospital he will establish a communication channel to Kathy, provided Q further believes that Kathy is at the hospital. Then these two beliefs--that the aforementioned conditional generation relation holds and that Kathy is at the hospital--along with certain other beliefs explain Q's query, and this fact in itself adds weight to the likelihood that Q indeed has them.

Dropping the assumption that R has complete knowledge may help underscore the point. When Q asks his query, R may not be aware that Kathy is at the hospital. But because R can explain Q's query by attributing this belief to him, along with a belief about the relation between two domain actions, where the plausibility of the latter belief is independently motivated, R has good reason for deciding that Q believes that Kathy is at the hospital. In fact, if R believes that Q may know more about Kathy's whereabouts than R herself does, she may accept this belief and may exclaim "What happened to Kathy?"

It is enlightening to recast Allen's model of plan inference in terms of the account I have been describing: one that is essentially driven by a mental phenomenon view of plans.[67] Allen's *partial plans*, whose construction depends upon the inferring agent's domain knowledge, can be seen to represent intentions that the inferring agent believes the actor plausibly has. When a partial plan includes an arc from α to β, we can view this as R's believing that it is plausible

[66]This last step is nonmonotonic; an agent is only entitled to believe that it is plausible for another agent to believe some P if she has no reason to believe that he believes $\neg$ P. This will be discussed further below.

[67]The points made here about Allen's model in particular also apply to the other plan inference models discussed in Chapter 2.

that Q intends to do α in order to do β, and, given the assumption of a strong belief requirement on intending, R's further believing that it is plausible that Q believes that by doing α he will do β. This latter belief corresponds to one that would satisfy Clause (ii) of the definition of eplans in Definition (P2). But what about an explanatory belief, that would satisfy Clause (v)? We can interpret the operator that is used in the construction of the arc from α to β as representing a belief about the way in which acts of type α are in general related to acts of type β, and can then view such a belief as explaining the prior belief, perhaps in conjunction with some additional beliefs about what will be true at the performance time of the plan. Of course, when invalid plans are not a concern, there is no need to keep track of such explanatory beliefs; it is a contribution of this thesis to develop an account of how they can be inferred when they differ from the beliefs of the inferring agent, and of how they influence cooperative response generation.

Given this reinterpretation of Allen's model, we can see that it embodies a synergy between the process of constructing plans that potentially underlie a query and that of accepting one of those as the plan that the agent has in mind, quite similar to the synergy I discussed earlier. In Allen's model, when a partial plan, which typically contains two parts, can be unified into a single plan linking the observed act to an expected goal, its "rating" is increased, reflecting the idea that by the very fact that the plan in question accounts for the observed act, it is likely actually to be the agent's plan. But where Allen encodes the synergetic effects of belief (and intention) ascription and plan inference on R's epistemic state only implicitly, in the control heuristics and the task structure, I have attempted to state these effects explicitly, in the axioms describing the inference process.

We thus have the following picture. R is asked some question, Θ, with a queried act of α and goal act of β. From this she can infer that Q intends to do α and to do β. In fact, she can infer even more: believing that Q intends to do β, she can also infer that Q has an eplan that consists merely in an intention to do β, i.e.

$$BEL(R, EPLAN(Q,\beta,[],[],t_2,t_1),t_1)$$

To see why this is justified, consider again the definition of eplans in Definition (P2). Clause (iii) is directly satisfied by R's belief that Q intends to do β; Clause (i) is satisfied as a result of the assumption that an intention to do α entails a belief that the agent will do α. Since n=1 in this eplan--there are no supporting acts--Clauses (ii), (iv), and (v) of Definition (P2) are vacuously satisfied. Analogously, from R's belief that Q intends his queried act α, R can infer that Q has an eplan that consists merely in an intention to do α. We can encode these two inferences in Axiom (Q1):

Axiom
(Q1) $BEL(R,QUERY(Q,\theta,\alpha,\beta,t_2,t_0),t_1)$

$\rightarrow$

$BEL(R,EPLAN(Q,\alpha,[],[],t_2,t_1),t_1) \wedge$

$BEL(R,EPLAN(Q,\beta,[],[],t_2,t_1),t_1)$[68]

Of course, if Q's query θ includes different information, and cannot be represented as $QUERY(Q,\theta,\alpha,\beta,t_2,t_0)$, a different rule will be needed to represent the eplans that R can infer from θ. The theory as developed leaves room for axioms that supplement Axiom (Q1) so that it can to handle other sorts of queries, but I will not present any such axioms here.

Having inferred a pair of eplans directly from θ, as modeled by Axiom (Q1), R can next use plan inference rules to reason from plausible sets of beliefs and intentions that Q has--plausible eplans--to further plausible eplans. To start the process, R must reason that the beliefs and intentions she ascribes to Q directly from θ are beliefs and intentions that it is plausible for Q to have, since the plan inference rules map from plausible beliefs and intentions to plausible beliefs and intentions, i.e., from plausible eplans to plausible eplans.

To model one agent's belief that it is plausible that another agent has certain beliefs or intentions, I will introduce two new operators, BEL' and INT', which represent plausible belief and plausible intention, respectively. All axioms that allow plausible belief or plausible intention to be deduced will only permit that deduction inside the scope of a BEL operator. The first two such axioms, (Q2) and (Q3), relate an agent R's belief that another Q believes (intends) something to R's belief that it is plausible for Q to believe (intend) the same thing.

Axioms
(Q2) $BEL(R,BEL(Q,p,t_2),t_1) \rightarrow BEL(R,BEL'(Q,p,t_2),t_1)$

[68] Although the query is asked at t_0, the assumption is made that Q's intentions do not change between then and t_1, the interval during which R infers Q's plan.

(Q3) $\mathrm{BEL}(R,\mathrm{INT}(Q,\alpha,t_2,t_0),t_1) \rightarrow \mathrm{BEL}(R,\mathrm{INT'}(Q,\alpha,t_2,t_0),t_1)$

Then, since believing that another agent has an eplan means believing that he has some configuration of beliefs and intentions, we can say that R believes that Q has a plausible eplan (or EPLAN') precisely when R believes that Q plausibly has the necessary configuration of beliefs and intentions.

Definition

(P3) $\mathrm{BEL}(R, \mathrm{EPLAN'}(Q,\alpha_n,[\alpha_1, \ldots,\alpha_{n-1}], [\rho_1, \ldots,\rho_{n-1}],t_2,t_1),t_1) \equiv$

(i) $\mathrm{BEL}(R,\mathrm{BEL'}(Q,\mathrm{EXEC}(\alpha_i,Q,t_2),t_1), t_1)$ for $i=1,\ldots,n-1 \wedge$

(ii) $\mathrm{BEL}(R,\mathrm{BEL'}(Q,\mathrm{GEN}(\alpha_i,\alpha_{i+1},Q,t_2),t_1),t_1)$ for $i=1,\ldots,n-1 \wedge$

(iii) $\mathrm{BEL}(R,\mathrm{INT'}(Q,\alpha_i,t_2,t_1),t_1)$ for $i=1,\ldots,n-1 \wedge$

(iv) $\mathrm{BEL}(R,\mathrm{INT'}(Q,\mathrm{by}(\alpha_i,\alpha_{i+1}),t_2,t_1),t_1)$ for $i=1,\ldots,n-1 \wedge$

(v) $\mathrm{BEL}(R, \mathrm{BEL'}(Q, \rho_i, t_1), t_1)$ for $i=1,\ldots,n-1$

where each ρ_i is $\mathrm{CGEN}(\alpha_i,\alpha_{i+1},C_i) \wedge \mathrm{HOLDS}(C_i,t_2)$

It is straightforward to show that believing that an agent has an eplan implies believing it plausible that he has the same eplan:

(Q4) $\mathrm{BEL}(R,\mathrm{EPLAN}(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho,\ldots,\rho_{n-1}],t_2,t_1),t_1)$

$\rightarrow$

$\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho,\ldots,\rho_{n-1}],t_2,t_1),t_1)$

Having inferred two eplans from Q's query θ, as expressed in the consequent of Axiom (Q1), R can thus reason that those are also eplans it is plausible for Q to have. She can then use plan inference rules to reason to further plausible eplans. When a plausible eplan is finally found that is "as large as required" by the query--i.e., that accounts for the query by linking the intention to do the queried act to the intention to do the goal act--R can "upgrade" the epistemic status of this plausible eplan, no longer considering it as a set of beliefs and intentions that Q plausibly has, but rather as a set of beliefs and intentions that, for the purposes of formulating a response, R can assume Q does have. This "upgrading" is encoded in the following axiom:

Axiom

(Q5) $\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\beta,[\alpha,\ldots,\alpha_{n-1}],[\rho_1,\ldots\rho_{n-1}],t_2,t_1),t_1) \wedge$

$\mathrm{BEL}(R, \mathrm{QUERY}(Q,\theta,\alpha,\beta,t_2,t_0), t_1)$

$\rightarrow$

$\mathrm{BEL}(R,\mathrm{UNDERLYING\text{-}EPLAN}(\theta,Q,\beta,$

$$[\alpha,\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1)$$

Axiom (Q5) reflects the close relation between belief ascription and plan inference. If a set of plausible beliefs and intentions can explain Q's query, in the sense of composing a plan that could underlie it, then the inferring agent is entitled to treat those as beliefs and intentions that Q does have, and to formulate her response on that basis.

Again, the theory leaves room for handling queries that cannot be represented as QUERY($Q,\theta,\alpha,\beta,t_2,t_0$). Axioms to supplement (Q5) can be introduced to define the conditions under which an eplan can count as underlying a query which has an alternative form.

The process of inferring an eplan is, in this model, the process of proving a theorem about the existence of beliefs and intentions that satisfy the consequent of Axiom (Q5) (and, in turn, Definition (P2)). It may be useful to consider the way in which the various axioms I have been presenting contribute to this inference. In fact, it may well be that efficient implementation of this model should depend not just on an unconstrained deduction effort, but rather on an effort that exploits the fact that the various axioms are designed to be used in certain constrained ways; i.e. they model certain ordered parts of the inference process. This is how SPIRIT, the system that implements this model, is designed. Figure 6-3 below diagrams the stages of the process of finding an eplan that underlies a query. Each stage of the process has now been described except for the actual application of plan inference rules. It is to that that we now turn.

6.4. Plan Inference Rules

Recall from the previous section how the plan inference rules are meant to work. They encode the principles by which an inferring agent R can reason from some set of beliefs and intentions--call this the antecedent eplan--that she thinks the actor Q plausibly has, to some further set of beliefs and intentions--call this the consequent eplan--that she also thinks it is plausible for him to have. The beliefs and intentions that the antecedent eplan comprises are a proper subset of those that the consequent eplan comprises. In order to reason from antecedent eplan to consequent eplan, R must, I claimed, attribute some explanatory belief to Q on the basis of something other than just Q's query. In more detail, if part of R's belief that Q plausibly has some eplan is a belief that Q plausibly intends to do some α, and R has reason to believe that Q plausibly believes that act-type

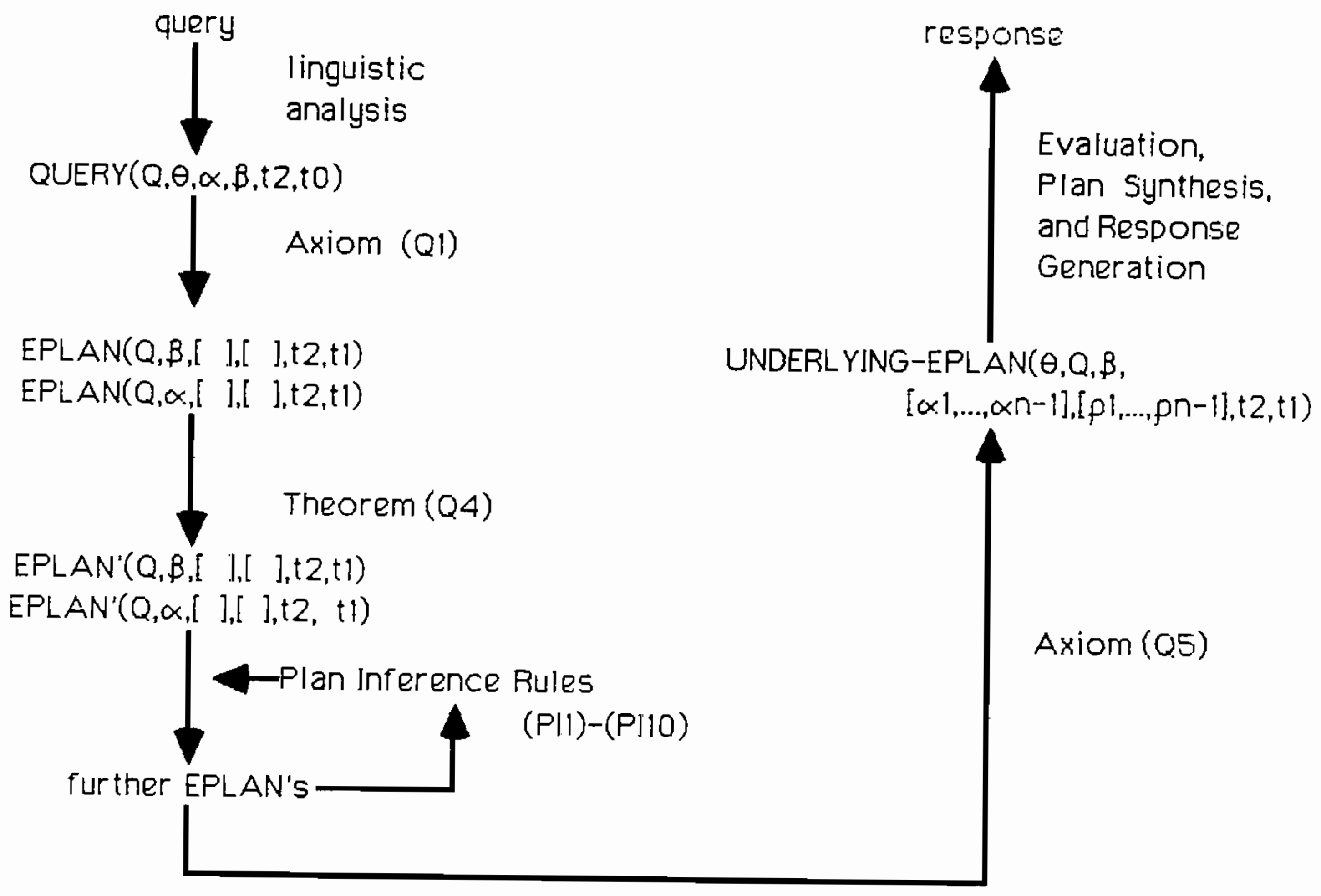


Figure 6-3: The Process of Inferring an Eplan

α conditionally generates act-type γ under condition C, then R can infer that Q plausibly intends to do α in order to do γ, believing that C will hold at performance time. R can as well reason in the other direction: if part of her belief that Q has some plausible eplan is a belief that Q plausibly intends to do some α and R has reason to believe that Q plausibly believes that act-type γ conditionally generates act-type α under condition C, then R can infer that Q plausibly intends to do γ in order to do α, believing that C will hold at performance time.

The plan inference rules encode the pattern of reasoning expressed in the last two sentences. Different plan inference rules encode the different bases upon which R may believe that it is plausible for Q to believe that a conditional generation relation holds between some α, an act of which is plausibly intended as part of the antecedent eplan, and some γ. This ascription of plausible beliefs, as well as the ascription of plausible intentions, is a nonmonotonic process. For an arbitrary proposition P, R will only decide that it is plausible for Q to believe that P if R has no reason to believe Q believes that $\neg$ P. Hence, in the plan inference rules presented below, the connective " $\rightarrow$ " should not be taken to represent strict implication; it is like Allen's " $\rightarrow_i$" and " $\rightarrow_c$" connectives. Note also that plausible belief and plausible intention are not deductively closed: R may well be able to infer that Q plausibly believes P and plausibly believes $\neg$ P; what she cannot infer is that Q plausibly believes P and $\neg$ P.

In the most straightforward case, R will ascribe to Q a belief that she herself has. This reasoning can be encoded in the representation language as follows:

Axiom

(PI1) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, CGEN(\alpha_n,\gamma,C), t_1)$
$\rightarrow$
$BEL(R,EPLAN'(Q,\gamma,[\alpha_1,...,\alpha_n],[\rho_1,...,\rho_n],t_2,t_1),t_1)$
where $\rho_n = CGEN(\alpha_n,\gamma,C) \wedge HOLDS(C,t_2)$

This rule says that if R's belief that Q has some plausible eplan includes a belief that Q plausibly intends to do an act of α_n, and R also believes that act-type α_n conditionally generates some γ under condition C, then R can (nonmonotonically) infer that Q plausibly has the additional intention of doing α_n in order to do γ. The plausibility of Q's having this intention depends upon his also having the supporting belief that α_n conditionally generates γ under some condition C, which (Q believes) will hold at performance time.

A rule symmetric to Axiom (PI1) is also needed since R not only can reason about what acts might be generated by an act that she already believes Q plausibly intends, but also about what acts might generate such an act. Axiom (PI2) is the symmetric rule:

Axiom

(PI2) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, CGEN(\gamma,\alpha_1,C), t_1)$
$\rightarrow$
$BEL(R,EPLAN'(Q,\alpha_n,[\gamma,\alpha_1,\ldots,\alpha_{n-1}],[\rho_0,\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1)$
where $\rho_0 = CGEN(\gamma,\alpha_1,C) \wedge HOLDS(C,t_2)$

Axiom (PI2) says that if R's belief that Q has some plausible eplan includes a belief that Q plausibly intends to do do an act of α_1, and R also believes that some γ conditionally generates act-type α_1 under condition C, then R can infer that Q plausibly intends to do an act of γ in order to do α_1, and plausibly believes that the supporting conditional generation relation holds and that its generation-enabling condition C will hold at the intended performance time.

Axioms (PI1) and (PI2) are quite similar to plan inference rules proposed by Allen. (PI1) corresponds to his Body-Action rule, and (PI2) to his Action-Body rule. One way to view Axioms (PI1) and (PI2) is as explications of Allen's two rules in terms of the mental phenomenon view of plans. However, in this explication, Axioms (PI1) and (PI2) enable the modeling of behavior that could not be modeled by Allen's rules. To see this, consider a situation in which R has significantly more domain knowledge than does Q, as might well be the case when Q is consulting R for information. In particular, R may have many more beliefs than does Q about the relations between act-types in the domain and, thus, may be able to infer a plan linking Q's queried act α to his goal act β that is at a level of detail that Q is unlikely to have had in mind. (Imagine that R is an electrical engineer who understands a great deal about the flow of electricity, and Q is a layman who simply wants to know how to turn on the light.) Although Axioms (PI1) and (PI2) model the situation in which R atttributes to Q domain beliefs that she herself has, their inclusion in the model does not mean that R will attribute to Q all the beliefs she has. In particular, she will not attribute to him any beliefs she thinks it is unlikely for him to have: so, for instance, if she realizes that he is less knowledgeable than she, she will refrain from attributing to him expertise about domain acts. An interesting result of this is that R may be able to inform Q that his plan will work, but not for the reasons he thinks. This could occur when R judges to be ill-formed the eplan that she infers to underly Q's query--a plan which may only include acts at a high level of detail--but when she can also find a different plan, which she judges to be valid, relating α and β--this being a plan that she does not attribute to Q because she does not think it is plausible that he has the component beliefs. Of course, in this analysis, R may also be mistaken in what she thinks Q plausibly believes, and may, for example, attribute to him a more fine-grained plan than he had in mind, and then criticize that plan on grounds that he had never considered. Since such mistaken inferences do sometimes occur in human behavior, this result should be seen as a feature of the current model.

We can now return to the specifics of Axioms (PI1) and (PI2). One important thing to observe about them is that, while in using them R draws only on beliefs that she herself has about the ways in which actions in the domain are related to one another, their use can nonetheless result in the inference of invalid plans. Although both rules require that R believe that Q plausibly believes an explanatory statement of the form

$$\mathrm{CGEN}(\alpha,\beta,\mathrm{C}) \wedge \mathrm{HOLDS}(\mathrm{C},t_2)$$

they do not require either that R believe that C holds at t_2, or that Q's doing α at t_2 is executable.

Q's question about calling Kathy at the hospital provides an example of the first case, in which Axiom (PI1) is used to infer a piece of an eplan that R judges invalid--more specifically, ill-formed--because she does not believe that the generation-enabling condition C in the inferred explanatory belief will hold at the intended performance time. Assume that sentence (i) below, which expresses R's belief about a plausible eplan of Q's, is true:

(i) $\mathrm{BEL}(\mathrm{R},\mathrm{EPLAN'}(\mathrm{Q},\mathrm{call(hospital)},[],[],t_2,t_1),t_1)$

Assume also that sentence (ii), expressing R's belief about a pair of domain act-types, is true:

(ii) $\mathrm{BEL}(\mathrm{R},\mathrm{CGEN}(\mathrm{call(X)},\mathrm{establish\text{-}channel(Y)},\mathrm{at(X,Y)}),t_1)$

So R believes that Q plausibly has an eplan that includes an intention to call the hospital, and R also believes that calling location X conditionally generates establishing a communication channel to agent Y under the condition that Y is at X. We want to apply Axiom (PI1) to derive a further belief of R's, namely that Q has a plausible eplan that includes an intention to call the hospital in order to establish a channel to some agent Y who is at the hospital. Applying Axiom (PI1) requires determining that the term call(hospital) in (i) can denote the same act-type as the term call(X) in (ii), so a unification mechanism is needed.[69] After unification, Axiom (PI1) applies and the following can be derived:

(iii) BEL(R,EPLAN'(Q, establish-channel(Y),
[call(hospital)],
[CGEN(call(hospital),establish-channel(Y),at(hospital,Y))
$\wedge$ HOLDS(at(hospital,Y),t_2)],
t_2,t_1),t_1)

Then by a further unification of the terms establish-channel(Y) and establish-channel(Kathy), (iv) is true:

(iv) BEL(R,EPLAN'(Q, establish-channel(Kathy),
[call(hospital)],
[CGEN(call(hospital),establish-channel(Kathy),
at(hospital,Kathy))
$\wedge$ HOLDS(at(hospital,Kathy),t_2)],
t_2,t_1),t_1)

Figure 6-4 diagrams the plausible eplan in (iv). As in Chapter 4, the nodes represent the type of the intended acts in the eplan. I have also labeled each arc with the inferred belief that is meant to explain the belief that the acts joined by that arc form a generational pair.

Although, given (iv), R believes that Q plausibly believes that Kathy will be at the hospital at t_2, R might not have that belief herself:

(v) BEL(R, HOLDS($\neg$ at(hospital,Kathy),t_2),t_1)

In the figure, I have italicized the part of the explanatory belief that R thinks is false. In this situation, R will judge the inferred portion of the eplan to be ill-formed. This particular type of ill-formedness, which consists in the generation-enabling condition of a valid conditional generation relation not holding at performance time, is the one type of invalidity that can, in principle, be handled by the standard models. As was mentioned in Chapter 1, however, dealing even with this kind of invalidity has not been a focus of existing systems. Further, without an analysis of the general problem of invalid plans and their effects on cooperative responses, the standard models would need to treat this as a special case of question-answering in which a variant response is required.

[69] To be strictly correct, the unification process used here should be somewhat more powerful than standard unification; see Litman [Litman 85,pp. 120-122].

```
establish-connection(Kathy)
∧
|
|         CGEN(call(hospital),establish-channel(Kathy),
|              at(hospital,Kathy)) ∧
|         HOLDS(at(hospital,Kathy),t2)
|
call(hospital)
```

Figure 6-4: Inferred Eplan for Q to Establish A Channel to Kathy

Let us now consider a second example. Recall the query that asked how to set the permissions on a file to faculty-read only, in order to prevent Tom from reading that file. We can represent this query as[70]

$$QUERY(Q,\Theta,\text{set-permissions}(file1,read,faculty),prevent(file1,read,Tom),t_2,t_0)$$

Let us suppose that (vi) and (vii) express true statements about R's domain beliefs:

(vi) $BEL(R,CGEN(\text{set-permissions}(X,P,Y), prevent(X,P,Z), \neg\, member(Z,Y) \wedge \neg\, \text{system-mgr}(Z)),t_1)$

(vii) $BEL(R,HOLDS(\text{system-mgr}(Tom),t_2),t_1)$

So once again, R believes that setting the permissions on file X to allow only group Y to do P (e.g., read or write) will prevent agent Z from doing P to Y provided Z is neither a member of group Y nor the system manager: this is what (vi) asserts. Sentence (vii) asserts that R believes that Tom is the system manager.

Now let us consider the steps in R's inference of an eplan that underlies Θ. First, from Axiom (Q1), we can derive (viii):

[70]Strictly speaking, the goal act-type as represented here should be taken to be something like "causing Tom to be prevented from having access to file1," for it is the occurrence of an act of this type that is intended during time interval t_2, the same time interval during which the act of setting the permissions is intended. The act of preventing Tom from accessing the file presumably has a much longer intended duration time.

(viii) BEL(R,EPLAN(Q,set-permissions(file1,read,faculty),[],[],t_2,t_1),t_1) ∧
BEL(R,EPLAN(Q,prevent(file1,read,Tom),[],[],t_2,t_1),t_1)

By the predicate calculus axiom $(p \wedge q) \rightarrow p$, we can derive (ix) from (viii):

(ix) BEL(R,EPLAN(Q,set-permissions(file1,read,faculty),[],[],t_2,t_1),t_1)

and by Theorem (Q4) we can then derive

(x) BEL(R,EPLAN'(Q,set-permissions(file1,read,faculty),[],[],t_2,t_1),t_1)

So far, we have simply shown that R can infer from Q's query that he has a plausible eplan that consists only in an intention to set the permissions of file1 to faculty-read only. Analogously (by application of $(p \wedge q) \rightarrow q$ to (viii), followed by Theorem (Q4)) we can derive (xi):

(xi) BEL(R,EPLAN'(Q,prevent(file1,read,Tom),[],[],t_2,t_1),t_1)

That is, R can also infer from Q's query that he has a plausible eplan that consists only in the intention to prevent Tom from reading file1.

Next, since (x) and (vi), after unification, satisfy the antecedents of Axiom (PI1), we can derive (xii):

```
(xii) BEL(R,EPLAN'(Q,prevent(file1,read,Z),
                    [set-permissions(file1,read,faculty)],
                    [CGEN(set-permissions(file1,read,faculty),
                          prevent(file1,read,Z),
                          ¬ member(Z,faculty) ∧ ¬ system-mgr(Z)) ∧
                    HOLDS(¬ member(Z,faculty) ∧ ¬ system-mgr(Z), t2)],
        t2,t1),t1).
```

That is, R can determine that Q plausibly has the same belief as she does about the relation between setting permissions on a file and preventing someone accessing to that file, as expressed in (vi). Therefore R can determine that Q plausibly intends to set the permissions on file1 to faculty-read only in order to prevent some agent Z from reading file1.

Next consider the expansions of (xi) and (xii) by Definition (P3). Sentence (xiii) expresses the beliefs of R's that are entailed by (xi), and sentence (xiv), those entailed by (xii). Each conjunct in (xiii) and (xiv) is labeled with a number denoting the clause of Definition (P3) to which it corresponds.[71]

(xiii)
1a. $BEL(R, BEL'(Q,EXEC(prevent(file1,read,Tom),Q,t_2),t_1), t_1) \wedge$
3a. $BEL(R, INT'(Q,prevent(file1,read,Tom),t_2,t_1), t_1)$

(xiv)
1a. $BEL(R, BEL'(Q,EXEC(prevent(file1,read,Z),Q,t_2),t_1),t_1) \wedge$
1b. $BEL(R, BEL'(Q,EXEC(set\text{-}permissions(file1,read,faculty),Q,t_2),t_1),t_1) \wedge$
2a. $BEL(R, BEL'(Q, GEN(set\text{-}permissions(file1,read,faculty),$
$prevent(file1,read,Z),t_2), t_1), t_1) \wedge$
3a. $BEL(R, INT'(Q, prevent(file1,read,Z),t_2,t_1), t_1) \wedge$
3b. $BEL(R, INT'(Q,set\text{-}permissions(file1,read,faculty),t_2,t_1), t_1) \wedge$
4a. $BEL(R, INT'(Q, by(set\text{-}permissions(file1,read,faculty),$
$prevent(file1,read,Z)),t_2,t_1), t_1) \wedge$
5a. $BEL(R, BEL'(Q, CGEN(set\text{-}permissions(file1,read,faculty),$
$prevent(file1,read,Z),$
$\neg member(Z,faculty) \wedge \neg system\text{-}mgr(Z)) \wedge$
$HOLDS(\neg member(Tom,faculty) \wedge \neg system\text{-}mgr(Tom), t_2),$
$t_1), t_1)$

After unification, the result is a set of beliefs that R has about Q's plausible beliefs and intentions sufficient for claiming that R has the plausible eplan expressed in (xv), and diagrammed in Figure 6-5.

(xv) $BEL(R, EPLAN'(Q, prevent(file1,read,Tom),$
$[set\text{-}permissions(file1,read,faculty)],$
$[CGEN(set\text{-}permissions(file1,read,faculty),$
$prevent(file1,read,Tom),$
$\neg member(Tom,faculty) \wedge \neg system\text{-}mgr(Tom)) \wedge$
$HOLDS(\neg member(Tom,faculty) \wedge \neg system\text{-}mgr(Tom), t_2)],$
$t_2, t_1), t_1)$

[71] Actually, neither clauses of (xiii) nor Clauses 1b or 3b of (xiv) correspond directly to clauses of Definition (P3). This is because Definition (P3), like Definition (P2), only requires a belief of executability about, and an intention to do, the constituent acts. But see Section 4.6 for a proof that the goal acts are subject to the same requirements.

```
prevent(file1,read,Tom)
∧
|
|         CGEN(set-permissions(file1,read,faculty),
|              prevent(file1,read,Tom),
|              ¬ (member(Tom,faculty) ∧  ¬ system-mgr(Tom))
|         HOLDS(¬ (member(Tom,faculty) ∧  ¬ system-mgr(Tom), t2)
|
set-permissions(file1,read,faculty)
```

Figure 6-5: Inferred Eplan for Q to Prevent Tom from Reading a File

Finally, by Axiom (Q5), the plausible eplan in (xv) can be upgraded to an eplan that R believes underlies Q's query. Since, as expressed in (vii), R does not believe that Tom is not the system manager, she will judge this plan to be ill-formed. Note that she may also judge the acts it contains to be unexecutable.

Axioms (PI1) and (PI2) are extremely basic plan inference rules: using them R will infer eplans that include beliefs and intentions explained by conditional generation relations that R herself believes true. Despite their simplicity, Axioms (PI1) and (PI2) can be used to account for a number of the observed examples of invalid plans. The plan inference model nonetheless can be enhanced by adding certain other axioms for the inference of eplans, axioms that allow the introduction of novel conditional generation relations in explanatory beliefs. I will discuss two types of these. The first involves a fairly straightforward variation of R's own beliefs, while the second involves other, less well-defined forms of reasoning.

Variations of R's Beliefs

In using only Axiom (PI1) and (PI2), R commits implicitly to the assumption that Q has the same beliefs about conditional generation as she does, although, as just discussed, this does not commit her to believing that Q's plan is valid. There is, however, the possibility that Q has different beliefs than R about the ways in which the actions in some domain are related to one another. One strategy for R to take in discovering such beliefs is to consider that Q's beliefs may be slight variations of her own. For instance, Q may believe that some act-type α conditionally generates another act-type β under conditions C that are weaker than the conditions R believes necessary. So, for example, a conditional generation relation that Q believes valid may be missing a conjunct of its enabling condition, in comparison to the relation that R believes valid. Axiom (PI3) and the symmetric Axiom (PI4) encode this possibility:

$$\begin{array}{ll}
\text{(PI3)} & \text{BEL}(R,\text{EPLAN'}(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),\ t_1) \wedge \\
& \text{BEL}(R,\text{CGEN}(\alpha_n,\gamma,C_1 \wedge \ldots \wedge C_m),\ t_1) \\
& \rightarrow \\
& \text{BEL}(R,\text{EPLAN'}(Q,\gamma,[\alpha_1,\ldots,\alpha_n],[\rho_1,\ldots,\rho_n],t_2,t_1),\ t_1) \\
& \text{where } \rho_n = \text{CGEN}(\alpha_n,\gamma,C_1 \wedge \ldots \wedge C_{i-1} \wedge C_{i+1} \wedge \ldots \wedge C_m) \wedge \\
& \quad \text{HOLDS}(C_1 \wedge \ldots \wedge C_{i-1} \wedge C_{i+1} \wedge \ldots \wedge C_m,\ t_2)
\end{array}$$

$$\begin{array}{ll}
\text{(PI4)} & \text{BEL}(R,\text{EPLAN'}(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),\ t_1) \wedge \\
& \text{BEL}(R,\text{CGEN}(\gamma,\alpha_1,C_1 \wedge \ldots \wedge C_m),\ t_1) \\
& \rightarrow \\
& \text{BEL}(R,\text{EPLAN'}(Q,\alpha_n,[\gamma,\alpha_1,\ldots,\alpha_{n-1}],[\rho_0,\rho_1,\ldots,\rho_n],t_2,t_1),\ t_1) \\
& \text{where } \rho_0 = \text{CGEN}(\gamma,\alpha_1,C_1 \wedge \ldots \wedge C_{i-1} \wedge C_{i+1} \wedge \ldots \wedge C_m) \wedge \\
& \quad \text{HOLDS}(C_1 \wedge \ldots \wedge C_{i-1} \wedge C_{i+1} \wedge \ldots \wedge C_m,\ t_2)
\end{array}$$

Axioms (PI3) and (PI4) thus encode the reasoning by which R ascribes to Q a plausible belief that some CGEN relation holds that is similar to one which R believes true, but which is missing a conjunct in the enabling condition--the C_i. These axioms can be generalized to allow several conjuncts to be dropped. Note that Axioms (PI3) and (PI4) are domain independent: they encode structural transformations of R's own beliefs that she can attempt to attribute to Q. Of course, within any domain, there may cases of specific variations that R should consider Q likely to believe--so, for instance,it might be that most users fail to realize that file permissions do not apply to the department chairperson--but these more specific variations should be enoced with Axioms (PI9) and (PI10), introduced below.

An example may help both to clarify the meaning of Axioms (PI3) and (PI4) and to show how they differ from Axioms (PI1) and (PI2). Recall again the detailed example given above, in which R infers an eplan underlying the question about how to set the protections on a file to faculty-read only. Assume again that (vi) and (vii) are true sentences expressing some domain beliefs that R has. Assume further that the reasoning encoded in (viii)-(ix) proceeds exactly as it did in the previous example. But now assume that R believes that Q knows that

Tom is the system manager. In that case, Axiom (PI1) will not apply, and R will not attribute to Q the explanatory beliefs embedded in (xii). Instead, she might reason that Q does not know that setting the permissions on a file does not affect the system manager. This reasoning is captured by an application of Axiom (PI3). The consequent eplan then includes (xvi) as one of the explanatory beliefs:

(xvi)BEL(R,BEL'(Q,CGEN(set-permissions(file1,read,faculty),
prevent(file1,read,Z),
$\neg$ member(Z,faculty)) $\wedge$
HOLDS($\neg$ member(Z,faculty), t_2), t_1), t_1)

Note that the conditional generation in (xvi) is a variation of what R believes, as expressed in (vi), in that it omits one of the generation-enabling conditions.

R can then continue to reason in exactly the same way as described in the preceding section, to the conclusion encoded in (xvii):

(xvii)BEL(R,UNDERLYING-EPLAN(Θ,Q,prevent(file1,read,Tom),
[set-permissions(file1,read,faculty)],
[CGEN(set-permissions(file1,read,faculty),prevent(file1,read,Tom),
$\neg$ member(Tom,faculty))
HOLDS($\neg$ member(Tom,faculty), t_2)],
t_2, t_1), t_1)

This eplan is shown in Figure 6-6. It may result in a response such as Example (40), in which R informs Q that setting the permissions on a file does not affect the system manager.

```
prevent(file1,read,Tom)
^
|
|          CGEN(set-permissions(file1,read,faculty),prevent(file1,read,Tom),
                 ¬ (member(Tom,faculty))
|          HOLDS(¬ (member(Tom,faculty), t2)
|
set-permissions(file1,read,faculty)
```

Figure 6-6: Alternate Eplan for Q to Prevent Tom from Reading a File

Note that in performing this inference, R makes use of the act-type constructor functions *set-permissions* and *prevent*. The former is a function from triples that consist of a file, a permission, and an agent, to an act-type, while the latter is a function from triples that consist of a file, an access, and an agent, also to an act-type. Within the representation language, we were able to express beliefs that R had about act-types that are in the range of these functions. It was then possible to show how R could reason about certain of these act-types even if they were not executable, as, for instance, setting the permissions on a file to faculty-read only. This may represent an act-type that Q has, but that R does not explicitly have until after she hears Q's query. Nonetheless, she can reason about it and infer Q's plan. When I discuss SPIRIT, in the next chapter, I will show the use of act-type constructor functions that are even more general than these: they encode doing an act to fewer objects, and speeding up a process.

Other Inference Rules

The final set of inference rules that I will describe rely upon principles of reasoning that are outside the scope of this thesis. I introduce them here as "hooks" available for expansions of this model.

First, there are rules that allow R to reason about confusions that may result because of similarities between two domain act-types. If R believes that two act-types α and β are quite similar, she thereby has reason to believe that it is plausible that Q has confused them, or has made a bad analogy from one to the other. Such reasoning is encoded in Axiom (PI5):

Axiom

(PI5) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, SIMILAR(\alpha_n,\delta),t_1) \wedge$
$BEL(R, CGEN(\delta,\gamma,C), t_1)$
$\rightarrow$
$BEL(R,EPLAN'(Q,\gamma,[\alpha_1,\ldots,\alpha_n],[\rho_1,\ldots,\rho_n],t_2,t_1),t_1)$
where ρ_n=$CGEN(\alpha_n,\gamma,C) \wedge HOLDS(C,t_2)$

Axiom (PI6) is the symmetric partner of Axiom (PI5):

Axiom

(PI6) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, SIMILAR(\alpha_1,\delta),t_1) \wedge$

$$\begin{array}{l}
\mathrm{BEL}(R,\ \mathrm{CGEN}(\gamma,\delta,C),\ t_1) \\
\rightarrow \\
\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\alpha_n,[\gamma,\alpha_1,\ldots,\alpha_{n-1}], \\
\qquad\qquad [\rho_0,\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1) \\
\text{where } \rho_0=\mathrm{CGEN}(\gamma,\alpha_1,C) \wedge \mathrm{HOLDS}(C,t_2)
\end{array}$$

These rules encode the situation in which R believes that Q may have confused some action α in the antecedent eplan with some other domain action δ. Alternatively, R may believe that Q has confused some action δ that is related by generation to an action α in the antecedent eplan with a similar action γ. Such reasoning is encoded in Axioms (PI7) and (PI8):

<u>Axioms</u>

(PI7)
$$\begin{array}{l}
\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],\ [\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1) \wedge \\
\mathrm{BEL}(R,\mathrm{CGEN}(\alpha_n,\delta,C),\ t_1) \wedge \\
\mathrm{BEL}(R,\ \mathrm{SIMILAR}(\delta,\ \gamma),t_1) \\
\rightarrow \\
\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\gamma,[\alpha_1,\ldots,\alpha_n],\ [\rho_1,\ldots,\rho_n],t_2,t_1),t_1) \\
\text{where } \rho_n=\mathrm{CGEN}(\alpha_n,\gamma,C) \wedge \mathrm{HOLDS}(C,t_2)
\end{array}$$

(PI8)
$$\begin{array}{l}
\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1) \wedge \\
\mathrm{BEL}(R,\ \mathrm{CGEN}(\delta,\gamma,C),\ t_1) \wedge \\
\mathrm{BEL}(R,\ \mathrm{SIMILAR}(\delta,\gamma),\ t_1) \\
\rightarrow \\
\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\alpha_n,[\gamma,\alpha_1,\ldots,\alpha_{n-1}],[\rho_0,\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1) \\
\text{where } \rho_0=\mathrm{CGEN}(\gamma,\alpha_1,C) \wedge \mathrm{HOLDS}(C,t_2)
\end{array}$$

The final set of inference rules that I will describe are useful for encoding specific misconceptions. Adopting them amounts to abandoning the Principle of Parsimony.

<u>Axiom</u>

(PI9)
$$\begin{array}{l}
\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1) \wedge \\
\mathrm{BEL}(R,\mathrm{USER\text{-}TYPE}(Q,S),t_1) \wedge \\
\mathrm{BEL}(R,\mathrm{TYPICAL\text{-}MISCONCEPTION}(S,\mathrm{CGEN}(\alpha_n,\gamma,C)),t_1) \\
\rightarrow \\
\mathrm{BEL}(R,\mathrm{EPLAN'}(Q,\gamma,[\alpha_1,\ldots,\alpha_n],\ [\rho_1,\ldots,\rho_n],t_2,t_1),t_1) \\
\text{where } \rho_n=\mathrm{CGEN}(\alpha_n,\gamma,C) \wedge \mathrm{HOLDS}(C,t_2)
\end{array}$$

Axiom (PI10), not shown, should be considered to be the symmetric partner of Axiom (PI9).

Axioms (PI9) and (PI10) assert if R believes that Q is likely to have some particular erroneous beliefs, she can make use of those beliefs in constructing an eplan that underlies Q's query. Misconceptions may be tied to certain characteristics of the questioner, so, for instance, former Teco users are likely to believe that you mark a region in Emacs by setting two marks. The formula USER-TYPE(G,S) encodes the information that G is in some group of users; then the formula TYPICAL-MISCONCEPTION(S,C) encodes the information that users of type S are likely to have some particular misconception C, e.g., former Teco users are likely to have the misconception mentioned just above. Much of the CAI literature has made use of this sort of reasoning [Genesereth 79, Brown 78, Woolf 83, Stevens 79].

CHAPTER VII
Automatic Plan Inference: SPIRIT

In this chapter, SPIRIT--a System for Plan Inference that Reasons about Invalidities Too--is described. SPIRIT is a small demonstration system, implemented in C-Prolog, that illustrates the plan inference model. It infers and evaluates the plans underlying questions asked by users about the domain of computer mail. The advantages of this domain are discussed. Then an overview of SPIRIT is provided, and the algorithms contained in each of its components considered. Finally, several examples of SPIRIT in operation are presented.

7.1. SPIRIT

SPIRIT--a System for Plan Inference that Reasons about Invalidities Too--is a small demonstration system that illustrates the plan inference model developed in this thesis. SPIRIT infers and evaluates the plans underlying questions asked by users about the domain of computer mail. It also uses the result of its inference and evaluation to generate simulated cooperative responses. SPIRIT is implemented in C-Prolog, and has run on several different machines, including a Sun Workstation, a Vax 11-750, and a DEC-20.

In this chapter, I will describe SPIRIT and demonstrate its operation. I will begin by discussing SPIRIT's domain of knowledge: a computer mail system. However, it should be borne in mind that SPIRIT's reasoning processes are quite domain-independent; in fact, while being developed, SPIRIT included another knowledge base, one that contained information about household activities, which was subsequently replaced by the computer-mail knowledge base.

After describing the domain, I will give an overview of SPIRIT, and will discuss the algorithms contained in each of its components. I will then provide several examples of SPIRIT in operation, both using an incorporated tracing facility to walk through the processing of a query, and demonstrating the system in its normal, nontracing mode.

7.2. The Domain

As already mentioned, the domain of knowledge chosen to demonstrate SPIRIT is a computer mail system. However, two caveats are in order. First, the details of the computer mail system encoded in SPIRIT's knowledge base do not correspond to any particular mail system currently in use. Rather, they have been constructed in order to facilitate the demonstration of SPIRIT's capabilities. In all cases, though, the "facts" of the hypothetical mail system about which SPIRIT knows are closely related to facts about either the MAIL system in use in the Department of Computer Science at the University of Pennsylvania, or one of the well-known MM systems running on various mainframes. The reader who has used a computer mail system should find the details of the hypothetical mail system encoded in SPIRIT's knowledge base to be quite reasonable. The second caveat is that the knowledge base is far from complete: only a few key examples, which are sufficient to demonstrate SPIRIT's capabilities, have been implemented. Again, this is because the purpose of SPIRIT is to demonstrate the plan inference model developed in this thesis, not--at least at this point in time--to serve as a useful tool. Of course, the database can be expanded in a straightforward manner.

There are two sorts of reasons for choosing computer mail as a demonstration domain. The first involve the characteristics of the domain itself. Computer mail in particular, and computer systems in general, are attractive domains for any system that reasons about actions. In part this is because within these domains, there is only one agent operating, apart from the system. Interactions among multiple agents' domain plans and goals need not be considered. Also, the effects of actions can be assumed to be certain, and can be precisely stated. When a mail system user, for example, types **SEND**, one can assume that this results in his current message being sent. No such simple assumptions can be made about the effects, say, of investing money in a mutual fund. Finally, computer systems contain a limited but nontrivial number of actions that the user can perform, and this small set of actions can be combined and structured in interesting ways. For the purposes of this thesis, this is particularly important: the domain of computer mail is simple enough to allow the axiomatization of the interactions between at least some of its actions, but rich enough that agents do form invalid plans in it.

In addition to this first set of reasons in support of the computer mail domain, there is a pragmatic reason that it was chosen. The availability of transcripts of naturally occurring dialogues in which mail system users sought advice, as described in Chapter 1, provided support for this choice. Several of the examples taken from or inspired by the transcripts have been discussed in detail throughout this thesis; they have also been implemented in SPIRIT and will be included in the examples of SPIRIT in operation in Section 7.5.

7.3. Overview of the System

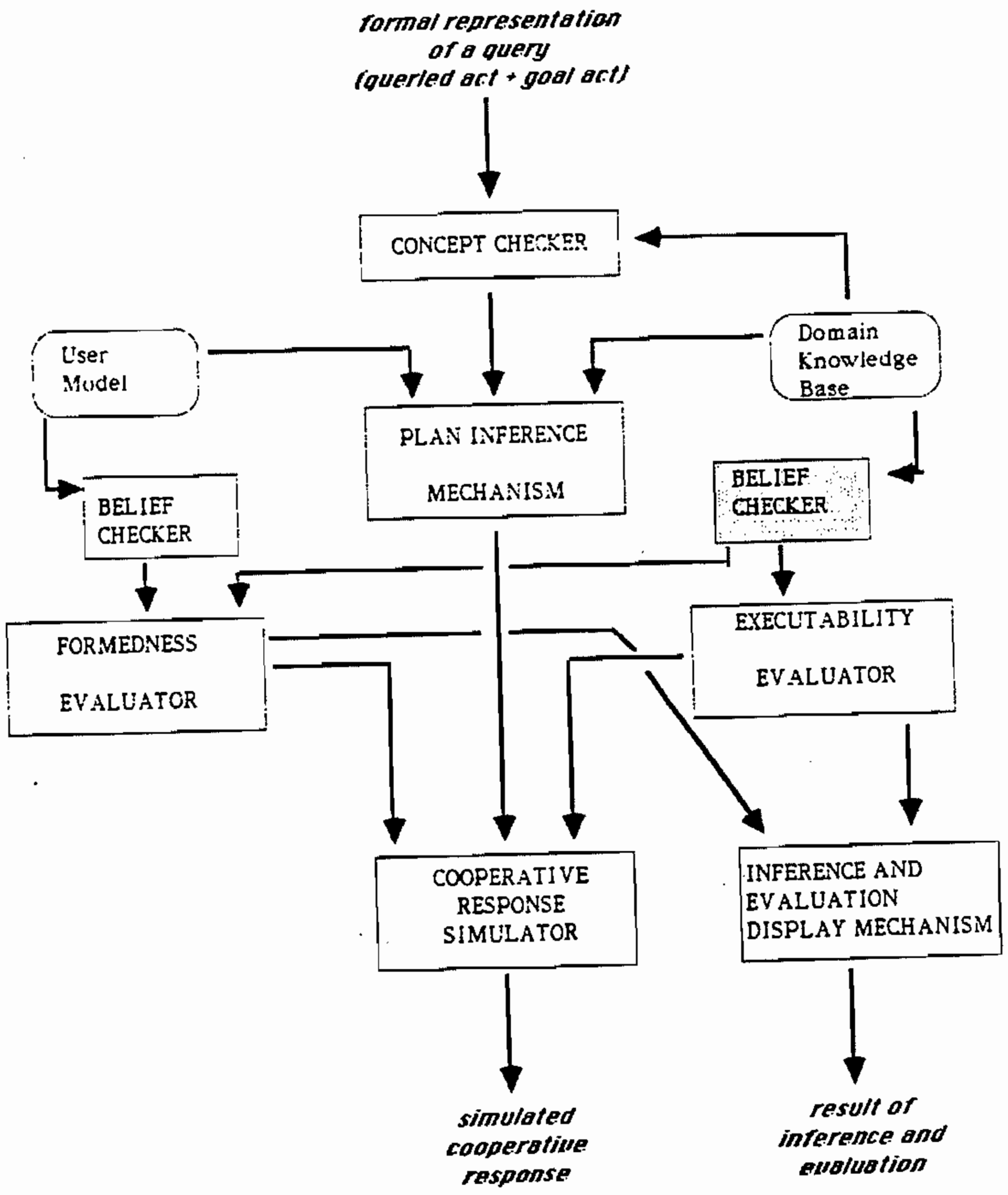
Figure 7-1 is a schematic diagram of SPIRIT. The input and output of SPIRIT are shown in italics. The input consists in a queried act and a goal act: future expansions to SPIRIT should generalize the input allowed. There are two kinds of output produced: one, by the Inference and Evaluation Display Mechanism, is simply a description of the result of the inference and evaluation processes, and the other is a simulated cooperative response. Intermediate output can be produced by a tracing facility that has been incorporated in SPIRIT.

SPIRIT includes two major knowledge stores, indicated in normal typeface inside of boxes: the **Domain Knowledge Base**, which includes SPIRIT's own domain beliefs, and the **User Model**, which includes SPIRIT's beliefs, if any, about the questioner's domain beliefs. It also includes seven major active components, shown in the figure in upper-case letters inside boxes. Briefly, these are:

1.) The Concept Checker--which determines that SPIRIT has at least some part of the act-type concepts in the queried and goal acts. To perform plan inference, it is not necessary for SPIRIT to have the entire concepts about which it is asked: this was the point of the "glothing fewer spurvs" example in Section 4.2. But it is necessary for SPIRIT to have some beliefs about at least the component act-types that form the actions about which it is asked: in the "glothing" example, in the absence of any beliefs about "glothing", it would need to know what it means to do an action to fewer objects, i.e., it would need to have the act-type constructor DO-TO-FEWER(X,Y). If the Concept Checker determines that SPIRIT has no beliefs that it can use to infer the plan behind a query, it rejects that query.

2.) The Plan Inference Mechanism--which takes the queried and goal actions and attempts to find an eplan that relates an intention to do the former to an intention to do the latter. To do this it relies both upon its own beliefs

Figure 7-1: Schematic Diagram of SPIRIT


formal representation of a query (queried act + goal act)
CONCEPT CHECKER
User Model
Domain Knowledge Base
PLAN INFERENCE MECHANISM
BELIEF CHECKER
BELIEF CHECKER
FORMEDNESS EVALUATOR
EXECUTABILITY EVALUATOR
COOPERATIVE RESPONSE SIMULATOR
INFERENCE AND EVALUATION DISPLAY MECHANISM
simulated cooperative response
result of inference and evaluation

about the domain, which are encoded in the Domain Knowledge Base, and upon its beliefs about the user's beliefs, encoded in the User Model.

3.) The Formedness Evaluator--which checks the whether or not an inferred eplan is well-formed. It also relies upon both the Domain Knowledge Base and upon the User Model, to determine whether there are any invalid explanatory beliefs, and if so, what their source is. The Formedness Evaluator, as well as the Executability Evaluator discussed below, do not access the Domain Knowledge Base or the User Model directly, but rather have access to them through the Belief Checker, which computes compound beliefs from simple ones.

4.) The Executability Evaluator--which determines, independently of the Formedness Evaluator, whether or not the queried and goal acts are executable, and if so, how they can be executed.

5.) The Belief Checker--which, as mentioned above, computes compound beliefs from the simple ones encoded in either the Domain Knowledge Base or the User Model. (There is only a single Belief Checker; two copies are shown in Figure 7-1 merely to make the diagram clearer.) The Belief Checker also determines, in cases in which a compound proposition is not believed, why this is so. For example, if it determines that SPIRIT does not believe some proposition $(p \wedge q)$, it keeps track of whether that is because SPIRIT does not believe p, or does not believe q, or both. This information is used in the production of a cooperative response.

6.) The Inference and Evaluation Display Mechanism--which displays the result of the inference and evaluation processes. It is a merely a formatting procedure.

7.) The Cooperative Response Simulator--which takes the inferred eplan, if there is one, as well as the evaluations of formedness and executability, and produces a cooperative response. No notion of relevance or salience is presently incorporated in the Response Simulator: instead, it always provides as much information as possible. As I discussed in Section 5.1, producing such responses does not directly model human conversational ability, which does rely heavily on relevance and salience. But the production of these full responses does serve to demonstrate the complete inference and evaluation processes, and further, as was also stated in Section 5.1, a question-answering system capable of generating such responses will perform better than one that does not attempt to infer or to take account of the eplans underlying the queries it is asked.

The reader may notice a minor difference between the SPIRIT schematic diagram in Figure 7-1 and the earlier schematic diagram of the process of answering a question, given in Figure 6-1. In Figure 6-1, executability evaluation, incorporated within plan synthesis, is shown to follow formedness evaluation, and to use the latter's output as input. In SPIRIT, executability evaluation and formedness evaluation proceed independently of one another--in fact, given a parallel architecture, these two mechanisms could run simultaneously. This difference between SPIRIT and the earlier account of question-answering arises because SPIRIT has no relevance or salience mechanisms. Previously, I postulated that in actual question-answering, an agent may, for example, decide not to attempt to compute whether some queried act is executable if she has already decided that the eplan is ill-formed: she would decide this if she felt that, as a result of the ill-formedness, the executability of the queried act was irrelevant. SPIRIT errs on the side of assuming that all information about the inferred eplan is relevant; it therefore always computes executability, even in cases of ill-formedness.

7.4. System Components

I will now return to each of the major system components, except the Inference and Evaluation Display Mechanism, to describe its operation in a bit more detail. Specifically, I will be concerned with the algorithms that each contains.

The Concept Checker

The operation of the Concept Checker is straightforward: it merely decomposes the queried and goal actions into their component act-type constructor functions, and then searches the Domain Knowledge Base for some beliefs about those components. If it succeeds for some component act-type constructor function of the queried act, and for some component act-type constructor function of the goal act, then SPIRIT has beliefs about both, and proceeds to attempt to infer an eplan. Otherwise, it rejects the query, informing the user that it does not know about one or both of the acts mentioned in his query. The Concept Checker will inform the user whether both his queried and goal acts are unknown to SPIRIT, or whether only one is.

The Plan Inference Mechanism

The Plan Inference Mechanism in SPIRIT has been implemented as a bidirectional graph search. It constructs two search graphs: one rooted in a node representing the queried act, and one rooted in a node representing the goal act. The remaining nodes in the two search graphs represent acts that SPIRIT believes the user plausibly intends, while the arcs are labeled, so that the arc between a node representing an act of α and a node representing an act of β denotes the explanatory belief that relates α and β in a plausible eplan. Thus each path through one of the graphs represents a plausible eplan. That is, when SPIRIT believes that the user, Q, plausibly has some eplan:

$$\text{BEL(SPIRIT, EPLAN'}(Q, \alpha_n, [\alpha_1,\ldots,\alpha_{n-1}], [\rho_1,\ldots,\rho_{n-1}], t_2, t_1), t_1),$$
$$\text{where each } \rho_i = \text{CGEN}(\alpha_i, \alpha_{i+1}, C_i) \wedge \text{HOLDS}(C_i, t_2)$$

one of the two graphs that the Plan Inference Mechanism is searching will include the following path:

$$\begin{tikzcd}
\alpha_1 \arrow[r, dashed, "C_1"] & \alpha_2 \arrow[r, dashed, "C_2"] & {} & \cdots & {} \arrow[r, dashed, "C_{n-1}"] & \alpha_n
\end{tikzcd}$$

Only the generation-enabling condition needs to be explicitly encoded on the arc between any two nodes: by itself it is sufficient for reconstructing the entire explanatory belief.

The Plan Inference Mechanism starts with two small graphs: one containing a single node representing the queried action (and no arcs), and one containing a single node representing the goal action (and no arcs). It then selects one of these, and applies to it the various plan inference rules, which relate acts that are believed to be plausibly intended to further acts that are believed plausibly intended, as described in Chapter 6. The application of a plan inference rule is modeled as the introduction of further nodes into the graph containing the current node. Of course, as each node is introduced, paths in the graph are expanded, thereby modeling the fact that SPIRIT has come to believe that there are additional plausible eplans for Q. After expanding one of the nodes, a new current node is chosen: it may come either from the graph that was previously being examined or from the other graph. An attempt is then made to see whether the newly chosen node can be merged with one of the nodes in the other graph. If it can, SPIRIT has found a set of beliefs that is large enough to relate Q's intention to do his queried act to his intention to do his goal act, and the path through both graphs that meets at the node in question represents an eplan, and not just a plausible eplan. It can be extracted, and passed along to the evaluation components. On the other hand, if the new current node cannot be merged with any nodes from the other graph, the expansion process is repeated. If at some time there are no more nodes to expand, the process is terminated and the query being analyzed is judged to be incoherent. With a much larger knowledge base, exhaustive search such as this would be unfeasible, and a cutoff mechanism that causes a query to be judged incoherent after a particular number of node expansions would need to be introduced. Introduction of such a mechanism corresponds to the weakening of the definition of incoherence, as discussed in Chapter 6.

The graph search algorithm in the Plan Inference Mechanism differs from a "vanilla" graph search, such as that described in Nilsson [Nilsson 80, pp. 64-65], in three ways. First, it is bidirectional: the search proceeds both from the queried act and from the goal act, making use of symmetric plan inference rules as discussed in Chapter 6. This bidirectionality necessitates the introduction of additional mechanisms for maintaining the two graphs separately. It also necessitates an alternate success test: the entire graph is complete if the two subgraphs merge. This obviously differs from the standard success test for unidirectional search, in which the graph is complete if any node in it satisfies some goal condition. The second way in which SPIRIT's Plan Inference Mechanism differs from a "vanilla" graph-search algorithm is that it maintains labels on the arcs of the graph. This is implemented by constructing complex nodes each of which includes, along with other information, the label of its incoming arc. Finally, SPIRIT's nodes and arcs need not be completely instantiated at the time that they are introduced into the graph. They can contain variables, and in applying the rules to expand them, unification is allowed to occur. The instantiation that is the result of a unification step must be iterated throughout the path that the node in question lies on. This need for unification is one of the reasons that Prolog was chosen as the implementation language for SPIRIT.

The implementation of plan inference as graph search is fairly standard; Nilsson [Nilsson 80] and Kautz [Kautz 85] contain discussion of this. What differs in SPIRIT is, for one thing, that the standardly used inference rules, which allow only valid arcs to be introduced, have been expanded, so that arcs that are not valid can also be introduced; act-type constructor functions have also been allowed so that nodes representing unexecutable actions can be introduced. In addition, SPIRIT's Plan Inference Mechanism differs from existing plan inference systems by including the inference of explanatory beliefs, which enables it to distinguish incoherent queries from those with ill-formed underlying plans, and which are useful in constructing a cooperative response. Also, careful analysis has been given to the beliefs represented by the graphs that SPIRIT constructs and the path that it may finally extract--the former corresponding to the plausible eplans, and the latter to a (full-fledged) eplan. This too sets SPIRIT apart from other plan inference systems. And finally, the Plan Inference Mechanism is only one part of SPIRIT: SPIRIT also contains the Evaluation components and the Cooperative Response Simulator.

The Formedness Evaluator

The job of the Formedness Evaluator is to determine whether SPIRIT believes each of the explanatory beliefs in the inferred eplan. Sometimes, problems with an explanatory belief are noted during the operation of the Plan Inference Mechanism. For instance, when a node is introduced by one of the rules that encode reasoning about confusion between similar actions, or reasoning about typical misconceptions--Axioms (PI5) through (PI10)--an annotation of this fact is included in the label of the node's incoming arc. In such cases, the job of the Formedness Evaluator is quite simple: it needs only to pass along this information to the Cooperative Response Simulator.

For each arc that has not been pre-annotated by the Plan Inference Mechanism, the Formedness Evaluator must call the Belief Checker to determine whether SPIRIT believes the explanatory belief that labels it. If the Belief Checker confirms that SPIRIT so believes, then the Formedness Evaluator annotates the eplan to indicate that the section of the eplan containing that arc is well-formed. If, on the other hand, the Belief Checker determines that SPIRIT does not so believe, the Formedness Evaluator indicates that the section of the eplan is ill-formed. In such cases, it must further distinguish between ill-formedness that is due to Q's belief in an erroneous cgen-axiom, such as would be introduced by the use of Axiom (PI3) or Axiom (PI4), and ill-formedness that is due to Q's erroneous belief that some generation-enabling condition will hold, such as might be introduced by the use of Axiom (PI1) or Axiom (PI2). To do this, the Formedness Evaluator invokes the Belief Checker to determine whether SPIRIT believes that Q believes to be false the generation-enabling conditions of the relevant explanatory belief.

If the Belief Checker determines that SPIRIT does believes that Q so believes, the Formedness Evaluator will decide that the ill-formedness is due to

Q's having an erroneous cgen-axiom. After all, if SPIRIT believes that Q believes to be false some generation-enabling condition C necessary for an act of α to generate an act of β, then since SPIRIT also believes that Q intends to generate β by α, it must conclude that Q does not know that C is a generation-enabling condition. When the Belief Checker cannot determine that SPIRIT believes that Q believes C to be false, the Formedness Evaluator will decide that the ill-formedness is due to Q's erroneously believing it to be true. The presumption is in favor of this latter sort of ill-formedness; to decide for the former, the Formedness Evaluator must receive confirmation from the Belief Checker that SPIRIT believes that Q believes false the fact in question: it is not enough for SPIRIT not to know whether Q holds that belief. The upshot of this reasoning is the annotation of the eplan with information that will allow the Cooperative Response Simulator to decide between a response like that in Example (39) and a response like that in Example (40).

The Executability Evaluator

The Executability Evaluator, like the Plan Inference Mechanism, is implemented as a graph search. However, the Executability Evaluator performs a unidirectional graph search. It begins with a node representing either the queried or the goal act, and reasons backward, looking for a path from that node to a node representing a basic act. The reason for searching unidirectionally here is to cut down on the size of the search: in principle, there are too many basic actions in any domain to reason forward from them.

In the node expansion phase, the Executability Evaluator finds all the act-types that conditionally generate the type of act represented by the current node. For each such act-type, it then checks whether or not SPIRIT believes that the generation-enabling condition relating it to the act-type of the current node holds. Only if the generation-enabling condition is believed true is a new node introduced into the graph. This has the effect of filtering out what amounts to ill-formed plans leading the act whose executability is being computed. All acts in the graph will, according to SPIRIT's beliefs, actually form generational pairs with each of the acts to which they are connected. This differs from the graphs inferred by the Plan Inference Mechanism, in which nodes can be introduced by rules such as Axioms (PI3) through (PI10), and in which even those nodes introduced by Axioms (PI1) and (PI2) may represent acts that fail to stand in a generation relation to the acts connected to them, because the requisite generation-enabling condition will not hold.

To test whether the search has succeeded, the Executability Evaluator checks whether the current node represents an act that is basic. For the computer mail domain, all acts of typing something are stipulated to be basic. I have made the simplifying assumption that agents are always in the standard conditions with respect to these acts. Thus, if a node represents a basic act, all nodes connected to it will represent executable acts. This is because, according to Definition (E1), acts are executable if they are generated by basic acts. The starting node of the search, which represents the act whose executability is being determined, will of course be connected to all other nodes in the graph, and therefore will be executable precisely when the search succeeds.

Although the Executability Evaluator will not judge an act to be executable when it is not, it may judge an act to be unexecutable when it really can be executed. That is because when the Executability Evaluator determines that SPIRIT believes false the generation-enabling conditions for some pair of act-types, it does not search for a way for Q to make that condition true. No attempt has been made to incorporate such facilities into SPIRIT because developing techniques for doing so in a principled way is large research question in its own right. As has already been discussed, it relies in part of a theory that distinguishes between what is within an agent's control and what is not; see Pelavin [Pelavin 86]. It also relies on account of reasonableness. If SPIRIT determines that Q's plan is ill-formed because Tom is the system manager and therefore setting the permissions on a file will not prevent him from reading it, then even if SPIRIT believes that it is within Q's power to cause Tom no longer to be the system manager, it may not be reasonable for it to consider ways for Q to bring this about.

The Belief Checker

The Belief Checker acts as an interface between the knowledge stores--the Domain Knowledge Base, which encodes SPIRIT's beliefs about the domain, and the User Model, which encodes SPIRIT's beliefs about the user's beliefs--and two of the active components that need to have access to these beliefs, specifically, the Formedness Evaluator and the Executability Evaluator. Such an interface is necessary because although the knowledge stores by and large contain unit clauses, the active components may need to know whether SPIRIT believes arbitrary logical combinations of these clauses.[72] So, for instance, while processing a query in the computer mail domain, the Formedness Evaluator may need to know whether SPIRIT believes that an agent Tom is neither the system manager nor a member of the faculty, i.e.:

```
?- believe(spirit, and(not(system_mgr(tom), not(belongs_to(to
m,faculty)))).
```

However, what the Domain Knowledge Base includes is a unit clause denoting the fact that SPIRIT believes that Tom is the system manager, i.e.,

```
system_mgr(tom).
```

It is the task of the Belief Checker is to determine from this that SPIRIT does not believe the conjunction expressed above.

The Belief Checker thus includes rules about first-order logical connectives. It adopts the Prolog notion of negation-as-failure: if it cannot prove that SPIRIT believes some sentence p, it assumes that SPIRIT believes not(p). Another feature of the Belief Checker is that, when it determines that SPIRIT does not believe some sentence, it keeps track of the reason for this. This is particularly useful in cases in which it is checking for belief in a conjunctive sentence. For the example in the previous paragraph, the Belief Checker should report to the component that invoked it not just that SPIRIT does not have the belief in question, but also that the reason it does not, namely, it does not believe that Tom is not the system manager (though it does believe, as a result of the negation-as-failure principle, that he is not a member of the faculty). Reporting this information facilitates the production of a more cooperative response: SPIRIT can tell the user that because Tom is the system manager, setting the permissions as requested will not prevent him from reading the file. Without the Belief Checker keeping track of the reason the conjunctive belief is thought false, all that SPIRIT could tell the user is that setting the permissions as requested will not prevent Tom from reading the file because either he is the system manager, or he is a member of the faculty, or both.

[72]The knowledge stores may also contain nonunit clauses: for example, we might want to encode certain implications that SPIRIT believes. What cannot be the case is that *all* of SPIRIT's beliefs are explicitly encoded in the knowledge stores, since given Axiom (B1), if SPIRIT has one belief, it has infinitely many.

Cooperative Response Simulator

The Cooperative Response Simulator uses the result of the plan inference and evaluation mechanisms to simulate the production of a cooperative response. It does not include any sophisticated language-generation capabilities: rather, it simulates the response production by using canned text. The responses it produces are tailored to the combinations of invalidities found: it produces different skeleton responses for each of the seven possible configurations of judgements of formedness, executability of the queried act, and executability of the goal act, as listed in Figure 5-2. Further variation in responses correlates with the particular types of any ill-formedness found in the eplan.

In a future version of SPIRIT, the Cooperative Response Simulator should be replaced by a combination of a strategic language-generation component, which uses the output of the plan inference and evaluation mechanisms, along with relevance and salience mechanisms, to determine how much of the computed information should be presented to the user, and a tactical language-generation component, which then transforms that information into English text.[73] The reason for including the Cooperative Response Simulator in the current version of SPIRIT is that with even simulated English responses as output it is easier to see the results of the inference and evaluation processes and to appreciate their effects on the construction of cooperative response, then it is with only the data structures output by the Inference and Evaluation Display Mechanism.

7.5. SPIRIT in Operation

We can now turn to several examples of SPIRIT in operation. To begin, we will consider a few variations of the example in which Q wants to set the permissions on his file--this time we will consider it to be his mailfile--in order to keep another agent from having access to it. For the first example, SPIRIT believes the following cgen-axiom, discussed previously in Chapter 6:

```
cgen(set_permissions(X,P,Y),
     prevent(X,P,Z),
     and(not(belongs_to(Z,Y)), not(system_mgr(Z)))).
```

It also believes that Tom is the system manager:

[73]For discussion of the difference between strategic and tactical components in language-generation systems, see [McKeown 85].

```
system_mgr(tom).
```

So far, it has no beliefs about what Q might believe.

The tracing facility incorporated in SPIRIT will be turned on in the example, so that we can examine its operation in detail. Information from the tracer begins with the string "++++++". I have inserted comments in italics. All other text through the end of the example is output from SPIRIT.

Example I of SPIRIT in Operation

```
************************************

Welcome to SPIRIT:
System for Plan Inference that Reasons about Invalidities Too

Do you have a query?  yes.
Queried Action?  set_permissions(mailfile,read,faculty).
Goal Action?  prevent(mailfile,read,tom).

++++++ Checking for the concepts set_permissions(mailfile,read
,faculty) and prevent(mailfile,read,tom)
```

The Concept Checker is making sure that SPIRIT has some beliefs about the queried and goal acts. It succeeds, and so the Plan Inference Mechanism is invoked.

```
++++++ Attempting to find an eplan relating an intention to do
set_permissions(mailfile,read,faculty) to an intention to do p
revent(mailfile,read,tom)
```

The Plan Inference Mechanism is now operating, attempting to find an eplan that underlies Q's query.

```
++++++ Checking for completion.  Current node is node(b1,b,pre
vent(mailfile,read,tom),nil,nil,1)
```

The six components in each node are: (1) a unique identifier; (2) its search direction--i.e., whether it is part of the forward graph or the backward graph; (3) the type of the act it denotes; (4) the unique identifier of its parent node; (5) the label of its incoming *arc, which is set to nil for the start nodes of both graphs; and (6) its rank, used in selecting the next node to expand.*

```
++++++ Graph not complete.  Expanding current node.

++++++ Finding nodes related to: node(b1,b,prevent(mailfile,re
ad,tom),nil,nil,1)

++++++ Adding the following nodes:

     node(b2,b,set_permissions(mailfile,read,_192),b1,and(not
belongsto(tom,_192),not system_mgr(tom)),0)
```

In this expansion cycle, only one new node has been added to the graph. The Plan Inference Mechanism will now select the highest ranking open node to expand next.

```
++++++ Selecting highest ranking open node

++++++ Highest ranking in forward direction is node(f1,f,set_p
ermissions(mailfile,read,faculty),nil,nil,1)

++++++ Highest ranking in backward direction is node(b2,b,set_
permissions(mailfile,read,_192),b1,and(not belongs_to(tom,_192
),not system_mgr(tom)),0)

++++++ Selecting as next node to expand: node(f1,f,set_permiss
ions(mailfile,read,faculty),nil,nil,1)

++++++ Increasing the rank of all nodes remaining open
```

When nodes are initially introduced into the graph, they are assigned a rank as follows: nodes introduced by Axioms (PI1) through (PI4) receive an initial rank of one less than their parent node; nodes introduced by Axioms (PI5) through (PI8), which encode reasoning about confusion over similar act, receive an initial rank of three less than their parent node; and nodes introduced by Axioms (PI9) and (PI10), which encode reasoning about typical misconceptions, receive an initial rank of two less than their parent node. Then, each expansion cycle that a node survives, its rank is increased by one.

```
++++++ Rank of node b2 ( set_permissions(mailfile,read,_192) )
increased to 1

++++++ Checking for completion.  Current node is node(f1,f,set
_permissions(mailfile,read,faculty),nil,nil,1)

++++++ Extracting plan from graph.
```

The two graphs can be merged, so the search is successful, and an eplan can be extracted.

```
++++++ Eplan to evaluate:

   set_permissions(mailfile,read,faculty)
               and(not belongs_to(tom,faculty),not system_mgr(
tom))
   prevent(mailfile,read,tom)

++++++ Determining executability of set_permissions(mailfile,r
ead,faculty)
```

The Executability Evaluator is now operating. It will first attempt to determine whether the queried act is executable.

```
++++++ Testing whether set_permissions(mailfile,read,faculty)
denotes a basic action.

++++++ Current act not basic. Expanding.

++++++ Considering the following expansions:

     pair(type(set_protections(mailfile,faculty,read)),and(or(
read=read,or(read=write,read=delete)),or(faculty=group,faculty
=owner)))
```

The Executability Evaluator is considering creating and adding a node whose act-type is the first member of the pair, and whose incoming arc is labeled with the second member. However, it will not do so unless it determines that SPIRIT believes the generation-enabling conditions that would label the introduced arc. Thus the Belief Checker is called.

```
++++++ Checking whether spirit believes and(or(read=read,or(re
ad=write,read=delete)),or(faculty=group,faculty=owner))
```

The Belief Checker will now check each conjunct.

```
++++++ Checking whether spirit believes or(read=read,or(read=w
rite,read=delete))
```

The first conjunct is itself a disjunction. The Belief Checker will check the disjuncts until it finds one that SPIRIT believes.

```
++++++ Checking whether spirit believes read=read

++++++ spirit believes read=read
```

Since the first disjunct succeeds, the whole disjunction that the first conjunct comprises succeeds. The Belief Checker can now move on to the second conjunct, which is also a disjunction.

```
++++++ Checking whether spirit believes or(faculty=group,facul
ty=owner)

++++++ Checking whether spirit believes faculty=group

++++++ spirit does not believe faculty=group

++++++ Checking whether spirit believes faculty=owner

++++++ spirit does not believe faculty=owner
```

None of the disjuncts are believed true by SPIRIT, so the entire disjunction fails. As a result, the conjunction of which this disjunction is part fails as well. There are no qualifying nodes for the Executability Evaluator to add to the graph.

```
++++++ Adding the following nodes:
```

At this point, there are no nodes left to expand, so the search fails; the Executability Evaluator judges the queried act to be unexecutable.

```
+++++ Search failed:  set_permissions(mailfile,read,faculty) j
udged not executable.
```

The process is now repeated for the goal act.

```
++++++ Determining executability of prevent(mailfile,read,tom)

++++++ Testing whether prevent(mailfile,read,tom) denotes a ba
sic action.

++++++ Current act not basic. Expanding.

++++++ Considering the following expansions:

     pair(set_permissions(mailfile,read,_745),and(not belongs_
to(tom,_745),not system_mgr(tom)))

++++++ Checking whether spirit believes and(not belongs_to(tom
,_745),not system_mgr(tom))

++++++ Checking whether spirit believes not belongs_to(tom,_74
5)

++++++ Checking whether spirit believes belongs_to(tom,_745)

++++++ Checking whether spirit believes not system_mgr(tom)

++++++ Checking whether spirit believes system_mgr(tom)

++++++ spirit believes system_mgr(tom)

++++++ Adding the following nodes:


+++++ Search failed:  prevent(mailfile,read,tom) judged not ex
ecutable.
```

The goal act is also judged to be unexecutable. Executability Evaluation is now complete. The Formedness Evaluator will now be invoked.

```
++++++ Determining formedness of inferred eplan

++++++ Checking whether spirit believes and(not belongs_to(tom
,faculty),not system_mgr(tom))
```

The Belief Checker is called to determine whether SPIRIT believes the generation-enabling condition on the first arc of the plan.

```
++++++ Checking whether spirit believes not belongs_to(tom,fac
ulty)

++++++ Checking whether spirit believes belongs_to(tom,faculty
)

++++++ Checking whether spirit believes not system_mgr(tom)

++++++ Checking whether spirit believes system_mgr(tom)

++++++ spirit believes system_mgr(tom)

++++++ spirit does not believe and(not belongs_to(tom,faculty)
,not system_mgr(tom))
```

The Belief Checker has determined that SPIRIT does not believe the generation-enabling condition on the arc. Consequently, the Formedness Evaluator now invokes the Belief Checker again, this time to determine whether SPIRIT believes that Q also believes that the generation-enabling condition will be false. Notice that only that conjunct that SPIRIT believes false--that Tom is not the system manager--is checked.

```
++++++ Checking whether spirit believes Q knows to be false an
y of:
     not system_mgr(tom)
```

The Belief Checker could not prove that Q believes this false, so it annotates the eplan to reflect an ill-formedness due to Q's having an erroneous belief about the truth of a generation-enabling condition.

```
++++++ Noting ill-formedness between set_permissions(mailfile,
read,faculty) and prevent(mailfile,read,tom) :

     sup_fact(set_permissions(mailfile,read,faculty),prevent(m
ailfile,read,tom),and(not belongs_to(tom,faculty),not system_m
gr(tom)),[not system_mgr(tom)])
```

Formedness Evaluation is now complete. The two output procedures will be called. First, the Inference and Evaluation Display Mechanism:

```
==========================================
 Plan Inference:
==========================================
Query COHERENT
INFERRED EPLAN:
   set_permissions(mailfile,read,faculty)
               and(not belongs_to(tom,faculty),not system_mgr(
tom))
   prevent(mailfile,read,tom)

==========================================
 Plan Evaluation:
==========================================
The queried action is UNEXECUTABLE.
The goal action is UNEXECUTABLE.
The inferred eplan is ILL-FORMED:
The following conditions which were
inferred to relate
set_permissions(mailfile,read,faculty) and
prevent(mailfile,read,tom) will not be true:
 not system_mgr(tom)
```

And now the Cooperative Response Simulator:

```
==========================================
Sample Cooperative Response:
==========================================

There is no way for you to do
set_permissions(mailfile,read,faculty) , and even
if you could it would not lead to doing
```

```
prevent(mailfile,read,tom) . That is because not
system_mgr(tom) will not be true . There is no way
for you to do prevent(mailfile,read,tom) .

=========================================
=========================================
```

To demonstrate how SPIRIT behaves when it determines that an ill-formedness is due to an erroneous cgen-axiom rather than an erroneous belief about a generation-enabling condition, I will next add the belief that Q believes that Tom is the system manager, and then provide the same input as before. This time however I will turn the tracing off.

Example II of SPIRIT in Operation

```
| ?- assert(believe(q, system_mgr(tom))).
yes
| ?- spirit.
************************************

Welcome to SPIRIT:
System for Plan Inference that Reasons about Invalidities Too

Do you have a query?  yes.
Queried Action?  set_permissions(mailfile,read,faculty).
Goal Action?  prevent(mailfile,read,tom).

=========================================
 Plan Inference:
=========================================
Query COHERENT
INFERRED EPLAN:
   set_permissions(mailfile,read,faculty)
                 and(not belongs_to(tom,faculty),not system_mgr(
tom))
   prevent(mailfile,read,tom)

=========================================
 Plan Evaluation:
=========================================
The queried action is UNEXECUTABLE.
The goal action is UNEXECUTABLE.
The inferred eplan is ILL-FORMED:
The rule that was inferred to relate
set_permissions(mailfile,read,faculty) and
prevent(mailfile,read,tom) is not a valid rule.
The following generation-enabling conditions
were omitted:
 not system_mgr(tom)


=========================================
Sample Cooperative Response:
=========================================

There is no way for you to do
set_permissions(mailfile,read,faculty) , and even
if you could it would not lead to doing
prevent(mailfile,read,tom) . That is because
for you to do prevent(x,p,z) by doing
set_permissions(x,p,y) all of these conditions
must be true: not belongs_to(z,y) and not
system_mgr(z) . But, as I believe you know, not
system_mgr(tom) will not be true . There is no way
for you to do prevent(mailfile,read,tom) .
```

Notice the difference in SPIRIT's response. Whereas before it informed the user that Tom is the system manager, in a fashion reminiscent of Example (39), this time it is reminiscent of Example (40), telling the user that an additional condition must hold for his plan to be well-formed.

SPIRIT is also able to reason about a combination of problems in any explanatory belief, as illustrated by the following example. Here I will add the belief that Tom is a member of the faculty. I will also turn the tracing back on, but will only display the interesting portion, which occurs while the Formedness Evaluator is operating.

Example III of SPIRIT in Operation

```
| ?- assert(belongs_to(tom, faculty)).
yes
| ?- spirit.
```

```
************************************

Welcome to SPIRIT:
System for Plan Inference that Reasons about Invalidities Too

Do you have a query?  yes.
Queried Action?  set_permissions(mailfile,read,faculty).
Goal Action?  prevent(mailfile,read,tom).
```

Tracing up until here is deleted.

```
++++++ Determining formedness of inferred eplan

++++++ Checking whether spirit believes and(not belongs_to(tom
,faculty),not system_mgr(tom))

++++++ Checking whether spirit believes not belongs_to(tom,fac
ulty)

++++++ Checking whether spirit believes belongs_to(tom,faculty
)

++++++ spirit believes belongs_to(tom,faculty)

++++++ Checking whether spirit believes not system_mgr(tom)

++++++ Checking whether spirit believes system_mgr(tom)

++++++ spirit believes system_mgr(tom)

++++++ spirit does not believe and(not belongs_to(tom,faculty)
,not system_mgr(tom))
```

The Belief Checker has determined that SPIRIT believes that both conjuncts of the generation-enabling condition are false. The Formedness Evaluator will now attempt to determine what SPIRIT's beliefs about Q's beliefs about each conjunct are.

```
++++++ Checking whether spirit believes Q knows to be false an
y of:
     not belongs_to(tom,faculty)
     not system_mgr(tom)

++++++ spirit believes that Q knows that not system_mgr(tom) i
s false
```

The Belief Checker has determined that SPIRIT believes that Q believes that Tom is the system manager, and therefore, that Q does not believe that Tom is not the system manager. However, it cannot prove that SPIRIT believes that Q believes that Tom is a member of the faculty, so it assumes that Q does not believe this. Accordingly, it annotates the arc in the eplan to indicate that part of the ill-formedness is due to an erroneous cgen-axiom that Q has, and part is due to an erroneous belief of Q's about the truth of a generation-enabling condition.

```
++++++ Noting ill-formedness between set_permissions(mailfile,
read,faculty) and prevent(mailfile,read,tom) :

     sup_rule(set_permissions(mailfile,read,faculty),prevent(m
ailfile,read,tom),and(not belongs_to(tom,faculty),not system_m
gr(tom)),[not system_mgr(tom)])
     sup_fact(set_permissions(mailfile,
read,faculty),prevent(mailfile,read,tom),and(not belongs_to(to
m,faculty),not system_mgr(tom)),[not belongs_to(tom,faculty)])

=========================================
 Plan Inference:
=========================================
Query COHERENT
INFERRED EPLAN:
   set_permissions(mailfile,read,faculty)
               and(not belongs_to(tom,faculty),not system_mgr(
tom))
   prevent(mailfile,read,tom)

=========================================
 Plan Evaluation:
=========================================
The queried action is UNEXECUTABLE.
The goal action is UNEXECUTABLE.
The inferred eplan is ILL-FORMED:
The rule that was inferred to relate
set_permissions(mailfile,read,faculty) and
prevent(mailfile,read,tom) is not a valid rule.
The following generation-enabling conditions
were omitted:
 not system_mgr(tom)


Also, the following conditions which were
included in the rule inferred to relate
set_permissions(mailfile,read,faculty) and
prevent(mailfile,read,tom) will not be true:
 not belongs_to(tom,faculty)


=========================================
Sample Cooperative Response:
=========================================

There is no way for you to do
set_permissions(mailfile,read,faculty) , and even
if you could it would not lead to doing
prevent(mailfile,read,tom) . That is because
for you to do prevent(x,p,z) by doing
set_permissions(x,p,y) all of these conditions
must be true: not belongs_to(z,y) and not
system_mgr(z) . But, as I believe you know, not
system_mgr(tom) will not be true . Also, not
belongs_to(tom,faculty) will not be true .
There is no way for you to do
prevent(mailfile,read,tom) .

=========================================
=========================================
```

So far, all the examples have been variations of a query with unexecutable queried and goal acts, whose inferred eplan is ill-formed in different ways. To see SPIRIT's handling of other configurations of invalidities, imagine that there is another agent, called Kelly; SPIRIT believes that Kelly is not the system manager, and that she is neither a member of the faculty nor a member of Q's group. In the next example, the queried act is still unexecutable, since there is still no way to set the permissions on a file to faculty-read only. However, the plan is well-formed, since if there were a way to do so, it would prevent Kelly from reading the file. Also, SPIRIT will determine that the goal act is executable. Tracing has been turned off.

Example IV of SPIRIT in Operation

```
Do you have another query?  yes.
Queried Action?  set_permissions(mailfile,read,faculty).
Goal Action?  prevent(mailfile,read,kelly).

========================================
 Plan Inference:
========================================
Query COHERENT
INFERRED EPLAN:
   set_permissions(mailfile,read,faculty)
               and(not belongs_to(kelly,faculty),not system_m
gr(kelly))
   prevent(mailfile,read,kelly)

========================================
 Plan Evaluation:
========================================
The queried action is UNEXECUTABLE.
The goal action is EXECUTABLE.  To execute it:

>>> type(set_protections(mailfile,group,read))

The inferred eplan is WELL-FORMED.
========================================
Sample Cooperative Response:
========================================

There is no way for you to do
set_permissions(mailfile,read,faculty) ,
but you can do prevent(mailfile,read,kelly)
by doing
type(set_protections(mailfile,group,read)) .
```

```
=========================================
=========================================
```

So, although there is no way to set the permissions on a file to faculty-read only, there is a way to set it to group-read only, and doing this will prevent Kelly from reading it. Of course, if Q had known this all along, her query would have had a valid plan, as in the following example:

Example V of SPIRIT in Operation

```
Do you have another query?  yes.
Queried Action?  set_permissions(mailfile,read,group).
Goal Action?  prevent(mailfile,read,kelly).


=========================================
 Plan Inference:
=========================================
Query COHERENT
INFERRED EPLAN:
   set_permissions(mailfile,read,group)
               and(not belongs_to(kelly,group),not system_mgr(
kelly))
   prevent(mailfile,read,kelly)

=========================================
 Plan Evaluation:
=========================================
The queried action is EXECUTABLE.  To execute it:

>>> type(set_protections(mailfile,group,read))

The goal action is EXECUTABLE.  To execute it:

>>> type(set_protections(mailfile,group,read))

The inferred eplan is WELL-FORMED.
=========================================
Sample Cooperative Response:
=========================================
You can do set_permissions(mailfile,read,group)
by doing
type(set_protections(mailfile,group,read)) .

=========================================
=========================================
```

To see a few more of SPIRIT's capabilities, we can now turn to another set of examples. In these, SPIRIT processes variations of the query about loading only the new messages into mail, also discussed in Chapter 6. SPIRIT's Domain Knowledge Base includes the following relevant cgen-axioms:

```
cgen(do_to(X,S),
     do_to_fewer(X, S1),
     and(normal_obj(X, S1), smaller(S, S1))).

cgen(do_to_fewer(X, S1),
     speed_up(X),
     speed_proportional_to_size(X)).

cgen(speed_up(X),
     speed_up(Y),
     part_of(X,Y)).

cgen(comp([spawn_proc, load(X)]), enter_mail, no_errors).
```

The first axiom says that the act of doing X to some set of items S generates an act of doing X to fewer items if S is a smaller than set than the items in some set S1 that X is normally done to. This general cgen-axiom is meant to represent intuitive knowledge about what it means to "do an action to fewer items (than it is normally done to)". The second axiom says that by doing an act X to fewer items than it is normally done to, one can speed up X if its speed is proportional to the size of the set of objects it is performed on. The third axiom says that by speeding up one act X, one can speed up another act Y, if X is part of Y. To make use of this axiom, SPIRIT must have a belief about what it means for one act to be part of another. This is encoded as follows:

```
part_of(X,Y) :-
    cgen(comp(L), Y, _),
    member(X, L).
```

The *comp* predicate (for "composition") replaces the semi-colon notation introduced earlier to denote a sequence of acts: if a1, a2, and a3 are act-types, then comp([a1, a2, a3]) denotes their temporal concatenation, i.e., what was earlier written as a1;a2;a3. Thus one act is part of another if the former is a member of a sequence of acts generating the latter. The final cgen-axiom in the set of four is the first example we have seen in SPIRIT of a sequence of generating acts. It encodes the fact that the sequential performance of a spawning act and a loading act conditionally generates an act of entering mail.

We might also want SPIRIT to be able derive beliefs about conditional generation from its cgen-axioms. One example of this is the following rule, included in SPIRIT's Domain Knowledge Base:

```
cgen(X, do_to(Y,S), Z) :- normal_obj(X,S), cgen(X,Y,Z).
```

This rule says that whenever the normal object that X operates on is some set S, then if acts of X generate acts of Y when conditions Z obtain, acts of X will also generate acts of doing Y to the members of S when Z obtains.

Also included in the Domain Knowledge Base are the following:

```
normal_obj(load(X), all_msgs).
speed_proportional_to_size(load(X)).
smaller(X,Y) :- includes(Y,X).
includes(all_msgs, new_msgs).
includes(all_msgs, old_msgs).
```

So SPIRIT believes that the set of all messages is normally loaded in mail, and that loading is a process whose speed is proportional to size of the set of objects loaded. It also has beliefs about what it means for one set to be smaller than another.

Let us now consider four examples. In each of them, Q asks how to load only the new messages, in order to speed up entry time into mail.

Example VI of SPIRIT in Operation

```
Do you have a query?  yes.
Queried Action?  do_to(load(x), new_msgs).
Goal Action?  speed_up(enter_mail).


========================================
 Plan Inference:
=========================================
Query COHERENT
INFERRED EPLAN:
   do_to(load(x),new_msgs)
               and(normal_obj(load(x),all_msgs),smaller(new_ms
gs,all_msgs))
   do_to_fewer(load(x),all_msgs)
               speed_proportional_to_size(load(x))
   speed_up(load(x))
               part_of(load(x),enter_mail)
   speed_up(enter_mail)

=========================================
 Plan Evaluation:
=========================================
The queried action is UNEXECUTABLE.
The goal action is UNEXECUTABLE.
The inferred eplan is WELL-FORMED.
=========================================
Sample Cooperative Response:
=========================================

There is no way for you to do
do_to(load(x),new_msgs) , and there is no way
for you to do speed_up(enter_mail) .

=========================================
=========================================
```

In this case, since both the queried and goal acts are unexecutable, and the inferred eplan is well-formed, there is not much that can be included in the response. For the next example, I will change SPIRIT's beliefs, so that it no longer thinks that loading messages is a process whose speed depends on the number of objects that are to be loaded. This example will demonstrate that SPIRIT can determine ill-formedness in part of an eplan, while other parts are judged well-formed.

Example VII of SPIRIT in Operation

```
| ?- retract(speed_proportional_to_size(load(X))).
```

```
X = _0 .
yes
```

The invocation of SPIRIT has been omitted.

```
Do you have a query?  yes.
Queried Action?  do_to(load(x), new_msgs).
Goal Action?  speed_up(enter_mail).


=========================================
 Plan Inference:
=========================================
Query COHERENT
INFERRED EPLAN:
   do_to(load(x),new_msgs)
               and(normal_obj(load(x),all_msgs),smaller(new_ms
gs,all_msgs))
   do_to_fewer(load(x),all_msgs)
               speed_proportional_to_size(load(x))
   speed_up(load(x))
               part_of(load(x),enter_mail)
   speed_up(enter_mail)

=========================================
 Plan Evaluation:
=========================================
The queried action is UNEXECUTABLE.
The goal action is UNEXECUTABLE.
The inferred eplan is ILL-FORMED:
The following conditions which were
inferred to relate do_to_fewer(load(x),all_msgs)
and speed_up(load(x)) will not be true:
speed_proportional_to_size(load(x))


=========================================
Sample Cooperative Response:
=========================================


There is no way for you to do
```

```
do_to(load(x),new_msgs) , and even if you could
it would not lead to doing speed_up(enter_mail) .
That is because
speed_proportional_to_size(load(x)) will not
be true so you will not do speed_up(load(x))
by doing do_to_fewer(load(x),all_msgs) .
There is no way for you to do speed_up(enter_mail) .

========================================
========================================
```

Of course, SPIRIT can also detect when several parts of the eplan are ill-formed, as demonstrated by the following example. Here SPIRIT's beliefs have again been changed: it no longer believes that loading messages is one of the steps involved in entering mail:

Example VIII of SPIRIT in Operation

```
Do you have a query?  yes.
Queried Action?  do_to(load(x), new_msgs).
Goal Action?  speed_up(enter_mail).


========================================
 Plan Inference:
========================================
Query COHERENT
INFERRED EPLAN:
   do_to(load(x),new_msgs)
               and(normal_obj(load(x),all_msgs),smaller(new_ms
gs,all_msgs))
   do_to_fewer(load(x),all_msgs)
               speed_proportional_to_size(load(x))
   speed_up(load(x))
               part_of(load(x),enter_mail)
   speed_up(enter_mail)

========================================
 Plan Evaluation:
========================================
The queried action is UNEXECUTABLE.
The goal action is UNEXECUTABLE.
The inferred eplan is ILL-FORMED:
The following conditions which were
inferred to relate do_to_fewer(load(x),all_msgs)
and speed_up(load(x)) will not be true:
speed_proportional_to_size(load(x))


The following conditions which were
inferred to relate speed_up(load(x)) and
speed_up(enter_mail) will not be true:
part_of(load(x),enter_mail)


=========================================
Sample Cooperative Response:
=========================================

There is no way for you to do
do_to(load(x),new_msgs) , and even if you could
it would not lead to doing speed_up(enter_mail) .
That is because
speed_proportional_to_size(load(x)) will not
be true so you will not do speed_up(load(x))
by doing do_to_fewer(load(x),all_msgs) . Also,
part_of(load(x),enter_mail) will not be true so you
will not do speed_up(enter_mail) by doing
speed_up(load(x)) . There is no way for you to do
speed_up(enter_mail) .

=========================================
=========================================
```

And finally, in one last example, SPIRIT has been told that there is a way to load only the new messages. With this information, SPIRIT determines that the queried act is executable, but that eplan is ill-formed and the goal act unexecutable, a combination we have not yet seen in the other examples.

Example IX of SPIRIT in Operation

```
Do you have a query?  yes.
```

```
Queried Action?  do_to(load(x), new_msgs).
Goal Action?  speed_up(enter_mail).


========================================
 Plan Inference:
========================================
Query COHERENT
INFERRED EPLAN:
   do_to(load(x),new_msgs)
               and(normal_obj(load(x),all_msgs),smaller(new_ms
gs,all_msgs))
   do_to_fewer(load(x),all_msgs)
               speed_proportional_to_size(load(x))
   speed_up(load(x))
               part_of(load(x),enter_mail)
   speed_up(enter_mail)

========================================
 Plan Evaluation:
========================================
The queried action is EXECUTABLE.  To execute it:

>>> type(set_switch,new_msgs)

The goal action is UNEXECUTABLE.
The inferred eplan is ILL-FORMED:
The following conditions which were
inferred to relate do_to_fewer(load(x),all_msgs)
and speed_up(load(x)) will not be true:
speed_proportional_to_size(load(x))


The following conditions which were
inferred to relate speed_up(load(x)) and
speed_up(enter_mail) will not be true:
part_of(load(x),enter_mail)


========================================
Sample Cooperative Response:
```

```
========================================

You can do do_to(load(x),new_msgs) by doing
type(set_switch,new_msgs) , but it will not lead
to doing speed_up(enter_mail) . That is because
speed_proportional_to_size(load(x)) will not
be true so you will not do speed_up(load(x))
by doing do_to_fewer(load(x),all_msgs) . Also,
part_of(load(x),enter_mail) will not be true so you
will not do speed_up(enter_mail) by doing
speed_up(load(x)) . There is no way for you to do
speed_up(enter_mail) .

========================================
========================================
```

The examples in this section have shown SPIRIT's capabilities. Using the theory of plans and plan inference developed in this thesis, SPIRIT can infer the plans that underlie queries, even when those plans suffer from various sorts of invalidities. It can evaluate an inferred eplan, to determine what invalidities, if any, it has. And it can use the result of that inference and evaluation to produce, in a primitive manner, cooperative responses. Several ways in which SPIRIT could be expanded are obvious. Certain of these depend upon expansions of the theory of plan inference itself. For instance, when the theory is expanded to handle plans with enabling actions more fully, SPIRIT can be likewise expanded. Similarly, when the theory is expanded to handle properly queries that contain information other than a queried and goal act, the range of acceptable input to SPIRIT can be broadened accordingly. Both of these potential expansions are briefly discussed in the next chapter. Also, theories of relevance and salience can usefully augment the current state of SPIRIT, enabling it to produce even more cooperative responses. There are, in addition, at least three potential enhancements of SPIRIT that need only existing theories and technologies. A module that translates natural-language queries into SPIRIT's input representation could be added; a more sophisticated response-generation module could replace the current Cooperative Response Simulator; and SPIRIT's knowledge stores, especially its Domain Knowledge Base, could be enriched, to give it more complete knowledge of the domain. Nonetheless, while it is obviously not yet a useful tool, SPIRIT as it now stands does succeed in demonstrating the theory presented in this work.

CHAPTER VIII
Conclusion

8.1. Summary

This thesis has been about the process of plan inference, specifically domain-plan inference, in question-answering. The computational-linguistics literature of the last decade has recognized the importance of plan inference in conversation; incorporating plan inference capabilities into systems that answer questions has enabled a range of cooperative behaviors. As we have seen, however, these systems have made the assumption that the questioner (or actor) and the respondent (or inferring agent) have identical beliefs about actions in the domain. I argued that this assumption is too strong, and that systems that rely upon it cannot in general produce appropriate responses to queries that arise from invalid plans (although they may be able to respond to queries that arise from particular invalid plans about which they have knowledge). Thus, as one step toward the development of computer systems that are more cooperative in communicating with their users, I developed a model of the plan inference process in question-answering that does not equate the inferring agent's beliefs with the actor's.

Development of this model rested on an analysis of plans as mental phenomena: "having a plan" was analyzed as having a particular configuration of beliefs and intentions. The beliefs and intentions are about the acts that play a role in the plan, where this was defined in terms of two relations over acts, *generation* and *enablement*. Such an analysis differs from the usual treatment in AI, in which plans have been primarily viewed as data structures. But a mental phenomenon view was shown to be especially useful in an account of plan inference: given the analysis of the state of having a plan, the state of believing that another agent has a plan can be directly related to believing that he has the characteristic configuration of beliefs and intentions. Then there is obviously room, in this account, for discrepancies between an agent's own beliefs and the beliefs that she ascribes to an actor when she thinks he has some plan. Such discrepancies were seen to provide an explanation for judgements that a plan is invalid.

I defined several types of invalidities from which a plan may suffer. Within the plan inference model, the role of a discrepant belief within the characteristic configuration is used to determine the type of invalidity that an inferred plan will be to judged have. Since for any inferred plan there may be several discrepant beliefs, a plan can be judged to have several types of invalidities, as well as more than one invalidity of a particular type. The types of invalidities of a plan inferred to underlie a query affect the content of a cooperative response. However, they do not determine it: I argued that the inferred plan, along with any invalidities noted in it, are but two factors contributing to response generation.

Consideration of certain question-answering situations led to a claim that to guarantee a cooperative response, the inferring agent must attempt to ascribe to the questioner more than just a set of beliefs and intentions of suitable configuration: she must also try to find beliefs that explain those beliefs and intentions. The *eplan*, or *explanatory plan*, construct was introduced to capture this requirement, and the job of the inferring agent in question-answering was then cast as an attempt to find an eplan that underlies the questions she is asked. A set of rules was provided that models the process of inferring eplans--that is, of ascribing to another agent plausible beliefs and intentions, along with explanatory beliefs. Finally, SPIRIT--a System for Plan Inference that Reasons about Invalidities Too--was discussed. SPIRIT is a demonstration system implemented to illustrate the model of domain plan inference in question-answering developed in this thesis.

8.2. Future Directions

There are a number of ways in which the research reported on in this thesis can be extended. Perhaps the most obvious involves enriching the plan inference model to handle plans with enabling actions properly. Recall that in the initial analysis of plans, in Chapter 3, I suggested that there are two distinct relations between the intended acts in plans: generation and enablement. In formalizing the analysis of plans in Chapter 4 and in developing the theory of plan inference in Chapters 5 and 6, I restricted my attention largely to what I termed *simple plans*: plans that do not contain any enabling acts. This restriction was necessary to make progress on the problem of inferring potentially invalid plans. However, while a good deal of interesting analysis can be performed even with just simple plans, many plans are *not* simple, and a general model of plan inference should be able to handle them. Preliminary study suggests that the classification of invalidity types is not greatly complicated by the inclusion of plans with enabling actions. Also, the representational tools developed in this thesis are sufficient for defining what it means to have a nonsimple plan and to believe that another agent has a nonsimple plan; they are also sufficient to define what it means for a nonsimple plan to be executable and well-formed. But the strategy for inferring nonsimple plans appears to be significantly more complicated than the strategy for inferring simple plans, and careful consideration will be needed to implement the revised strategy efficiently.

A second avenue for extension of this work is the generalization of the forms of queries that the model can handle. In this thesis, I have focussed on queries in which both the queried and goal act are explicitly indicated. But, as I mentioned in Chapter 6, naturally occurring dialogue includes many examples of queries that do not conform to this requirement. There are, for example, queries in which the goal act is left implicit. It seems fairly obvious how to extend the model to treat these queries: within the plan inference process, the ascription of plausible eplans should terminate when one is found that relates the queried act to some likely goal act. Depending upon the size of the set of likely goal acts at any time, it may also be worthwhile to perform the search unidirectionally, working from the queried act only. It also seems fairly straightforward to extend the model to handle queries in which both a queried act and a goal act are mentioned, but in which it is not indicated which is which: in these cases, two bidirectional inference processes should proceed concurrently, one seeking a path from α to β, and the other, a path from β to α. What will require more work is extending the model to handle queries in which several acts acts are mentioned, such as the following, taken from the mail transcripts:

(47) Q: "[I]s there anyway to dynamically load a file of definitions and commands and have them interpreted by mail? I've wanted to do this so I could have a large file of net addresses and aliases that I would manually load if I wanted to send mail to one of them. I figgure *(sic)* this would decrease the startup time for mail, since I would take all the symbol defs out of my mailinit file."

It will also be important to handle queries that contain other sorts of information, such as constraints on the class of solutions that will be acceptable to the user, for instance, the following two examples, also from the mail transcripts:

(48) Q: "Once I am in a bboard, is there any way to get back into my regular mail short of exiting to DCL and re-entering mail?"

(49) Q: "I always wondered if it was possible to access the bboards directly (i.e. without defining the commands in my mailinit file)"

A third direction for future research involves exploring the consequences of abandoning the Correct-Knowledge and Closed World Assumptions. For the purposes of answering questions and providing advice, making these assumptions is reasonable. But in general they are too strong. In ordinary conversation, when one agent detects a difference between her own beliefs and the beliefs she ascribes to her conversational partner, she is not necessarily entitled to assume that her partner is in error. Sometimes she will instead update her own beliefs; other times she may seek further information before deciding one way or another. When the CKA and CWA are abandoned, the plan inference process does not change, nor does the process by which the inferring agent compares her own beliefs to those that she ascribes to the actor. What does change is the range of possible behaviors that result from detecting a discrepancy in the two sets of beliefs. If the plan inference model developed here is extended to handle these additional behaviors, it will become more of a general model of conversation, and less one that is restricted to question-answering.

Finally, a particularly significant direction for future research appears to be the development of more sophisticated models of belief and intention, and their use in analyses, like the present one, of cooperative conversation. As was mentioned in Chapter 4, there has been a good deal of recent work in both AI and philosophy on the representation of belief. There has been less work on the representation of intention, and even less on integrating and explaining the functional dependencies between the two. But if explaining conversation really does depend in part upon explaining plan inference, and if plans really are most fruitfully conceived of as being mental phenomena, then robust models of conversation, as well as systems that converse cooperatively with their users, will demand precise representations of belief and intention, and considered theories of their interaction. Only after these phenomena have received much more study will it be just as unremarkable to overhear the conversation in Example (1) occur between a human and a computer, as it currently is to overhear it occur between two humans.

APPENDIX A
Axioms for Plan Inference

Belief and Intention

(B1) $BEL(G,p,t) \wedge BEL(G,p \rightarrow q,t) \rightarrow BEL(G,q,t)$

(B2) $BEL(G,p,t) \rightarrow BEL(G, BEL(G,p,t), t)$

(B3) $\neg BEL(G,p,t) \rightarrow BEL(G, \neg BEL(G,p,t), t)$

(B4) $\neg BEL(G,\mathit{false},t)$

(I1) $INT(G,\alpha,t_2,t_1) \wedge BEL(G,GEN(\alpha,\beta,G,t_2),t_1) \rightarrow$
$INT(G,\beta,t_2,t_1)$

(no axiom I2)

(I3) $INT(G, iot(\alpha,\beta), t_2, t_1) \rightarrow$
$INT(G,\alpha,t_2,t_1) \wedge$
$INT(G,\beta,t_2,t_1) \wedge$
$BEL(G,GEN(\alpha,\beta,G,t_2),t_1)$

(Q1 with axioms on Queries)

(Q2) $BEL(R,BEL(Q,p,t_2),t_1) \rightarrow BEL(R,BEL'(Q,p,t_2),t_1)$

(Q3) $BEL(R,INT(Q,\alpha,t_2,t_0),t_1) \rightarrow BEL(R,INT'(Q,\alpha,t_2,t_0),t_1)$

Generation

(C1) $\mathrm{CGEN}(\alpha, \beta, C) \equiv$

(i) $[\forall G_1 \forall t_1[\mathrm{HOLDS}(C,t_1) \wedge \mathrm{OCCURS}(\alpha,G_1,t_1) \rightarrow \mathrm{OCCURS}(\beta,G_1,t_1)] \wedge$

(ii) $\exists G_2 \exists t_2[\mathrm{OCCURS}(\alpha,G_2,t_2) \wedge \neg \mathrm{OCCURS}(\beta,G_2,t_2)] \wedge$

(iii) $\exists G_3 \exists t_3[\mathrm{HOLDS}(C,t_3) \wedge \neg \mathrm{OCCURS}(\beta,G_3,t_3)]]$

(no axioms G1-G3)

(G4) $\mathrm{GEN}(\alpha,\beta,G,t) \equiv$

(i) $\exists C[\ \forall G_1 \forall t_1(\mathrm{HOLDS}(C,t_1) \wedge \mathrm{OCCURS}(\alpha,G_1,t_1) \rightarrow \mathrm{OCCURS}(\beta,G_1,t_1)) \wedge$

(ii) $\exists G_2 \exists t_2(\mathrm{OCCURS}(\alpha,G_2,t_2) \wedge \neg \mathrm{OCCURS}(\beta,G_2,t_2)) \wedge$

(iii) $\exists G_3 \exists t_3(\mathrm{HOLDS}(C,t_3) \wedge \neg \mathrm{OCCURS}(\beta,G_3,t_3)) \wedge$

(iv) $\mathrm{HOLDS}(C,t)]$

Executability

(E1) $\mathrm{EXEC}(\beta,G,t) \equiv$

(i) $[\mathrm{BASIC}(\beta,G,t) \wedge \mathrm{SC}(\beta,G,t)] \vee$

(ii) $\exists \alpha_1 \ldots \exists \alpha_n[\ \mathrm{BASIC}(\alpha_1,G,t) \wedge \ldots \wedge (\mathrm{BASIC}(\alpha_n,G,t) \wedge$
$\mathrm{SC}(\alpha_1,G,t) \wedge \ldots \wedge \mathrm{SC}(\alpha_n,G,t) \wedge$
$\mathrm{GEN}(\alpha_1; \ldots ;\alpha_n,\beta,G,t)]$

Plans

(P1) SIMPLE-PLAN($G,\alpha_n,[\alpha_1, \ldots,\alpha_{n-1}],t_2,t_1$) $\equiv$

(i) BEL(G,EXEC($\alpha_i,G,t_2),t_1$) for $i=1,\ldots,n-1$ $\wedge$

(ii) BEL(G,GEN($\alpha_i,\alpha_{i+1},G,t_2),t_1$) for $i=1,\ldots,n-1$ $\wedge$

(iii) INT(G,α_i,t_2,t_1) for $i=1,\ldots,n-1$ $\wedge$

(iv) INT(G,iot($\alpha_i,\alpha_{i+1}),t_2,t_1$) for $i=1,\ldots,n-1$

(P2) BEL(R, EPLAN($Q,\alpha_n,[\alpha_1, \ldots,\alpha_{n-1}]$,
$[\rho_1, \ldots,\rho_{n-1}],t_2,t_1),t_1$) $\equiv$

(i) BEL(R,BEL(Q,EXEC($\alpha_i,Q,t_2),t_1), t_1$) for $i=1,\ldots,n-1$ $\wedge$

(ii) BEL(R,BEL(Q,GEN($\alpha_i,\alpha_{i+1},Q,t_2),t_1),t_1$) for $i=1,\ldots,n-1$ $\wedge$

(iii) BEL(R,INT($Q,\alpha_i,t_2,t_1),t_1$) for $i=1,\ldots,n-1$ $\wedge$

(iv) BEL(R,INT(Q,iot($\alpha_i,\alpha_{i+1}),t_2,t_1),t_1$) for $i=1,\ldots,n-1$ $\wedge$

(v) BEL(R, BEL(Q, ρ_i, t_1), t_1) for $i=1,\ldots,n-1$
where each ρ_i is CGEN($\alpha_i,\alpha_{i+1},C_i$) $\wedge$ HOLDS(C_i,t_2)

(P3) BEL(R, EPLAN'($Q,\alpha_n,[\alpha_1, \ldots,\alpha_{n-1}]$,
$[\rho_1, \ldots,\rho_{n-1}],t_2,t_1),t_1$) $\equiv$

(i) BEL(R,BEL'(Q,EXEC($\alpha_i,Q,t_2),t_1), t_1$) for $i=1,\ldots,n-1$ $\wedge$

(ii) BEL(R,BEL'(Q,GEN($\alpha_i,\alpha_{i+1},Q,t_2),t_1),t_1$) for $i=1,\ldots,n-1$ $\wedge$

(iii) BEL(R,INT'($Q,\alpha_i,t_2,t_1),t_1$) for $i=1,\ldots,n-1$ $\wedge$

(iv) BEL(R,INT'(Q,iot($\alpha_i,\alpha_{i+1}),t_2,t_1),t_1$) for $i=1,\ldots,n-1$ $\wedge$

(v) BEL(R, BEL'(Q, ρ_i, t_1), t_1) for $i=1,\ldots,n-1$
where each ρ_i is CGEN($\alpha_i,\alpha_{i+1},C_i$) $\wedge$ HOLDS(C_i,t_2)

Validity Judgements

(J1) $BEL(G, UNEX(\alpha,H,t_2), t_1) \equiv \neg BEL(G, EXEC(\alpha,H,t_2), t_1)$

(J2) $BEL(G, \neg EXEC(\alpha,H,t_2), t_1) \rightarrow BEL(G, UNEX(\alpha,H,t_2), t_1)$

(J3) $BEL(R, ILL\text{-}FORMED(Q, \alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1), t_1)$

$\equiv$

$BEL(R, EPLAN(Q, \beta,[\alpha_1,\ldots,\alpha_{n-1}],[\rho_1, \ldots, \rho_{n-1}], t_2, t_1), t_1) \wedge$
$[\neg BEL(R, GEN(\alpha_i,\alpha_{i+1},Q,t_2), t_1)]$ for some $i=0,\ldots,n-1$

(J4) $BEL(R, INCOHERENT(Q, \theta),t_1) \equiv$
$\neg \exists \alpha_1 \ldots \exists \alpha_n \exists \rho_1 \ldots \exists \rho_{n-1} [BEL(R,EPLAN(Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],$
$[\rho_1,\ldots,\rho_{n-1}],t_2,t_1), t_1) \wedge$
$BEL(R,UNDERLYING\text{-}EPLAN(\theta,Q,\alpha_n,[\alpha_1,\ldots,\alpha_{n-1}],$
$[\rho_1,\ldots,\rho_{n-1}],t_2,t_1), t_1)]$

Queries

(Q1) $BEL(R,QUERY(Q,\theta,\alpha,\beta,t_2,t_0),t_1)$

$\rightarrow$

$BEL(R,EPLAN(Q,\alpha,[],[],t_2,t_1),t_1) \wedge$
$BEL(R,EPLAN(Q,\beta,[],[],t_2,t_1),t_1)$

(Q2 and Q3 with axioms on Belief and Intention; no Q4)

(Q5) $BEL(R,EPLAN'(Q,\beta,[\alpha,\ldots,\alpha_{n-1}],[\rho_1,\ldots\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, QUERY(Q,\theta,\alpha,\beta,t_2,t_0), t_1)$

$\rightarrow$

$BEL(R,UNDERLYING\text{-}EPLAN(\theta,Q,\beta,$
$[\alpha,\alpha_1,\ldots,\alpha_{n-1}],[\rho_1,\ldots,\rho_{n-1}],t_2,t_1),t_1)$

Plan Inference

(PI1) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, CGEN(\alpha_n,\gamma,C), t_1)$
$\rightarrow$
$BEL(R,EPLAN'(Q,\gamma,[\alpha_1,...,\alpha_n],[\rho_1,...,\rho_n],t_2,t_1),t_1)$
where $\rho_n = CGEN(\alpha_n,\gamma,C) \wedge HOLDS(C,t_2)$

(PI2) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, CGEN(\gamma,\alpha_1,C), t_1)$
$\rightarrow$
$BEL(R,EPLAN'(Q,\alpha_n,[\gamma,\alpha_1,...,\alpha_{n-1}],[\rho_0,\rho_1,...,\rho_{n-1}],t_2,t_1),t_1)$
where $\rho_0 = CGEN(\gamma,\alpha_1,C) \wedge HOLDS(C,t_2)$

(PI3) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1), t_1) \wedge$
$BEL(R,CGEN(\alpha_n,\gamma,C_1 \wedge ...\wedge C_n), t_1)$
$\rightarrow$
$BEL(R,EPLAN'(Q,\gamma,[\alpha_1,...,\alpha_n],[\rho_1,...,\rho_n],t_2,t_1), t_1)$
where $\rho_n=CGEN(\alpha_n,\gamma,C_1 \wedge ... \wedge C_{i-1} \wedge C_{i+1} \wedge ... \wedge C_n) \wedge$
$HOLDS(C_1 \wedge ...\wedge C_{i-1} \wedge C_{i+1} \wedge ... \wedge C_n, t_2)$

(PI4) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1), t_1) \wedge$
$BEL(R,CGEN(\gamma,\alpha_1,C_1 \wedge ...\wedge C_n), t_1)$
$\rightarrow$
$BEL(R,EPLAN'(Q,\alpha_n,[\gamma,\alpha_1,...,\alpha_{n-1}],[\rho_0,\rho_1,...,\rho_n],t_2,t_1), t_1)$
where $\rho_0=CGEN(\gamma,\alpha_1,C_1 \wedge ... \wedge C_{i-1} \wedge C_{i+1} \wedge ... \wedge C_n) \wedge$
$HOLDS(C_1 \wedge ...\wedge C_{i-1} \wedge C_{i+1} \wedge ...\wedge C_n, t_2)$

(PI5) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, SIMILAR(\alpha_n,\delta),t_1) \wedge$
$BEL(R, CGEN(\delta,\gamma,C), t_1)$
$\rightarrow$
$BEL(R,EPLAN'(Q,\gamma,[\alpha_1,...,\alpha_n],[\rho_1,...,\rho_n],t_2,t_1),t_1)$
where $\rho_n=CGEN(\alpha_n,\gamma,C) \wedge HOLDS(C,t_2)$

(PI6) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, SIMILAR(\alpha_1,\delta),t_1) \wedge$

$BEL(R, CGEN(\gamma,\delta,C), t_1)$

$\rightarrow$

$BEL(R,EPLAN'(Q,\alpha_n,[\gamma,\alpha_1,...,\alpha_{n-1}],$
$[\rho_0,\rho_1,...,\rho_{n-1}],t_2,t_1),t_1)$

where ρ_0=$CGEN(\gamma,\alpha_1,C) \wedge HOLDS(C,t_2)$

(PI7) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}], [\rho_1,...,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R,CGEN(\alpha_n,\delta,C), t_1) \wedge$
$BEL(R, SIMILAR(\delta, \gamma),t_1)$

$\rightarrow$

$BEL(R,EPLAN'(Q,\gamma,[\alpha_1,...,\alpha_n], [\rho_1,...,\rho_n],t_2,t_1),t_1)$

where ρ_n=$CGEN(\alpha_n,\gamma,C) \wedge HOLDS(C,t_2)$

(PI8) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R, CGEN(\delta,\gamma,C), t_1) \wedge$
$BEL(R, SIMILAR(\delta,\gamma), t_1)$

$\rightarrow$

$BEL(R,EPLAN'(Q,\alpha_n,[\gamma,\alpha_1,...,\alpha_{n-1}],[\rho_0,\rho_1,...,\rho_{n-1}],t_2,t_1),t_1)$

where ρ_0=$CGEN(\gamma,\alpha_1,C) \wedge HOLDS(C,t_2)$

(PI9) $BEL(R,EPLAN'(Q,\alpha_n,[\alpha_1,...,\alpha_{n-1}],[\rho_1,...,\rho_{n-1}],t_2,t_1),t_1) \wedge$
$BEL(R,USER\text{-}TYPE(Q,S),t_1) \wedge$
$BEL(R,TYPICAL\text{-}MISCONCEPTION(S,CGEN(\alpha_n,\gamma,C)),t_1)$

$\rightarrow$

$BEL(R,EPLAN'(Q,\gamma,[\alpha_1,...,\alpha_n], [\rho_1,...,\rho_n],t_2,t_1),t_1)$

where ρ_n=$CGEN(\alpha_n,\gamma,C) \wedge HOLDS(C,t_2)$

(PI10 is the symmetric partner of PI9)

APPENDIX B
Compendium of Examples

This appendix provides a cross-reference listing for all the dialogue examples discussed in the text of the thesis. All the variations of each example are grouped together. For each set of variations, I first provide the original, naturally occurring example that motivated the set, identifying its source.[74] I then provide a list of all the examples in the text that are variations of the naturally occurring example.

Q: "I called Kathy 10 times at least. The phone is always busy! Do I have the right number? Maybe you can give it to me again."
R: "I don't know why the number at the hospital is always busy, but in any case, Kathy's at home now. Her number there is (xxx)xxx-xxxx."

Source: computer-mail message collected in October, 1984.

Example #	Page
1	1
8	29
9	29
25	100
29	103
31	103
34	105
37	107

---

[74] The second set of examples was motivated by a constructed example, as listed below.

Q: "How can I protect my mail messages? I don't want the system manager to be able to read them."

R: "Even if you set the protections, the system manager can override them."

Source: constructed example.

Example #	Page
2	3
3	3
4	3
5	4
6	4
7	4
27	102
28	102
38	108
39	112
40	112
41	113

Q: "Is there any way to tell a bboard to load only new messages (I would like to speed up entry into infomac)?"

R: "As far as loading only new messages, Mail cannot do this because it parses the entire message file whenever you start up. We considered using a special format file for Mail so that access to messages could be optimized but decided that it was more important that the mail file be a normal text file which could be edited with Emacs. I will clean out the infomac bboard (moving old messages to an archive file) so that it will load faster."

Source: Penn mail transcripts, Dialogue 1-2, collected July, 1984.

Example #	Page
10	30
11	30
12	31
15	93
16	93
17	93
18	93
19	93
20	96
26	101
30	103
32	104
43	124

Q: "How can I scroll up and down in mail? I usually forget what number mail I'm looking at and if the info I need is off the screen, I can't get to it easily."

R: "Unless your terminal lets you scroll back and forth (which it should---ask Ira about this) Mail can't do anything about scrolling for you. However, the last message you were working with (for example, reading) is called the "current message". Mail lets you refer to the current message without knowing its number. In place of a number, you type a period (".") in the Mail command to refer to the current message. For example, to list the current message you could type "list ." (without the "s of course). Since listing a message prints out its number, that's one easy way to find out what number message you were using.

However, you may not ever need to know the message number. You could delete the current message by typing "delete .", read it again with "type .", print it with "print .", answer it with "answer ." and so on. The "." works just like a message number. I usually don't use message numbers at all when I read my mail. When you come into mail the first new message is the "current message" so you can say "type ." to get things going. The "next" command types out the next message following the current one. Once the next message is typed out it becomes the new current message. Thus, you could read your first new message, delete it, read the next message, print it, and then delete it as follows:

```
$ mail
  + 5) blah blah blah...
  + 6) blahh blahh blahh...
>> type .
- 5 -
   this is message five...
>> delete .
1 message deleted
>> next
- 6 -
   this is number six....
>> print .
1 message printed (or whatever)
>> delete .
```

```
1 message deleted.
>>
```

Hope that answers the question."

Q: "Yes, that answers many questions, much of which I already knew. What I was thinking of was things like lists or schedules which are sent through the mail. Say, I'm looking at the fall schedule of graduate courses and someone asks who's teaching CIS640; then they want info about 523, which by this time is off my screen. It's a pain to go through mail commands when I would prefer to scroll. But thanks for your time.

Source: Penn mail transcripts, Dialogue 2, collected August, 1984.

Example #	Page
13	31
21	97
44	124

Q: "Once I am in a bboard, is there any way to get back into my regular mail short of exitting to DCL and re-entering MAIL ?"

R: "Being in a bboard is sort of like being in an archive mail file except that Mail knows that you cannot permanently modify the file. So you can get back to your regular mail file by typing "use main" (I don't know if you knew that you could do this from an archive file.)"

Source: Penn mail transcripts,Dialogue 1-1,collected July, 1984.

Example #	page
48	185

Q: "is there anyway to dynamically load a file of definitions and commands and have them interpreted by mail? I've wanted to do this so I could have a large file of net addresses and aliases that I would manually load if I wanted to send mail to one of them. I figgure *(sic)* this would decrease the startup time for mail, since I would take all ths symbol defs out of my mailinit file."

R: "There is currently no way to dynamically load a file of definitions. I've considered this but wasn't sure how useful it would be. Perhaps I'll put it in if I have time."

Source: Penn mail transcripts, Dialogue 7, collected August, 1984.

Example #	Page
14	32
22	97
47	184

Q: "Is there a way to tell mail that I'd like to deliver a message after some particular time and/or date? I couldn't see anything in the documentation that hinted at this, maybe I just missed it ."

R: "That's an easy one---no. I can't really see any easy way to do this either (and guarantee it). If you really wanted to do that you could submit a batch job to run after a certain time and have it send the message."

Source: Penn mail transcripts, Dialogue 10, collected August, 1984.

Example #	Page
23	98

Q: "Re your request for questions about the mail system. Is there some way to mark messages other than to actually look at them or print them? Thanks!"

R: "I don't quite know what you mean by 'mark a message'? Do you mean, make mail think you've read the message?"

Q: "Yes, I meant to be able to mark a message as having been read without needing to actual *(sic)* do so."

R: "There's no neat way to mark a message "read" without reading it. It is possible to delete a message without reading it. Also, I don't know if you're aware of this but once you get into Mail, whether or not you read the messages it tells you are new you'll never be notified about them again outside of Mail (like when you log in). If you do want to delete messages without reading them you probably would want to put the following line into you "mailinit" file: >>>set no warn-on-deletion. This tells Mail not to ask for confirmation before deleting a new message.
I hope this seems to answer your question. If not, I'll be happy to try again.

Source: Penn mail transcripts, Dialogue 11, collected August, 1984.

Example #	Page
24	98

Q: "Is there any way to append a mail message to the end of an existing file, instead of having it written to a new file using COPY? Sometimes I like to store away a series of messages in a single file, and it would be convenient to be able to do this from within mail. The messages mail not all be in my mail file at the same time."

R: " 'copy' does append to a file if it already exists for exactly the reason you mentioned. When I store away the questions I get in mailquest.msg I do a 'move' (which is just like a copy except that it deletes the message after copying)."

Source: Penn mail transcripts, Dialogue 13, collected August, 1984.

Example #	Page
45	125

Q: "I always wondered if it was possible to access the bboards directly (i.e. without defining the commands in my mailinit file) from MAIL."

R: "You can access bboard from within mail by typing 'bboard colloq' or 'bb general' or whatever just as you can from the $ prompt. Since the bboard system is really just part of MAIL you can also do this from another bboard. If you like, you could define the 'bbg', 'bba', 'bbc' symbols in your mailinit to work just like the ones you have defined in your login.com. (I'll tell you how if you want to do this)."

Source: Penn mail transcripts, Dialogue 5-1, collected August, 1984.

Example #	Page
49	185

Q: "I'd like to take advantage of the discount offer you sent me, and subscribe to your magazine. How can I specify that I don't want the magazines to starting arriving until October, since I'm going to be moving before then?"

R: "Sorry, you can't do that. To get the discount rate you have to take the subscription before August 31st, and once you take it, the magazines will start arriving automatically within 3 weeks."

Source: telephone conversation, collected July 1985.

Example #	Page
33	104

Q: "...I'll take Fundamental Algorithms next semester, so ... I thought I'll take Operating Systems or some other course this semester."

R: "You can do that except Fundamental Algorithms is a 3000 level prerequisite so usually it makes sense to take this course before the electives."

Source: McKeown's course registration transcripts, collected August 1984.

Example #	Page
36	106

Q: "Track eleven?"
R: "Track eleven?"
Q: "Yeah."
R: "Ah, you work around here?"
Q: "No."
R: "What do you want to go to track eleven for?"
Q: "There's an employment office there. CP."
R: "CP Employment is behind gate nine. Back there."
Q: "The gate nine."
R: "Behind gate nine."

Source: Horrigan train-station dialogues, Dialogue 3, collected 1977.

Example #	Page
35	106
42	114

Q: "Going to Stratford, what gate would it be?"
R: "Which one is that?"
Q: "Two fifteen, I think is the--"
R: "Yeah. Two fifteen. Gate number eight."
Q: "Number eight?"
R: "Right."
. . .

Source: Horrigan tran-station dialogues, Dialogue 2, collected 1977.

Example #	Page
46	125

References

[Allen 79] Allen, James F. *A Plan Based Approach to Speech Act Recognition.* Technical Report TR 121/79, University of Toronto, 1979.

[Allen 80] Allen, James F. and C. Raymond Perrault. Analyzing Intention in Utterances. *Artificial Intelligence* 15:143-178, 1980.

[Allen 83a] Allen, James F. Recognizing Intentions from Natural Language Utterances. In Brady, Michael and Robert C. Berwick (editor), *Computational Models of Discourse*, pages 107-166. MIT Press, Cambridge, Ma., 1983.

[Allen 83b] Allen, James F. ARGOT: A System Overview. *International Journal of Computers and Mathematics* 9:97-110, 1983.

[Allen 83c] Allen, James F. and Johannes A. Koomen. Planning Using a Temporal World Model. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 741-717. Karlsruhe, West Germany, 1983.

[Allen 84] Allen, James F. Towards a General Theory of Action and Time. *Artificial Intelligence* 23(2):123-154, 1984.

[Anscombe 58] Anscombe, G. E. M. *Intention.* Cornell University Press, Ithaca, N.Y., 1958.

[Appelt 85] Appelt, Douglas E. *Planning English Sentences.* Cambridge Univerisity Press, New York, 1985.

[Audi 73] Audi, Robert. Intending. *Journal of Philosophy* 70:387-403, 1973.

[Austin 75] Austin, J. L. *How To Do Things With Words.* Harvard University Press, Cambridge, Ma., 1975. Second Edition.

[Beardsley 78] Beardsley, Monroe. Intending. In Goldman, Alvin I. and Jaegwon Kim (editor), *Values and Morals: Essays in Honor of William Frank.* Reidel, Dordrecht, 1978.

[Bratman 78] Bratman, Michael. Individuation and Action. *Philosophical Studies* 33:367-375, 1978.

[Bratman 83] Bratman, Michael. Taking Plans Seriously. *Social Theory and Practice* 9:271-287, 1983.

[Bratman 86] Bratman, Michael. *Intention, Plans and Practical Reason.* Harvard University Press, Cambridge, Ma., 1986. Forthcoming.

[Brown 78] Brown, John Seely and Richard R. Burton. Diagnostic Models for Procedural Bugs in Basic Mathematical Skills. *Cognitive Science* 2:155-192, 1978.

[Bruce 75] Bruce, Bertram. *Belief Systems and Language Understanding.* Technical Report 21, Bolt Beranek And Newman, 1975.

[Bruce 78] Bruce, Bertram and Denis Newman. Interacting Plans. *Cognitive Science* 2:195-233, 1978.

[Carberry 85] Carberry, M. Sandra. *Pragmatic Modeling in Information System Interfaces.* PhD thesis, University of Delaware, 1985.

[Chisholm 76] Chisholm, Roderick. *Person and Object.* Open Court, LaSalle, Ill., 1976.

[Cohen 79] Cohen, Philip R. and C. Raymond Perrault. Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science* 3:177-212, 1979.

[Cohen 80] Cohen, Philip R. and Hector J. Levesque. Speech Acts and the Recognition of Shared Plans. In *Proceedings of the Third Biennial Conference,* pages 263-271. Canadian Society for Computational Studies of Intelligence, Victoria, B.C., 1980.

[Cohen 82] Cohen,P., C.R. Perrault And J. Allen. Beyond Question Answering. In Lehnert, W. And Ringle, M. (editor), *Strategies for Natural Language Processing,* pages 245-275. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1982.

[Cohen 85] Cohen, Philip R., and Hector J. Levesque. Speech Acts and Rationality. In *Proceedings of the 23rd Conference of the Association for Computational Linguistics,* pages 49-59. Stanford, Ca., 1985.

[CohenP 82] Cohen, Paul R. and Edward Feigenbaum. *The Handbook of Artificial Intelligence.* HeurisTech Press, Stanford, Ca., 1982.

[Danto 63] Danto, Arthur. What We Can Do. *Journal of Philosophy* LX:435-445, 1963.

[Danto 65] Danto, Arthur. Basic Actions. *American Philosophical Quarterly* II:141-148, 1965.

[Davidson 80a] Davidson, Donald. Actions, Reasons, and Causes. In Donald Davidson (editor), *Essays on Actions and Events*, pages 3-20. Clarendon Press, New York, 1980.

[Davidson 80b] Davidson, Donald. Intending. In Donald Davidson (editor), *Essays on Actions and Events*, pages 83-102. Clarendon Press, New York, 1980.

[Davidson 80c] Davidson, Donald. Agency. In Donald Davidson (editor), *Essays on Actions and Events*, pages 43-61. Clarendon Press, New York, 1980.

[Davis 79] Davis, Laurence H. *Theory of Action*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1979.

[Dennett 84] Dennett, Daniel. Cognitive Wheels: The Frame Problem of AI. In C. Hookway (editor), *Minds, Machines, and Evolution*. Cambridge University Press, New York, 1984.

[Eccli 83] Eccli, Eugene, David Klein, and Peter Natali. Dialogues with the Wharton Terminal-Room Consultant. 1983.Unpublished.

[Fikes 71] Fikes, R. E. and Nils J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2:189-208, 1971.

[Finin 82] Finin, Tim and Jeff Schraeger. An Expert System that Volunteers Advice. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 339-340. Pittsburgh, Pa., 1982.

[Fischer 85] Fischer, Gerherd, Andreas Lemke, and Thomas Schwab. Knowldege-Based Help Systems. In *Proceedings of the CHI'85 Conference on Human Factors in Computing Systems*, pages 161-167". ACM SIGCHI, San Francisco,. 1985.

[Genesereth 79] Genesereth, M.R. The Role of Plans in Automated Consultation. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pages 311-319. Tokyo, 1979.

[Georgeff 85] Georgeff, Michael, Amy Lansky, and Pierre Bessiere. A Procedural Logic. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 516-523. Los Angeles, 1985.

[Goldman 70] Goldman, Alvin I. *A Theory of Human Action*. Prentice-Hall, Englewood Cliffs, N.J., 1970.

[Grice 68] Grice, H. P. Utterer's meaning, sentence-meaning, and word-meaning. *Foundations of Language* 4, 1968.

[Grice 69] Grice, H. P. Utterer's Meaning and intentions. *Philosophical Review* 68(2):147-177, 1969. Originally delivered at Oberlin College, April, 1968.

[Grice 71] Grice, H. Paul. Intention and Uncertainty. *Proceedings of the British Academy* 57:263-279, 1971.

[Grice 75] Grice, H. P. Logic and Conversation. In Cole, P. and Morgan, J.L. (editors), *Syntax and Semantics.* Academic Press, New York, 1975. From 1967 lectures.

[Grosz 77] Grosz, Barbara J. *The Representation and Use of Focus in Dialogue Understanding.* Technical Report 151, SRI International, 1977.

[Halpern 85] Halpern, Joseph Y., and Yoram Moses. A Guide to the Modal Logics of Knowledge and Belief. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 480-490. Los Angeles, 1985.

[Hintikka 62] Hintikka, J. *Knowledge and Belief.* Cornell University Press, Ithaca, N.Y., 1962.

[Horrigan 77] Horrigan, M.K. *Modelling simple dialogs.* Technical Report TR 108, Dept. of Computer Science, University of Toronto, 1977.

[Israel 85] Israel, David. *A Weak Logic of Knowledge and Belief: Epistemic and Doxastic Logic for the Yuppie Generation.* Technical Report 359, SRI International, 1985.

[Joshi 82] Joshi, Aravind K. Mutual Belief in Question Answering Systems. In N. Smith (editor), *Mutual Belief.* Academic Press, New York, 1982.

[Joshi 84] Joshi, Aravind K., Bonnie Webber, and Ralph Weischedel. Living Up to Expectations: Computing Expert Responses. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 169-175. Austin, Tx., 1984.

[Kautz 82] Kautz, Henry A. *A First-Order Dynamic Logic for Planning.* Technical Report CSRG-144, University of Toronto, 1982.

[Kautz 85] Kautz, Henry A. *Toward a Theory of Plan Recognition.* Technical Report TR162, University of Rochester, 1985.

[Konolige 84] Konolige, Kurt. Belief and Incompleteness. In Hobbs, J. R. and R. C. Moore (editor), *Formal Theories of the Commonsense World*, pages 359-403. Ablex Publishing Corp., Norwood, N.J., 1984.

[Lansky 85] Lansky, Amy L. *Behavioral Specification and Planning for Multiagent Domains.* Technical Report 360, SRI International, 1985.

[Lenzen 78] Lenzen, W. Recent Work in Epistemic Logic. *Acta Philosophica Fennica* 30:1-219, 1978.

[Litman 84] Litman, Diane and James Allen. *A Plan Recognition Model for Subdialogues in Conversation.* Technical Report TR 141, University of Rochester, 1984.

[Litman 85] Litman, Diane. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues.* PhD thesis, University of Rochester, 1985.

[Mark 81] Mark, William. Representation and Inference in the Consul System. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence,* pages 375-381. Vancouver, B.C., 1981.

[McCarthy 69] McCarthy, J. and P. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Mitchie (editor), *Machine Intelligence, Vol. 4,* pages 463-502. American Elsevier, New York, 1969.

[McCue 83] McCue, Daniel and Victor Lesser. *Focusing and Constraint Management in Intelligent Interface Design.* Technical Report COINS 83-36, Univeristy of Massachusetts, Amherst, 1983.

[McDermott 82] McDermott, Drew. A Temporal Logic for Reasoning About Processes and Plans. *Cognitive Science* 6:101-155, 1982.

[McKeown 85] McKeown, Kathleen R. *Text Generation.* Cambridge University Press, New York, 1985.

[Moore 84] Moore, Robert C. A Formal Theory of Knowledge and Action. In Hobbs, J. R. and R. C. Moore (editor), *Formal Theories of the Commonsense World*, pages 319-358. Ablex Publishing Corp., Norwood, N.J., 1984.

[Mourelatos 78] Mourelatos, A.P.D. Events, processes, and states. *Linguistics and Philosophy* 2:415-434, 1978.

[Nilsson 80] Nilsson, Nils J. *Principles of Artificial Intelligence.* Tioga Publishing Co., Palo Alto, Ca., 1980.

[Pelavin 86] Pelavin, Richard. A Formal Logic for Planning with a Partial Description of the Future. 1986. Forthcoming University of Rochester Doctoral Dissertation.

[Perrault 80] Perrault, C. Raymond and James F. Allen. A Plan-Based Analysis of Indirect Speech Acts. *American Journal of Computational Linguistics* 6:167-182, 1980.

[Pollack 82] Pollack, Martha, Julia Hirschberg and Bonnie Webber. User Participation in the Reasoning Processes of Expert Systems. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 358-361. 1982. A longer version appears as University of Pennsylvania Technical Report MS-CIS-82-9.

[Putnam 79] Putnam, Hilary. On Properties. In *Mathematics, Matter and Method: Philosophical Papers, 2nd Ed.*, pages 305-322. Cambridge University Press, New York, 1979.

[Reiter 78] Reiter, Ray. On Closed World Data Bases. In Gallaire, H. and J. Minker (editor), *Logic and Data Bases*, pages 55-76. Plenum Press, New York, 1978.

[Rosenschein 81] Rosenschein, Stanley J. Plan Synthesis: A Logical Perspective. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence.* Vancouver, B.C., 1981.

[Sacerdoti 77] Sacerdoti, Earl D. *A Structure for Plans and Behavior.* American Elsevier, New York, 1977.

[Schank 77] Schank, Roger and R. Abelson. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[Schmidt 78] Schmidt, C.F., N.S. Sridharan and J.L. Goodson. The plan recognition problem: an intersection of artificial intelligence and psychology. *Artificial Intelligence* 10:45-83, 1978.

[Sidner 81] Sidner, Candace L. and David Israel. Recognizing Intended Meaning and Speaker's Plans. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 203-208. Vancouver, B.C., 1981.

[Sidner 83] Sidner, Candace L. What the speaker means: the recognition of speakers' plans in discourse. *International Journal of Computers and Mathematics* 9:71-82, 1983.

[Sidner 85] Sidner, Candace L. Plan Parsing for Intended Response Recognition in Discourse. *Computational Intelligence* 1(1), 1985.

[Stevens 79] Stevens, Albert, Allan Collins and Sarah E. Goldin. Misconceptions in Student's Understanding. *Intl. J. Man-Machine Studies* 11:145-156, 1979.

[Waldinger 81] Waldinger, Richard. Achieving Several Goals Simultaneously. In Webber, Bonnie Lynn and Nils J. Nilsson (editor), *Readings in Artificial Intelligence*, pages 250-271. Tioga Press, Palo Alto, Ca., 1981.

[Wilensky 83] Wilensky, Robert. *Planning and Understanding: A Computational Approach to Human Reasoning.* Addison-Wesley, Reading, Ma., 1983.

[Wilensky 84] Wilensky, Robert, Yigal Arens, and David Chin. Talking to UNIX in English: An Overview of UC. *Communications of the ACM* 27(6):574-593, 1984.

[Wilkins 83] Wilkins, David E. Representation in a Domain-Independent Planner. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 733-740. 1983.

[Woolf 83] Woolf, Beverly and David McDonald. Human-Computer Discourse in the Design of a PASCAL Tutor. In *Proceedings of the CHI'83 Conference on Human Factors in Computing Systems.* ACM SIGCHI, 1983.

INDEX

Act-token 46, 58
Act-type 46, 50, 58, 68
Act-type constructor function 50, 75, 143, 149, 152
Actions
  basic 67, 94, 157
  representation of 17, 19, 46
  unexecutable 91, 111, 116, 121
Actor 2, 10
Appropriate-Query Assumption 26, 28

Basic actions 67, 94, 157
Belief
  axioms for 73
  plausible 130
  relation to intention 43, 185

Closed-World Assumption 25, 88, 185
Conditional generation 58
  definition of 59
Cooperation 10, 12, 90
Correct-Knowledge Assumption 24, 88, 185

Enablement 39, 83, 182
Executability 45, 111, 156
  definition of 72
Explanatory belief 110, 111, 127, 132, 141, 143
Explanatory plans 8, 111, 183
  definition of 111
  plausible 131

Forking heuristic 11

Generation 40, 45, 52, 83, 182
  causal 56
  definition of 57, 60
  properties of 53
  sortal 56
Generation-enabling conditions 58, 59, 143
Generational pair 55

Identity thesis 52
Ill-formed plans 98, 111, 116
  definition of 101
  relation to unexecuable acts 102
Incoherence 107
  definition of 114
Inferring agent 2, 10
Intended recognition 11
Intention
  axioms for 74
  plausible 130
  relation to belief 43, 185

Keyhole recognition 10

Naturally occurring dialogues 5, 31, 96, 104, 114
NOAH 17, 20, 34
Noncooperation
  active 10, 12
  passive 10, 12
Nonmonotonic inference 128

Occurrence 46, 50
Operator 18
Operator library 18

Plan inference 9, 19, 88, 111, 118, 126, 153, 182
  behavior enabled by 2, 28, 29
  standard rules for 20
Plan inference rules 132, 134, 142, 144, 145
Plans
  belief requirements for 34, 41, 79
  commonsense definition of 44
  communicative 13
  data-structure view of 33, 83, 182
  domain 13
  explanatory 8, 111, 183
  ill-formed 98, 111, 116
  intention requirements for 34, 42, 79
  invalid 1, 87, 182

mental phenomenon view of 33, 35, 83, 182
partial 37, 128
representation of 17, 19, 78, 83
simple 7, 45
Precondition-action rule 2, 22
Principle of Parsimony 24, 145
Prolog 8, 78, 147, 154
Psychological reality 5

Queries 29
How vs. Wh 125
representation of 123
Question-answering 2, 5, 13, 26, 89, 100, 111, 115, 118, 182

Response strategies 89, 115, 159

Simple plans 7, 45
definition of 78
Single-branch assumption 11
SPIRIT 8, 89, 91, 123, 132, 144, 147
components of 149, 152
Standard conditions 69
STRIPS 17

Unexecutable acts 91, 111, 116, 121
definition of 94
relation to ill-formed plans 102

Valid-Plan Assumption 25, 28
Validity properties 114