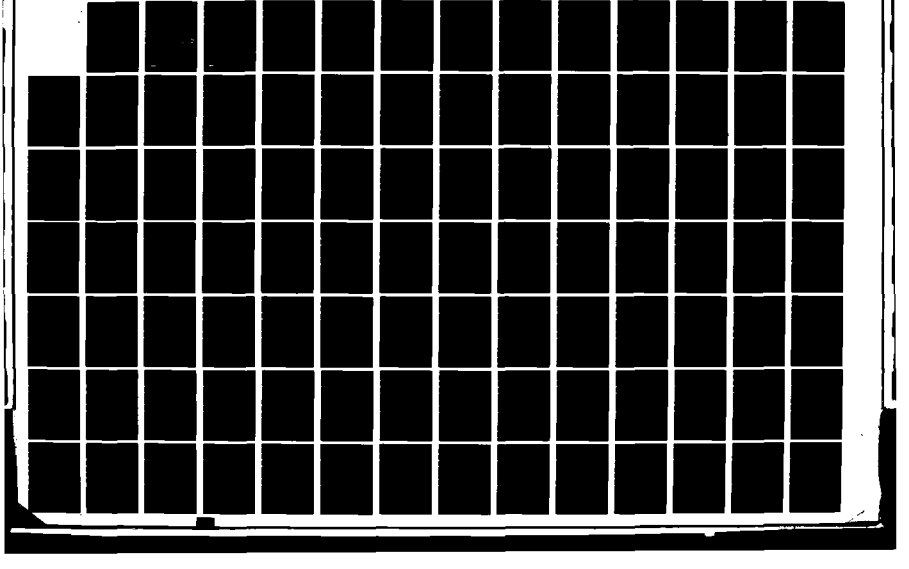
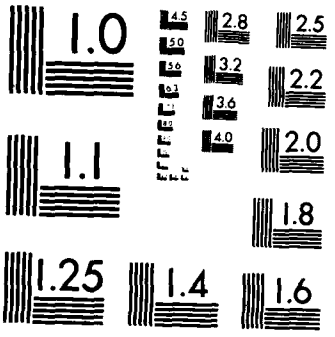
AD-A126 244 REASONING ABOUT KNOWLEDGE AND ACTION(U) SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER R C MOORE OCT 80 SRI-TN-191 1/3

UNCLASSIFIED F/G 6/4 NL





REASONING ABOUT KNOWLEDGE AND ACTION

Technical Note 191

October 1980

By: Robert C. Moore, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

This report is a slightly revised version of a thesis submitted to the Department of Electrical Engineering and Computer Science of the Massachusetts Institute of Technology on February 9, 1979, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
(415) 326-6200
Cable: SRI INTL MPK
TWX: 910-373-1246

SRI International

REASONING ABOUT KNOWLEDGE AND ACTION

Technical Note 191

October 1980

By: Robert C. Moore, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

This report is a slightly revised version of a thesis submitted to the Department of Electrical Engineering and Computer Science of the Massachusetts Institute of Technology on February 9, 1979, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Approved:

Nils J. Nilsson, Director
Artificial Intelligence Center

David H. Brandin, Executive Director
Computer Science and Technology Division

Abstract

This report deals with the problem of making a computer reason about the interactions between knowledge and action. In particular, we want to be able to reason about what knowledge a person must have in order to perform an action, and what knowledge a person may gain by performing an action. The first problem we face in achieving this goal is that the basic facts about knowledge which we need to use are most naturally expressed as a modal logic. There are, however, no known techniques for efficiently doing automatic deduction directly in modal logics. We solve this problem by taking the possible-world semantics for a modal logic of knowledge and axiomatizing it directly in first-order logic. This means that we reason not about what facts someone knows, but rather what possible worlds are compatible with what he knows. We integrate this theory with a logic of actions by identifying possible worlds with the situations before and after an action is performed. We use these notions to express what knowledge a person must have in order to perform a given action and what knowledge a person acquires by carrying out a given action. Finally, we consider some domain-specific control heuristics that are useful for doing deductions in this formalism, and we present several examples of deductions produced by applying these heuristics.

This report is a slightly revised version of a thesis submitted to the Department of Electrical Engineering and Computer Science of the Massachusetts Institute of Technology on February 9, 1979, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Contents

Abstract ii

List of Illustrations iv

1. Introduction 1

1.1 The Importance of Knowledge in Reasoning about Action 1
1.2 Overview of the Thesis 9

2. Reasoning about Knowledge 15

2.1 Formalizing Properties of Knowledge 15
2.2 Computational Problems in Reasoning About Knowledge 19
2.3 Possible-World Semantics for Knowledge 25
2.4 A Note on Belief 33
2.5 Knowledge, Equality, and Quantification 35
2.6 Other Work on Reasoning about Knowledge 47

3. An Integrated Theory of Knowledge and Action 53

3.1 Possible-World Semantics for Actions 53
3.2 The Dependence of Action on Knowledge 62
3.3 The Effects of Action on Knowledge 67

4. Formalizing the Possible-World Semantics for Knowledge 75

4.1 Object Language and Meta-Language 75
4.2 A First-Order Treatment of the Propositional Logic of Knowledge 78
4.3 Introducing Predicates, Quantifiers, and Equality 88

5. A First-Order Theory of Knowledge and Action 101

5.1 Formalizing the Possible-World Semantics for Actions 101
5.2 Formalizing the Dependence of Action on Knowledge 112
5.3 Formalizing the Effects of Action on Knowledge 118
5.4 An Example of Acquiring Knowledge Required for an Action 127

6. Automating Deductions about Knowledge 137

6.1 Procedural Deduction and First-Order Logic 137
6.2 Outline of a Procedural Deduction System 140
6.3 Procedural Interpretation of Complex Assertions 144
6.4 Inference Rules for Equality 152
6.5 Procedural Interpretation of the Axioms for Knowledge 155
6.6 Some Examples 160

7. Automating Deductions about Knowledge and Action 169

7.1 Interpreting Axioms for Knowledge and Action 169
7.2 An Example of an Action which Requires Knowledge 179
7.3 An Example of an Action which Produces Knowledge 185
7.4 An Example of Acquiring Knowledge Required for an Action 191
7.5 Remarks on the Examples 204

8. Summary and Conclusions 209

8.1 What has been Achieved? 209
8.2 Limitations and Extensions of the Current Approach 212
8.3 Conclusions 221

Bibliography 223

Appendix A: First-Order Axioms for Knowledge and Action 227

Appendix B: Procedurally Interpreted Axioms for Knowledge and Action 231

Illustrations

2.1 "John knows that P." - "John doesn't know whether Q." 28

2.2 "A knows that P." = "P is true in every world which is compatible with what A knows." 28

2.3 "If A knows that P then he knows that he knows that P." 30

2.4 "John knows that Bill knows that P." 31

3.1 $Know(A_1,Res(Do(A_2,Act),P)) =$
$\forall w_1 (K(:A_1,W_0,w_1) \supset \exists w_2 (R(:Do(:A_2,:Act),w_1,w_2) \wedge T(w_2,P)))$ 59

3.2 The effect of performing an action that is not knowledge-producing on the knowledge of the agent 70

3.3 The effect of performing a knowledge-producing action on the knowledge of the agent 71

3.4 The effect of a test on the knowledge of the agent 73

5.1 A typical blocks-world problem 108

5.2 "John can open Sf_1 by dialing $Comb(Sf_1)$." 116

5.3 "C_1 is the combination of Sf_1." 123

5.4 "C_1 is not the combination of Sf_1." 126

5.5 Acquiring knowledge required for an action 130

1. Introduction

1.1 The Importance of Knowledge in Reasoning about Action

Planning sequences of actions and reasoning about the effects of actions is one of the areas which has received the most attention from researchers in artificial intelligence (AI). Systems such as SHRDLU (Winograd, 1971), STRIPS (Fikes and Nilsson, 1971), BUILD (Fahlman, 1973), HACKER (Sussman, 1973), and NOAH (Sacerdoti, 1977) have explored issues including plan generation and debugging, representing changes to a world, skill acquisition, resolving conflicting goals, and hierarchical plan refinement. To date, however, little attention has been paid to the important role that the agent's knowledge plays in planning and acting to achieve a goal.

Almost all AI planning systems assume that they have complete knowledge of all relevant aspects of the problem domain and problem situation in which they operate. Often, any statement which cannot be inferred to be true is assumed to be false. In the real world, however, planning and acting must frequently be performed without complete knowledge of the situation. This imposes two additional burdens on an intelligent agent trying to act effectively. First, when the agent entertains a plan for achieving some goal, he must consider not only whether the physical prerequisites of the plan are satisfied, but also whether he has all the information necessary to carry out the plan. Second, he must be able to reason about what he can do to obtain the necessary information that he currently lacks.

Consider the problem of trying to open a safe. Typically, AI systems assume that if there is an action that an agent is physically able to perform, and that action results in some proposition P being true, then the agent can achieve P. In the case of opening a safe, there is certainly some action that any human agent of normal abilities is physically able to perform and that will result in the safe being open, namely, dialing the combination of the

safe. It would be highly misleading, however, to claim that an agent could open the safe simply by dialing the combination unless he *knew* what the combination of the safe was. If, on the other hand, he had a piece of paper that had the combination of the safe written on it, he could open the safe by reading what was on the piece of paper and then dialing the combination of the safe, even if he did not know the combination initially.

What we seek are techniques for creating computer systems capable of drawing conclusions such as this based on a general understanding of the relationship between knowledge and action. The question of generality is somewhat problematical, since different actions obviously have different prerequisites and results that involve knowledge. To make this issue concrete, consider trying to take knowledge into account in the STRIPS approach to the representation of actions. In this approach, knowledge about an action is represented by three lists: a list of preconditions that must be satisfied for the action to be applicable, a list of deletions which might not be true any longer after the action has been performed, and a list of additions which become true as a result of the action being performed. For instance, the action of pushing an object from one location to another is represented by the following schema (Fikes and Nilsson, 1971, p. 201):

Push(k,m,n): Robot pushes object k from place m to place n.
Preconditions: At(k,m), Atr(m)
Deletions: Atr(m), At(k,m)
Additions: Atr(n), At(k,n)

The interpretation of At(k,m) is that object k is at place m, and the interpretation of Atr(m) is that the robot is at place m. Thus, the interpretation of the entire schema is that for the robot to push an object from one place to another, the robot and the object must both be in the first place, and after the robot pushes the object, the robot and the object are no longer in the first place, but are now in the second place.

The problems that we will point out in trying to represent facts about the interaction of

knowledge and action in a STRIPS-like formalism are not unique to that system. Similar difficulties would arise in trying to extend any of the systems mentioned above to take knowledge into account. While the more recent systems use more sophisticated planning techniques than STRIPS does, their representations of the effects of actions are roughly equivalent to that used in STRIPS.

If we want to represent an action like dialing the combination of a safe, the obvious thing to do (and essentially the only thing that can be done within the STRIPS approach) would be to have one of the preconditions be that the agent knows the combination of the safe. This is much more specific than it needs to be, however. Doing things this way fails to suggest any connection between the fact that dialing the combination of the safe requires knowing the combination and the fact that calling someone on the telephone requires knowing his phone number or the fact that pushing a block to a certain location requires knowing where that location is.

What all these examples have in common is that being able to use any action to achieve a goal requires knowing what action to take. From this point of view, knowing the combination of a safe is not really a precondition for dialing the combination of the safe; rather, it is required for knowing what action dialing the combination of the safe *is*. Similarly, knowing what action constitutes calling someone on the telephone normally requires knowing his phone number, and knowing what action constitutes pushing something to a certain location requires knowing where that location is.

Now, we propose that for a general action like dialing combinations of safes, if an agent knows what the action is (i.e., he knows how to dial combinations of safes "in general"), then he knows what some specific instance of the action is (e.g. dialing combination C_1 on safe Sf_1) if he knows what objects the action is being applied to in that instance. So knowing what combination C_1 is and knowing what safe Sf_1 is would be sufficient for knowing what

action dialing C_1 on Sf_1 is. On the other hand, an agent could know that dialing the combination of Sf_1 on Sf_1 will result in Sf_1 being open, but not be able to open Sf_1 because he doesn't know what combination the description "combination of Sf_1" refers to, and therefore doesn't know what action constitutes dialing the combination of Sf_1. A similar analysis applies to the examples of calling someone on the telephone or pushing a block to a certain location, so we have one general principle that covers all the examples, rather than different knowledge preconditions for each case.

Adequately representing the effects of actions on knowledge also goes beyond what can easily be represented using the STRIPS approach. This might seem to be rather straightforward. If we have an information gathering operation, like looking into a box, we could simply put on the list of additions for the action that the agent knows what is in the box and put on the list of deletions that he does not know what is in the box. This might be all right for gains in knowledge by direct observation, but there are more subtle problems that it overlooks.

Consider the notion of a test. The essence of a test is that it is an action that has a directly observable result that depends conditionally on an unobservable precondition. In the use of litmus paper to test the pH of a solution, the observable result is whether the paper is red or blue, and the unobservable precondition is whether the solution is acid or alkaline. What makes such a test useful for acquiring knowledge is that the agent can infer from his knowledge of the behavior of litmus paper and the observed color of the paper whether the solution is acid or alkaline. In a test it is usually this inferred knowledge, rather than what is directly observed, that is important. After all, the color of a piece of litmus paper is seldom an intrinsically interesting piece of information.

If we follow the previous suggestion in trying to formulate a STRIPS operator for using litmus paper, we will have to include the result that the agent knows whether the solution is

acid or alkaline as a separate fact from the result that he knows the color of the paper. If we do this, however, we completely miss the point that the knowledge of the pH of the solution is inferred from other knowledge, rather than being a direct observation. Moreover, we are in effect *specifying* what actions constitute possible tests, rather than creating a system that is able to *infer* what actions are possible tests.

If we want to capture the inference that the agent must make to use a test, we have to represent several independent pieces of knowledge that the agent must have. Obviously, we have to represent that after the test is performed, the agent knows the observable result. This much is handled by the STRIPS approach. Furthermore, we have to represent the fact that he knows that the test *has been performed.* If he just walks into the room and sees the litmus paper on the table, he will know what color it is, but unless he knows its recent history, he won't have gained any knowledge about the acidity of the solution. Representing this knowledge in a principled way is a problem for the STRIPS approach. The formulas on the lists of additions and deletions are taken to be true or false at a particular time, without reference to other times. We could introduce an ad hoc predicate on actions, Has-Just-Occurred(x), but there is no way to relate this predicate to the notion of time implicit in the distinction between preconditions and postconditions (additions and deletions) in the descriptions of actions.

We also need to represent the fact that the agent understands how the test works; that is, he knows how the observable result of the action depends on the unobservable precondition. Even if he sees the litmus paper put into the solution, and he sees the paper change color, he still won't know whether the solution is acid or alkaline, unless he knows how the color of the paper is related to the acidity of the solution. This knowledge is, in fact, just what would be expressed by the STRIPS description of the action. This creates another problem for the STRIPS approach, since descriptions of actions are not part of the

language that preconditions and postconditions are written in. If knowing how the physical preconditions of an action affect the physical results of an action is a precondition to using the action as a test, then the language in which preconditions are written must be able to describe at least the physical effects of the action.

Finally, the system must be able to reason that if the agent knows (i) that the test took place, (ii) the observable result of the test, and (iii) how the observable result depends on the unobservable precondition, then he knows the unobservable precondition. Thus the system must incorporate a logic of knowledge to tell it when someone's knowing a certain collection of facts implies that he knows other facts.

From the preceding discussion, we can conclude that any system that is capable of reasoning about tests at this level of detail must be able to explicitly represent facts of the following types:

(1) A knows that Q will be true after he performs Act just in case P is true now.

(2) After A performs Act, he knows that he has just performed Act.

(3) After A performs Act, he knows whether Q is true.

In order to reason that an agent can use a certain test to find out a piece of information the system must also embody or be able to represent general principles sufficient to conclude:

(4) If (1), (2), and (3) are true, then after performing Act, A will know whether P was true before the action was performed.

(5) If A knows that it is possible for him to achieve P by performing Act and he knows what action Act is, then he can achieve P by performing Act.

(6) If Act is a specific instance of a class of actions that A knows how to perform in general, then he knows what action Act is just in case he knows what the arguments of Act refer to.

It is important to emphasize that for any work on these problems to be of real value it

must seek general principles. For instance, it would be possible to represent (1), (2), and (3) in an arbitrary ad hoc way and add an axiom which explicitly states (4), thereby "capturing" the notion of a test. Such an approach, however, would simply restate the observations that we have made in this section. Our goal in this thesis will be to create a system in which specific facts like (4) follow from the most basic principles of reasoning about knowledge and action.

There has been little previous work in AI on these problems. McCarthy and Hayes (1969) were the first AI workers to take note of the problem of actions with knowledge preconditions. Their proposed solution is somewhat sketchy, and it seems to have some problems. They first present a set of axioms expressed in the situation calculus that can be used to deduce that dialing the combination of a safe will result in the safe being open. They then point out that this procedure may be infeasible for an agent, because he may not know the combination of the safe. Next they introduce the expression idea-of-combination(p,sf,s) to mean person p's idea of the combination of the safe sf in situation s, and suggest, but do not formalize, that it would be feasible for anyone to dial his idea of the combination of the safe, since he presumably does know that. Given this, if it can be shown that p's idea of the combination of the safe is, in fact, the combination of the safe, then dialing the combination of the safe is both feasible for p and effective in opening the safe, so it is possible for p to open the safe.

The requirement that the action be feasible for the agent seems too weak, however. Suppose that 11L-22R-33L is the combination of the safe. If this is true, then dialing 11L-22R-33L will result in the safe being open. Furthermore, dialing 11L-22R-33L will be feasible for anyone who understands in general how to dial combinations of safes. But McCarthy and Hayes's argument would lead us to infer that any such person could open the safe, whether or not he knew the combination. The central role of knowing the combination has been missed.

Another problem with McCarthy and Hayes's approach is the ad hoc character of the idea-of-combination function. The problem is that the logic does not make any special connection between idea-of-combination and combination, the function which picks out the actual combination of a safe. Therefore, for each term in the language that we wanted to talk about someone's knowledge of, we would have to introduce a separate idea-of... function. McCarthy and Hayes acknowledged the clumsiness of this approach, but saw no other way of preserving the property of referential transparency, the ability to substitute equals for equals. They would have preferred to use a general idea-of function, such that idea-of(p,combination-of(sf),s) would refer to p's idea in s of the combination of sf. The trouble is that if equals can be substituted for equals, and the combination of sf_1 is the same as the combination of sf_2, then this would imply that p's idea of the combination of sf_1 would have to be the same as p's idea of the combination of sf_2, which is not necessarily the case. We will present a much more elegant solution to this problem in section 2.5. More recently, McCarthy (1979) has also taken a different approach to this problem which we will discuss in section 2.6.

The only other work in AI that deals explicitly with the interaction of knowledge and action seems to be that of Cohen (1978). Cohen's formalism is a straightforward encoding of the STRIPS approach into semantic network notation, with all the limitations that we have pointed out. Cohen never faces any of the issues we have raised, because he does not really deal with the problems of *reasoning* about knowledge. His system generates plans that have effects on what people know and that require knowledge to execute, but all statements about knowledge must be explicitly asserted; the system has no ability to infer them.

1.2 Overview of the Thesis

This thesis attacks the problems of representing the kinds of facts and making the kinds of inferences described in the previous section. First we will discuss the representation problems. Then we will describe a formalism that captures the distinctions we need to make and permits reasonably efficient automatic inferencing. Then we will outline a system (as yet unimplimented) for automatically carrying out deductions in this formalism and illustrate its operation with several hand-simulated examples. We will organize the presentation around a set of examples having to do with dialing combinations and opening safes. Starting from a set of premises that respect the generalizations we have discussed, we will show how to automatically deduce that:

(1) If John is at the same place as the safe Sf_1, and he knows the combination of the safe, he can open the safe by dialing the combination.

(2) If C_1 is the combination of Sf_1, and if John tries to open Sf_1 by dialing C_1, he will then know that C_1 is the combination of Sf_1.

(3) If John is at the same place as the Sf_1 and the piece of paper Ppr_1, and he knows that the combination of Sf_1 is the only thing written on Ppr_1, he can open Sf_1 by reading the piece of paper and dialing the combination.

The first of these examples involves understanding what knowledge is sufficient for being able to achieve a goal by performing a certain action. The second example shows how knowledge can be acquired by using an action as a test. The third example involves carrying out a sequence of actions, first performing one action to obtain some knowledge, and then using that knowledge to perform another action that achieves a goal.

It should be emphasized that we are not attacking the problem of automatically generating plans which take into account the acquisition and use of knowledge. Rather, as the examples suggest, we are limiting our efforts to reasoning about how knowledge

interacts with a given action or sequence of actions. Although we would claim that this is a prerequisite to solving the planning problem, it is certainly not sufficient by itself.

Besides wanting to extend the capabilities of AI systems in reasoning about actions, there are more general reasons for undertaking this study. First of all, there is a need for AI to break out of what might be termed "the blocks-world syndrome". Most of the work in AI on common-sense reasoning and common-sense problem solving has dealt only with discrete physical objects and physical relations, sometimes operated on by simple sequences of actions.

This leaves a multitude of representational problems untouched. Some of these problems include modalities such as possibility ("It might be the case that..."), necessity ("It must be the case that..."), ability ("John can do..."), permissibility ("John may do..."), and obligation ("John should do..."). Other problems include counterfactual conditionals ("If I had struck the match, it would be burning now."), action modifers ("almost", "quickly", "carefully"), and propositional attitudes ("wants", "fears", "believes", "knows"). Most AI systems treat time as a sequence of discrete states, rather than as a continuum, and little work has been done on reasoning about continuous substances such as water and air. (This last observation is due to Hayes (1974).)

There is a large literature in modern philosophical logic in many of these areas which seems directly applicable to AI problems, but very little of it has been explored by the AI community. One of the goals of this thesis is to take the ideas of philosophical logicians in one particular area, in this case the logic of knowledge, and see to what extent they can be applied to AI problems.

Another goal of the thesis is to provide a testbed for exploring ideas about automatic deduction. Most work on automatic deduction has been done in the area of mathematical theorem proving, with what is generally perceived to be mixed results. Despite years of

effort, no theorem proving program has ever proved a significant new result in mathematics. The difficulty in evaluating this work is that the problems are so hard that it is not clear what should count as success. After all, the number of people who have proved interesting new results in mathematics is miniscule compared to the number who can solve the block stacking problems that have been studied in AI.

The example problems which we will look at do not share this uncertainty. They are clearly solvable by anyone of normal intelligence; yet they will turn out to be a non-trivial test for our deductive system. So this domain gives us problems that are rich enough to be challenging, but easy enough that we are sure we should be able to solve them.

The first problem we will face in carrying out this project is that the basic facts about knowledge that we need to use are most naturally expressed as a modal logic (Hughes and Cresswell, 1968). So far, no satisfactory way of applying automatic deduction techniques directly to modal logics has been developed. In chapter 2, we first discuss what properties of knowledge we need to formalize, and explain the computational problems created by some simple approaches. Fortunately, we can get around these problems by making use of the possible-world semantics for the logic of knowledge developed by Hintikka (1962, 1969), based on the possible-world semantics for necessity of Kripke (1963a, 1963b). The approach which we will pursue is to axiomatize this model theory for the logic of knowledge directly in first-order logic. We will have axioms which say such things as that A knows that P if and only if P is true in every possible world which is compatible with what A knows. In this way, we can reason about simple relations among possible worlds rather than troublesome modal operators like Know. In the rest of chapter 2, we briefly discuss applying the same approach to reasoning about belief, consider in detail the issues raised by the introduction of quantifiers and equality into knowledge contexts, and review some alternative approaches that have been proposed.

In chapter 3 we show how to extend this approach to reasoning about the interaction of knowledge and action. The key ideas are (1) to recast McCarthy's situation calculus (McCarthy, 1963), (McCarthy and Hayes, 1969), as a modal logic with a corresponding possible-world semantics, and (2) to unify this formalism with the one for knowledge by identifying possible worlds in the formalism for knowledge with situations in the formalism for actions. We show how to describe both the dependence of action on knowledge in terms of knowing precisely what action to perform, and how to describe the effects of action on knowledge in terms of relations between possible worlds.

In chapter 4, we present the details of a first-order axiomatization of the possible-world semantics for knowledge, and illustrate its use with a number of examples of formal deductions. In chapter 5 we extend the formalism to handle our integrated theory of knowledge and action, giving more examples.

In chapter 6, we turn to problems of automatically generating deductions involving statements about knowledge. We first present an outline of an automatic deduction system in which certain formulas are given procedural interpretations. This is in the tradition of PLANNER (Hewitt, 1972) and related formalisms, and was studied in detail by Moore (1975). We then discuss the appropriate procedural interpretations for facts about knowledge, and show how to use these interpretations to generate some simple deductions.

In chapter 7 we consider automatic deductions involving both knowledge and action, and discuss in detail the choice of procedural interpretations for the axioms describing dialing the combination of a safe and reading a piece of paper. This chapter concludes with detailed explanations of automatically generated deductions of the three sample problems given in this section. Finally, chapter 8 summarizes and evaluates our results, and suggests possible extensions.

This thesis makes a number of substantial original contributions. One of these pointing

out the efficiency advantages of using a first-order formalization of the possible-world semantics of a modal logic of knowledge and action over some of the more obvious approaches to reasoning directly in the modal logic itself. The idea of doing deductions in modal logics indirectly using first-order formalizations of their semantics has been suggested a few times in the AI literature (McCarthy and Hayes, 1969), (Morgan, 1976), but has not been extensively pursued. Our point is that the possible-world approach is not merely one of several possible ways of reducing a modal logic to an ordinary first-order logic, but that it has important properties that enable standard deduction techniques to be used with reasonable efficiency. The reasons for this are explained in section 2.3, with analysis of the shortcomings of alternative approaches being given in sections 2.2 and 2.6.

Also, our formalism seems to be the first using this approach to give a fully adequate treatment of quantification and equality for the logic of knowledge. The only previous application (of which I am aware) of a first-order formalization of the possible-world semantics for modal logics to reasoning about knowledge is in some unpublished notes by McCarthy and one of his students (McCarthy, 1975), (Goad, 1976). Their work considers only a propositional form of the logic of knowledge, however. Our formalism, on the other hand, deals with a full quantified logic of knowledge with equality, which is essential for carrying out the inferences about knowledge and action that we want our system to handle.

The ideas on integrating reasoning about knowledge and action seem to be entirely new. The main contributions here are the idea of describing the effects of actions in terms of a modal logic parallel to the modal logic for knowledge, unifying the two logics by identifying the situations in the semantics of the logic of actions with possible worlds in the semantics of the logic of knowledge, analyzing the knowledge preconditions for actions in terms of knowing what action to perform, and describing the effects of actions on knowledge in terms of relations between possible worlds. These ideas are the major

theoretical contribution of this thesis, and they make it possible to do reasoning about knowledge and action with the kind of generality that we are seeking. For instance, they make it possible to derive the properties of tests which we discussed in the previous section from our general theory of knowledge and action. The possible-world semantics for action also provides a very attractive picture of the relation between procedures and processes. It falls out naturally from this semantics that a procedure is a description of a process. Technically, the denotation of a certain procedure in a certain environment is the process which results from executing the procedure in the environment.

Finally, no other work has seriously investigated the problems of doing automatic deductions in this domain. Most of the techniques we present are not new (although many of them are due to this author (Moore, 1975)), but applying them to our formalism requires extensive and subtle analysis. In fact, it is probably fair to say that this is the most complex formalization of a common-sense domain to which these sorts of techniques have been applied and represents their most severe test to date.

2. Reasoning about Knowledge

2.1 Formalizing Properties of Knowledge

Since techniques for reasoning about action have been extensively studied in AI, while techniques for reasoning about knowledge have not, we will attack the problems of reasoning about knowledge first. In chapter 3 we will see that the formalism that we are led to as a solution to these problems turns out to be well suited for an integrated system for reasoning about both knowledge and action.

The first step in devising a formalism for reasoning about knowledge is to decide what general properties of knowledge we want that formalism to capture. It should be emphasized, however, that we are not going to attempt to define what knowledge is. That enterprise, which belongs to the branch of philosophy called epistemology, has been going on for several thousand years without reaching a consensus, and we cannot hope to solve the problem in this thesis. More importantly, it is not necessary to solve that problem for our purposes. The goal of epistemology is to have an *explanatory* theory of knowledge, whereas all we need is a *descriptive* theory. We can perfectly well do common-sense reasoning about knowledge without having a theory of epistemology, just as we can do common-sense reasoning about physical objects without having a theory of physics. What we will need to do is to specify some of the basic properties of the common-sense notion of knowledge, or more precisely, *a* common-sense notion of knowledge that is useful for reasoning about planning and acting. *Any* philosphical theory of knowledge that explains these properties would be acceptable from this point of view, but it is not necessary for us to have such a theory to achieve our goals.

This being said, the properties of knowledge that we will be most interested in formalizing are the ones that are relevant to planning and acting. One such property is

that anything that someone knows is true; it is impossible to have false knowledge. If P is false, we would not want to say that John knows P. We might say that John believes P or that John believes he knows P, but if P is false, then it simply could not be the case that John knows P.

This is, of course, a major difference between knowledge and belief. If we say that John believes P, we are not committed to saying that P is either true or false, but if we say that John knows P, we are committed to the truth of P. The reason that this distinction is important for planning and acting is simply that for an agent to achieve his goals, the beliefs that he bases his actions on must generally be true. After all, merely believing that performing a certain action will bring about a desired goal is not sufficient for being able to achieve the goal; the action must actually have the intended effect.

Another fact that turns out to be important for planning is that if someone knows something, he knows that he knows it. This principle is often required for reasoning about plans consisting of several steps. Suppose an agent plans to use Act_1 to achieve his goal, but in order to perform Act_1 he needs to know whether P is true and whether Q is true. Suppose further that he already knows that P is true, and can find out whether Q is true by performing Act_2. The agent needs to be able to reason that after performing Act_1 he will know whether P is true and whether Q is true. We will be willing to assume that he knows that he will know whether Q is true if he understands the effects of Act_2, but how does he know that he will know whether P is true? Presumably it works something like this: He knows that P is true, so he knows that he knows that P is true, and he knows how Act_2 affects P, so he knows that he will know whether P is true after he performs Act_2. The key step in this argument is an instance of the principle that if someone knows something, he knows that he knows it.

It might seem that we would also want to have the principle that if someone doesn't

know something he knows that he doesn't know it, but this turns out to be false. Suppose that John believes that P, but in fact, P is not true. Since P is false, John certainly doesn't know that P, but it is highly unlikely that he knows that he doesn't know, since he thinks that P is true.

Probably the most important fact about knowledge that we will want to capture is that people can reason based on their knowledge. All of the examples we have given depend on the assumption that if an agent trying to solve a problem has all the relevant information, he will apply his knowledge to get a solution. This presents a difficulty for us, however, since people don't, in fact, know all the logical consequences of their knowledge. The trouble is that we never can be sure which of the inferences that a person *could* make, he *will* make. I believe that the best solution is to adopt the principle that if P is implied by what someone knows, then he also knows P, but to treat it as a "plausible implication".

By "plausible implication", we will mean an implication schema that we will accept in any particular case unless we have other information to the contrary. A plausible implication, then, would behave just like an ordinary implication so long is nothing is inferred which contradicts a previous conclusion. If only one plausible inference is made in a chain of reasoning that leads to a contradiction, then that inference is almost certainly the one which should be withdrawn. If more than one plausible inference is involved, then we get into the complicated problem of choosing between alternative plausible views of the world (McDermott, 1974), (Doyle, 1978). This is a very general problem of which the effects of adopting the proposed principle are only one example. However, since our examples involve such mundane bits of reasoning that it is extemely unlikely that any intelligent agent would fail to make them, we will treat the principle as if it were an ordinary implication, and not consider the problem any further.

Finally, we will need to include the fact that these basic properties of knowledge are

themselves *common knowledge*. By this we mean that everyone knows them, and everyone knows that everyone knows, and everyone knows that everyone knows that everyone knows, etc. This type of principle is obviously needed when reasoning about what someone knows about what someone else knows, but it is also important in planning, because an agent must be able to reason about what he will know at various times in the future. In such a case, his "future self" is analogous to another person.

In his pioneering work on the logic of knowledge and belief, Hintikka (1962) presents a formalism that captures all these properties. We will define a formal logic based on Hintikka's ideas, but modified somewhat to be more compatible with the additional developments in this thesis. So, what follows is similar to the system developed by Hintikka in spirit, but not in detail.

The language of this system is the language of propositional logic augmented with the operator Know. The formula Know(A,P) is interpreted to mean that the person denoted by the term A knows the proposition corresponding to the formula P. So if John refers to John and Likes(Bill,Mary) means that Bill likes Mary, Know(John,Likes(Bill,Mary)) means that John knows that Bill likes Mary. The closure of the following axiom schemata with respect to the inference rule modus ponens (from (P ⊃ Q) and P, infer Q) defines the theorems of the system:

M1. Axioms of ordinary propositional logic (e.g. as in Rogers (1971))
M2. Know(A,P) ⊃ P
M3. Know(A,P) ⊃ Know(A,Know(A,P))
M4. Know(A,(P ⊃ Q)) ⊃ (Know(A,P) ⊃ Know(A,Q))
M5. If P is an axiom, then Know(A,P) is an axiom.

This system is very similar to the systems studied in modal logic. In fact, if A is held fixed, the resulting system is isomorphic to the modal logic S4 (Hughes and Cresswell, 1968). We will refer to this system as the modal logic of knowledge.

These axioms formalize in a straightforward way the principles for reasoning about

knowledge which we discussed. M2 says that anything that is known is true. M3 says that if someone knows something, he knows that he knows it. M4 says that if someone knows a formula P and a formula of the form $(P \supset Q)$, then he knows the corresponding formula Q. That is, everyone can (and does) apply modus ponens. M5 is a recursive schema which tells us that all the axioms are common knowledge. It first applies to M1 - M4, which says that everyone knows the basic facts about knowledge, but it also applies to its own output, so we get axioms that say that everyone knows that everyone knows, etc. Since M5 applies to the axioms of propositional logic (M1), we can infer that everyone knows the facts they represent. Furthermore, since modus ponens is the only inference rule needed in propositional logic, the presence of M4 will enable us to infer that someone knows any propositional consequence of his knowledge.

2.2 Computational Problems in Reasoning about Knowledge

The modal logic of knowledge that we have just presented is an elegant and concise formalization of the properties of knowledge that we want to capture, but it has one major drawback as a basis for AI systems for reasoning about knowledge; so far, there have not been devised any satisfactory ways of applying automatic deduction techniques directly to systems of this type. The reason that the standard techniques cannot be applied directly is that Know is an *intensional* rather than an *extensional* operator. Classical logics are extensional because the truth value of a complex formula depends only on the extensions, or denotations, of its subexpressions. (The denotation of a term is the individual it refers to, the denotation of a predicate symbol is the set of individuals that satisfy it, and the denotation of a formula is its truth value.) For instance, the truth of $(P \vee Q)$ depends only on the truth of P and the truth of Q; no other properties of P and Q matter. In particular, the intensions, or meanings, of P and Q are irrelevant, except in so far as meaning

determines truth value. This is true of both first-order and higher-order classical logics. This restriction was recognized by the founders of modern logic (Whitehead and Russell, 1910), and it is one of their great triumphs that they succeeded in formalizing essentially all of mathematics within a purely extensional framework.

Knowing, on the other hand, is an intensional notion because the truth of "A knows that P," depends generally on the meaning of P, rather than just its truth value. The truth value of P is clearly important, since it is impossible to know a false proposition, but it is not the whole story, since it is possible to know some true propositions and not know other true propositions. The standard techniques for automatic deduction have been worked out only for extensional formalisms, so we either have to extend the known techniques or find a way to convert the modal logic of knowledge into an extensional formalism.

To see what the difficulties really are we will examine some simple approaches and point out where they fail. Suppose that we have a system for doing deductions in propositional logic and we simply add formulas of the form Know(A,P) to the data base. It is easy to imagine that such a system could match formulas like this, even though they are not propositional in the strictest sense, and be able to infer things like Know(A,Q) from Know(A,P) and Know(A,P) ⊃ Know(A,Q). The system would not, however, be able to do any inferences that depend on the occurrence of logical operators inside of a Know operator. The rules of propositional logic alone would not be sufficient to infer Know(A,Q) from Know(A,P) and Know(A,(P ⊃ Q)).

The obvious next step would be to add the axioms of the modal logic of knowledge to the data base. This might present something of a problem, since they are schemata rather than simple axioms (M5 would seem to be particularly troublesome), but we will let this pass, since there are more severe difficulties to follow. This would produce a system capable of doing all the deductions permitted by our logic of knowledge, but would be horrendously

inefficient if done in the obvious way. Basically, this move reduces the role of the underlying deductive system to that of a simple interpreter, with the real control structure of the deductive process being encoded in the axioms. Using a set of axioms to specify a procedure is not necessarily inefficient. Indeed, recent work in "logic programming" (e.g. Kowalski (1974)) is based on this very notion. Sussman and his colleagues (de Kleer, et al., 1977) also use axioms to specify control information, but in a quite different way. The point is that such axiom sets must be carefully designed to produce reasonable procedures when interpreted. The modal logic of knowledge which we have been considering was obviously not designed for that purpose. Consider deductions involving the axiom schema M4:

Know(A,(P ⊃ Q)) ⊃ (Know(A,P) ⊃ Know(A,Q)).

How should this schema be used? If we use it to add new facts to the data base, it will match any formula of the assertion of the form Know(A,(P ⊃ Q)), producing a new assertion of the form (Know(A,P) ⊃ Know(A,Q)). Used this way, however, M4 will interact with M5 to add infinitely many new assertions to the data base. The result of applying M5 once to M4 could be written as:

Know(B,Know(A,(P ⊃ Q)) ⊃ (Know(A,P) ⊃ Know(A,Q))),

but M4 would apply to this in turn to produce:

Know(B,Know(A,(P ⊃ Q))) ⊃ Know(B,(Know(A,P) ⊃ Know(A,Q))).

This would apply to the schema that results from applying M5 twice to M4, eventually producing a formula that would apply to the axioms that result from applying M5 three times to M4, etc., ultimately producing analogues of M4 for all depths of nesting of Know.

So, to avoid generating infinitely may new assertions, we would have to use M4 as a

subgoal generator, if it is to be used at all. Used in this way, it would match any goal of the form Know(A,Q), generating the conjunctive subgoals Know(A,(P ⊃ Q)) and Know(A,P). Whichever of these goals is attempted first, M4 applies again, producing still more complicated goals. It is possible to continue in this way to an arbitrary depth without ever considering any substantive facts relevant to deducing the original goal.

The fundamental problem with this approach is that no matter how smart the basic deductive system is, the axioms for knowledge seize control of the deductive process. These axioms were originally selected for their elegance and brevity, not for efficent generation of proofs. What seems to be needed, then, is a way of running the basic deduction system "inside" the operator Know.

One way of producing such a system would be to avoid having axioms for knowledge altogether by finding a computational analogue of knowing, some computational structure which can serve as a direct representation of someone knowing something. There is an idea along these lines that initially seems very appealing. Using the multiple data-base capabilities of advanced AI languages, we could set up a separate data base for each person whose knowledge we have some information about. We then can record what we know about his knowledge in that data base, and simulate his reasoning by running our standard inference routines in that data base. This would allow us to eliminate any explicit reference to knowing individual facts, and so avoids dealing with the modal operator Know. This idea seems to have wide currency in AI circles, and I advocated it myself in an earlier paper (Moore, 1973).

This idea handles simple statements about knowledge quite well. Suppose that DB_{rw} contains what we believe to be true about the real world. If we want to assert that John knows that P we would create a new data base, $DB_{rw.john}$, assert P in this new data base, and set a pointer in the old data base to indicate where we can find information about

John's knowledge. Furthermore, to assert that John knows that Bill knows that P, all we have to do is iterate this process. We create a third data base, $DB_{rw.john.bill}$, assert P in this data base, and set a pointer to it in $DB_{rw.john}$ labeled "Bill's knowledge". Since this is already in a data base which is restricted to John's knowledge, it would automatically be interpreted as what John knows about what Bill knows.

If we want to make assertions that are logically more complex, however, we run into trouble. Consider the problem of representing "John knows that P or John knows that Q." We can't represent this by simply adding (P ∨ Q) to $DB_{rw.john}$, because this would mean "John knows that P or Q," - something quite different. We could set up two data bases, $DB_{rw.john1}$ and $DB_{rw.john2}$, add P to one and Q to the other, and then assert in DB_{rw} "$DB_{rw.john1}$ represents John's knowledge, or $DB_{rw.john2}$ represents John's knowledge." However, if we also wanted to assert "John knows that R, or John knows that S, or John knows that T," we would need *six* data bases to represent all the possibilites for John's knowledge - one for each of the combinations P and R, Q and R, P and S, etc. As we add more disjunctive assertions, we get a combinatorial explosion in the number of data bases. A more sophisticated approach might retain the modal operator Know for the basic representation and convert to the data base representation only after the specific facts relevant to the problem at hand have been identified, thus limiting the combinatorics. Stallman and Sussman (1976) and Doyle (1978) have worked out advanced techniques for handling multiple data bases that might be useful in this approach, but the details remain to be worked out.

A more serious problem is representing what someone *doesn't* know. Suppose we want to represent "John doesn't know that P." We can't add ¬P to $DB_{rw.john}$, because this would be asserting "John knows that ¬P," and simply omitting P from $DB_{rw.john}$ means that we don't know whether John knows that P.

I have heard two suggestions as to how this problem might be overcome. One suggestion is that we change conventions so that omitting something from $DB_{rw.john}$ means that John doesn't know it. This would, however, require explicitly representing all aspects of John's knowledge of which we are ignorant, a prospect which seems far more troublesome than the original problem.

The other suggestion is to let the logic used in the data bases be three-valued - true, false, and undefined. If P is marked as true in $DB_{rw.john}$, then John knows that P; if P is marked as false, then John knows ¬P; if P is marked as undefined, then John doesn't know one way or the other. This doesn't work for multiple embeddings of Know, however. Representing "John knows that Bill doesn't know whether P," is no problem. We simply mark P as undefined in $DB_{rw.john.bill}$. But how can we represent "John doesn't know whether Bill knows that P."? There is no assertion in $DB_{rw.john}$ to mark as undefined, because "Bill knows that P," is represented implicitly by asserting P in $DB_{rw.john.bill}$.

It appears, then, that what John doesn't know has to be kept separate from what he does know. But there are inferences that require looking at both. For example, if we have "John doesn't know that P," and "John knows that Q implies P," we might want to conclude that "John doesn't know that Q," is probably true.

This is representative of a class of inferences that the data base approach doesn't capture. There seems to be a fundamental problem in saying things about a person's knowledge that go beyond simply enumerating what he knows. There may be ways of getting around these difficulties, but it is clear that any adequate solution is going to be much more complex than "just using data bases".

2.3 Possible-World Semantics for Knowledge

So far, all of the proposals that we have seen for reasoning directly about formulas of the form Know(A,P) have led to problems. There may well be solutions to these problems, but it turns out that they can be circumvented entirely by changing the language we use to describe what people know. Rather than talk about the individual statements that someone knows we will talk instead about what states of affairs are compatible with what he knows. In philosophy, these states of affairs are usually called "possible worlds", so we will adopt that term as well.

This move to describing knowledge in terms of possible worlds is based on a rich and elegant formal semantics for systems like our modal logic of knowledge that was developed by Hintikka (1962, 1969) in his work on knowledge and belief. The advantage of this approach is that it can be formalized within ordinary first-order classical logic in a way that permits the use of standard automatic deduction techniques in a reasonably efficient manner.

Possible-world semantics was first developed for the logic of necessity and possibility. It is a very old idea in philosophy (usually attributed to Leibniz) that a proposition is necessarily true if and only if it is true in all possible worlds. Conversely, a proposition is possibly true if and only if there is some possible world where it is true. Intuitively, a possible world may be thought of as a set of circumstances that might have been true in the actual world. This informal analysis leaves many questions unanswered, however. We have said that a necessary truth must be true in all possible worlds, but must it be necessarily true in all of them? Or, are possibly true propositions necessarily possible? Axiomatizations of modal logics have proliferated as philosophers have argued various sides of questions such as these.

In the early 1960's, the development of formal possible-world semantics provided a

unifying framework for viewing these various axiom systems. The key new idea was to regard different worlds as being possible, not absolutely, but only relative to other worlds. That is, the world W_1 might be a possible alternative to W_2, but not to W_3. The structure of which worlds are possible alternatives to which other worlds is said to define an *accessibility relation*. The high point in the development of this theory came when Kripke (1963a) proved that the differences among some of the most important proposed axiom systems for modal logic corresponded exactly to certain restrictions on the accessibility relation of the possible-world models of those systems. These results are reviewed in Kripke (1963b).

Concurrent with these developments, Hintikka (1962) published the first of his work on the logic of knowledge and belief, which included a model theory that was much like Kripke's possible-world semantics. Hintikka's original semantics was done in terms of sets of sentences, which he called *model sets*, rather than possible worlds. Later (Hintikka, 1969), however, he recast his semantics into Kripke's terms, and it is that formulation which we will use here.

Kripke's semantics for necessity and possibility can be converted into Hintikka's semantics for knowledge by changing the interpretation of the accessibility relation. In order to analyze statements of the form Know(A,P), we will introduce a relation K, such that $K(A,W_1,W_2)$ means that the possible world W_2 is compatible or consistent with what A knows in the possible world W_1. In other words, for all that A knows in W_1, he might just as well be in W_2. It is the set of worlds $\{w_2 \mid K(A,W_1,w_2)\}$ that we will use to characterize what A knows in W_1. We will discuss A's knowledge in W_1 in terms of this set, the set of states of affairs that are consistent with his knowledge in W_1, rather than in terms of the set of propositions that he knows. For the present we will assume that the first argument position of K admits the same set of terms as the first argument position of Know. When we consider

quantifiers and equality in section 2.5, we will have to modify this assumption, but it will do for now.

Introducing K is the key move in our analysis of statements about knowedge, so understanding what K means is particularly important. To illustrate, suppose that in the actual world - call it W_0 - John knows that P, but doesn't know whether Q. If W_1 is a world where P is false, then W_1 is not compatible with what John knows in W_0, so we would have $\neg K(John,W_0,W_1)$. Suppose that W_2 and W_3 are compatible with everything John knows, but Q is true in W_2 and false in W_3. Since John doesn't know whether Q is true, for all he knows, he might be in either W_2 or W_3 instead of W_0. Hence, we would have both $K(John,W_0,W_2)$ and $K(John,W_0,W_3)$. This is depicted graphically in figure 2.1.

Some of the properties of knowledge can be captured by putting constraints on the accessibility relation K. For instance, requiring that the actual world W_0 be compatible with what each knower knows in W_0, i.e., $\forall a_1(K(a_1,W_0,W_0))$, is equivalent to saying that anything that is known is true. That is, if the actual world is compatible with what everyone (actually) knows, then no one has any false knowledge. This corresponds to the modal axiom M2.

The definition of K implies that if A knows that P in W_0, then P must be true in every world W_1 such that $K(A,W_0,W_1)$. To capture the fact that people can reason with their knowledge, we will assume the converse is also true. That is, we assume that if P is true in every world W_1 such that $K(A,W_0,W_1)$, then A knows that P in W_0. (See figure 2.2.) This principle is the model-theoretic analogue of axiom M4 in the modal logic of knowledge. To see that this is so, suppose that A knows that P and that $(P \supset Q)$. Therefore, P and $(P \supset Q)$ are both true in every world that is compatible with what A knows. If this is the case, though, then Q must be true in every world that is compatible with what A knows. By our assumption, then, we conclude that A knows that Q.

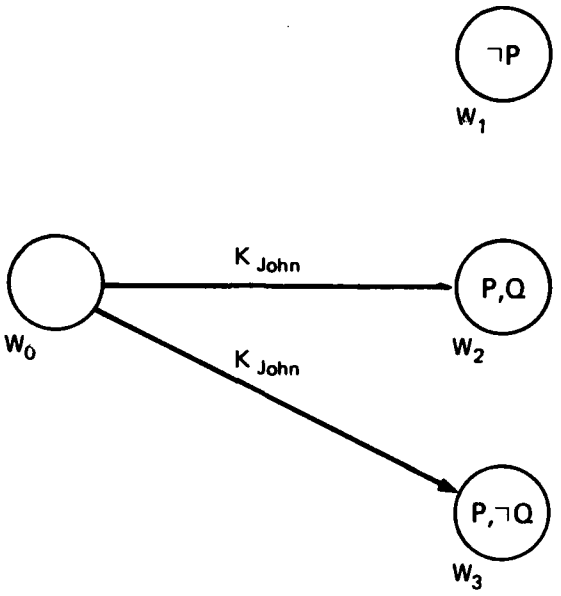


Figure 2.1 "John knows that P."
"John doesn't know whether Q."

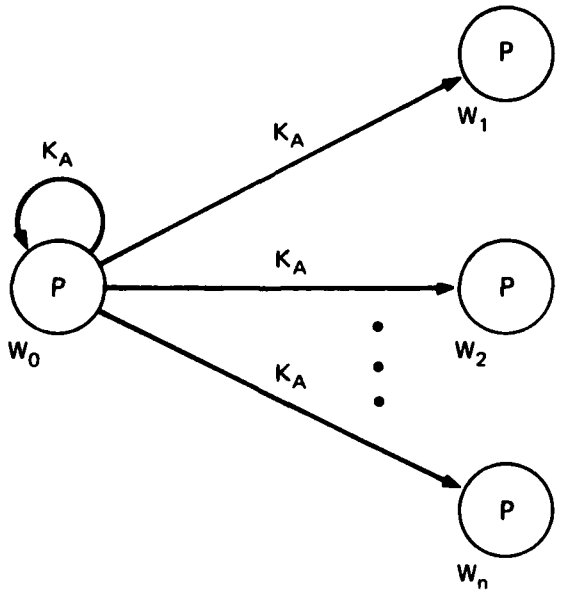


Figure 2.2 "A knows that P."
=
"P is true in every world which
is compatible with what A knows."

Since this assumption, like M4, is equivalent to saying that a person knows all the logical consequences of his knowledge, it should be interpreted only as a plausible implication. In a particular instance, the fact that P follows from A's knowledge would be a justification for concluding that A knows P. However, we should be prepared to retract the conclusion that A knows P in the face of stronger evidence to the contrary.

With this assumption, we can get the effect of M3, the axiom that if someone knows something, he knows that he knows it, by requiring that for any W_1 and W_2, if W_1 is compatible with what A knows in W_0 and W_2 is compatible with what A knows in W_1, then W_2 is compatible with what A knows in W_0. Formally this is:

$$\forall a_1,w_1,w_2(K(a_1,W_0,w_1) \supset (K(a_1,w_1,w_2) \supset K(a_1,W_0,w_2)))$$

By our previous assumption, the facts that A knows are the facts that are true in every world that is compatible with what A knows in the actual world. Furthermore, the facts that A knows that he knows are those that are true in every world that is compatible with what he knows in every world that is compatible with what he knows in the actual world. By the constraint we have just proposed however, all these worlds must also be compatible with what A knows in the actual world (see figure 2.3), so if A knows that P he knows that he knows that P.

Finally, we can get the effect of M5, the assertion that the basic facts about knowledge are themselves common knowledge, by generalizing these constraints so that they hold not only for the actual world but for all possible worlds. This follows from the fact that if these constaints hold for all worlds, they hold for all worlds that are compatible with what anyone knows in the actual world, and they hold for all worlds that are compatible with what anyone knows in all worlds that are compatible with what anyone knows in the actual world, etc. Therfore, everyone knows the facts about knowledge that the constraints

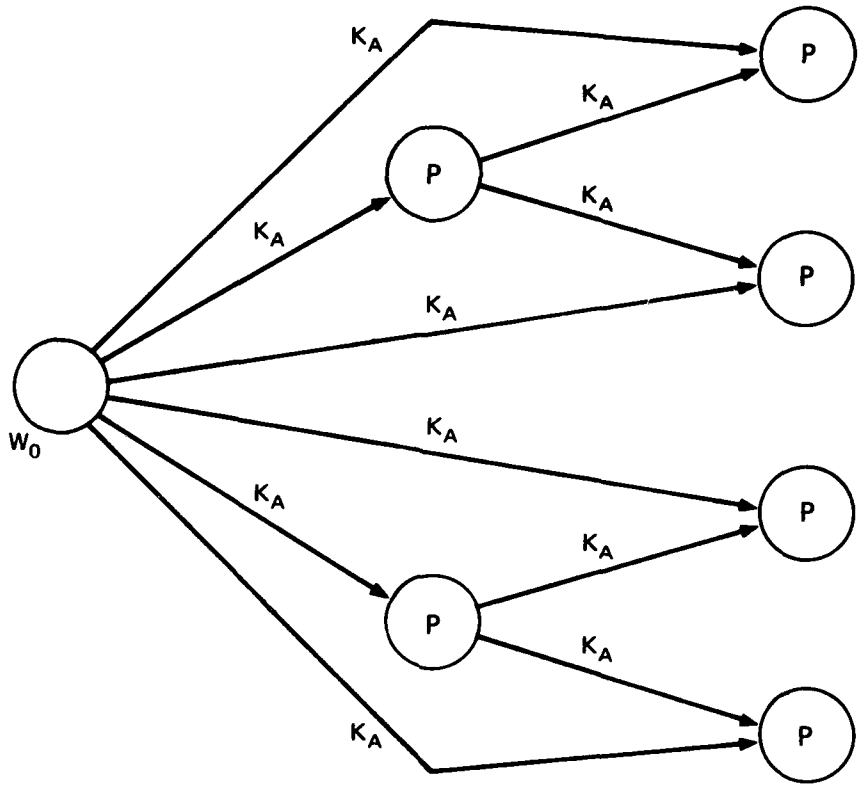


Figure 2.3 "If A knows that P, then he knows that P."

represent, and everyone knows that everyone knows, etc. Notice that this generalization has the interesting effect that the constraint that corresponds to M2 becomes the requirement that for a given knower, K is reflexive, and the constraint corresponding to M3 becomes the requirement that for a given knower, K is transitive. In other words, for each knower, K specifies a partial ordering on the set of possible worlds.

Analyzing knowledge in terms of possible worlds gives us a very nice treatment of knowledge about knowledge. Suppose John knows that Bill knows that P. Then if the actual world is W_0, in any world W_1 such that $K(John,W_0,W_1)$, Bill knows that P. We now continue the analysis relative to W_1, giving us that in any world W_2 such that $K(Bill,W_1,W_2)$, P is true. Putting both stages together, we get that for any worlds W_1 and W_2, if

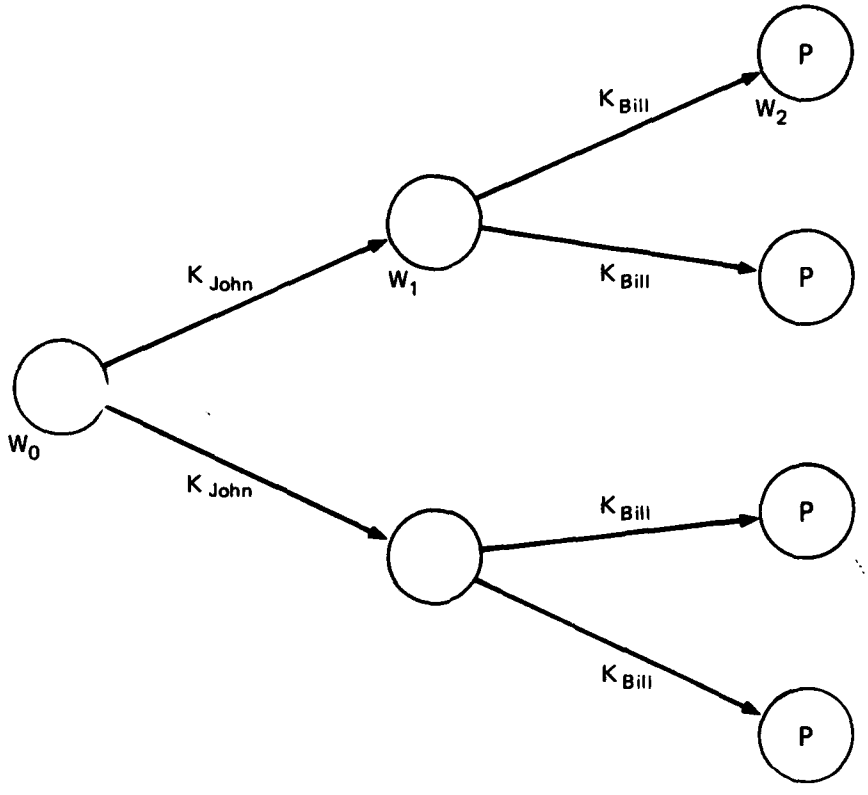


Figure 2.4 "John knows that Bill knows that P."

$K(John,W_0,W_1)$ and $K(Bill,W_1,W_2)$, then P is true in W_2. (See figure 2.4.) This is somewhat similar to the treatment of knowledge about knowledge in the data base approach. There we used chains of pointers between data bases to represent what one person knows about what another person knows. Here we are using chains of accessibility relationships between possible worlds for the same purpose.

Given these constraints and assumptions, whenever we want to assert or deduce something that would be expressed in the modal logic of knowledge by Know(A,P), we can instead assert or deduce that P is true in every world which is compatible with what A knows. We can express this in ordinary first-order logic, by treating possible worlds as

individuals (in the logical sense), so that K is just an ordinary relation. We will then introduce an operator T such that T(W,P) means that the formula P is true in the possible world W. If we let W_0 denote the actual world, then we can convert the assertion Know(A,P) into:

$$\forall w_1 (K(A,W_0,w_1) \supset T(w_1,P))$$

It may seem that we haven't made any real progress, since, although we have gotten rid of one nonclassical operator, Know, we have introduced another one, T. T, however, has an important property that Know does not. Namely, T "distributes" over ordinary logical operators. That is, $\neg P$ is true in W just in case P is not true in W, $(P \vee Q)$ is true in W just in case P is true in W or Q is true in W, etc. We might say that T is extensional, relative to a possible world. (The strict sense of extensionality requires that only the *actual* world be considered.) Thus, in contrast to Know, logical operators cannot become "trapped" inside of T where they are inaccessible to the ordinary inference procedures. This means that we can transform any formula so that T is applied only to atomic formulas. We can then turn T into an ordinary first-order relation by treating all the nonintensional atomic formulas as logical individuals. This is no loss to the expressive power of the language, since where we would have previously asserted P, we simply assert $T(W_0,P)$ instead.

In this way, we can transform a modal propositional logic with the nonstandard intensional operator Know into an ordinary first-order theory containing the relations K and T, and in which possible worlds and the atomic formulas of the modal logic are treated as individuals. It may seem that we have introduced notions such as possible worlds and formulas as individuals with too little regard for whether such things actually exist, i.e., without worrying whether the resulting theory is actually true. The answer to this type of objection is that from an AI point of view, it just doesn't matter. What we are seeking are

ways of creating systems that exhibit certain desired behaviors. Any notion that helps us achieve this goal is an acceptable analytical tool. We are not required to believe that possible worlds "really exist" for our systems to work any more than the electrical engineer who uses complex analysis is required to believe that imaginary numbers "really exist" for his circuits to work.

2.4 A Note on Belief

The ideas we have presented for formalizing statements about knowledge could easily be extended to handle the related concept of belief. We could give a modal axiomatization of belief very similar to the one for knowledge, the main difference being that there would be no analogue to M2, the axiom that states that anything that is known must be true. In corresponding fashion, we could define a possible-world semantics for this theory. This semantics and the one for knowledge would differ mainly in that the accessibility relation for belief would not be reflexive, since there is no reason to expect the actual world to be compatible with everything that someone believes. In other words, we would want to allow for false beliefs.

It might even be argued that we ought to take belief as the more fundamental notion and define knowledge in terms of belief. We have two reasons for not doing this, one theoretical and one practical. The theoretical reason is that it is not at all clear that knowledge can be defined in terms of belief. The idea that knowledge is simply true belief would probably not get us into trouble in the examples in this thesis, but it is certainly not correct in general. For example, a compulsive gambler who firmly believes that the number he has chosen will hit has no better claim to knowledge on those rare occasions when he guesses right than on the many occasions when he is wrong. Knowledge, therefore, is more than simply true belief. Exactly what else is required is one of the classical questions of

epistemology, and still has no generally accepted answer. Gettier (1963) has pointed out counter-examples to some widely held views.

The practical reason for not basing our formalism on the notion of belief is that we want to concentrate on issues relating to actions, and the effects of actions on belief are much harder to state than than the effects of actions on knowledge. The problem is that, while knowledge tends to be cumulative, belief does not. In we observe or perform a physical action, we generally know everything we knew before, plus whatever we have learned from the action. Similarly, if someone tells us something true, we usually gain new knowledge without having to give up any old knowledge.

It is true that some actions, like shuffling a pack of cards, can in a sense reduce our knowledge. But this depends on the frame of reference. If we know the order of a deck of cards at time t_1, and we shuffle the cards until t_2, we still know the order of the cards at t_1. What the shuffling does is to prevent us from acquiring some new knowledge, the order of the cards at t_2.

On the other hand, if the results of an action or the contents of a message contradict a previous belief, it is much harder to say what happens. If the new information is to be accepted, then certainly the contradictory belief must be given up. However, individual beliefs are part of complex belief structures which may have to undergo global adjustments in order to remove the discarded belief. Even true beliefs may be given up in this process, if they were based on other false beliefs (which underscores the inadequacy of "true belief"-type theories of knowledge). Furthermore, there is the problem of whether the new information will be accepted at all, since it contradicts what the person thinks he knows.

These issues are difficult enough for a system to handle in revising its own beliefs, where it is at least possible for the system to know the dependency structure of the beliefs (Doyle, 1978). To replace knowledge by belief in our system, however, would require the

system to reason about how some other agent would revise his beliefs, without necessarily knowing the relevant dependencies. People often deal with this problem by trying to find out what those dependencies are. For instance, in trying to persuade another person to give up some belief of his, someone might ask, "Why do you believe that?", so that he can argue against the basis for the belief. It is certainly an important area for research to try to devise systems with these capabilities, but it would lead us in a different direction from the one we want to follow in this thesis.

2.5 Knowledge, Equality, and Quantification

The formalization of knowledge presented so far is purely propositional. A number of problems are encountered when we attempt to extend the theory to handle equality and quantification. The first person to recognize the special problems that contexts such as knowledge and belief present for the logic of equality was Frege (1892). He pointed out that since the phrases "the morning star" and "the evening star" both refer to the planet Venus, according to Leibniz's principal of substituting equals for equals, for any sentence containing "the morning star", the corresponding sentence containing "the evening star" ought to have the same truth value. Yet this is not the case. The sentence "John knows that the morning star is a body illuminated by the sun," may be true, while "John knows that the evening star is a body illuminated by the sun," may be false, if John does not know that the morning star and the evening star are the same.

Frege's solution to this problem depends on distinguishing the denotation of an expression from its sense. The denotation of an expression is the object in the world to which the expression refers. In the case of "the morning star", the denotation would be Venus. The sense of an expression is an abstract entity "in which is contained the manner and context of presentation," (Frege, 1892, p. 86). Thus "the morning star" has a different

sense from "the evening star", because the first attempts to present an object as the star seen in the morning, while the second attempts to present an object as the star seen in the evening. We have to qualify these statements with the word "attempts", because a phrase can have a sense, and still not refer to anything. Frege gives the example of "the series with the least [i.e., slowest] convergence".

Having made this distinction, Frege goes on to assert that in indirect discourse, in which he includes knowledge and belief contexts, the denotation of a term is not its usual denotation. Instead, the denotation of a term in the context of indirect discourse is claimed to be the usual sense of the term. Since the usual senses of "the morning star" and "the evening star" are different, Leibnitz's law does not apply to them in the context "John knows that ..... is a body illuminated by the sun." Therefore the invalid inference which we were worried about cannot be made.

Frege does not go beyond this informal analysis to provide us with a logic of sense and denotation which we would need in order to use these ideas. Formally, what is required is that in the logic of knowledge, Know(A,P(B)) and (B = C) should not entail Know(A,P(C)). Frege's sense/denotation distinction seems to be adequate for this. However, we also want to account for the fact that Know(A,P(B)) and Know(A,(B = C)) does imply (at least plausibly) Know(A,P(C)). Frege gives us no help here.

The possible-world analysis of knowledge provides a very neat solution to this problem, once we realize that a term can denote different objects in different possible worlds. For instance, it might be possible that had the history of the solar system been different it would have been Mercury that was "the morning star" rather than Venus, or that two different planets that don't even exist in the actual solar system would have been "the morning star" and "the evening star". Thus, we will say that an equality statement such as (B = C) is true in a possible world W just in case the denotation of the term B in W is the same as the

denotation of the term C in W. This is a special case of the more general rule that a formula of the form $P(A_1,...,A_n)$ is true in W just in case the tuple consisting of the denotations in W of the terms $A_1,...,A_n$ in W is in the extension in W of the relation P. In other words, we fix the interpretation of "=" in all possible worlds to be the identity relation.

Now everything will work correctly. If Know(A,P(B)) and Know(A,(B = C)) are true, then in all worlds which are compatible with what A knows the denotation of B is in the extension of P and is the same as the denotation of C, hence the denotation of C is in the extension of P. But from this we can infer that Know(A,P(C)) is true. If (B = C) were true, but not Know(A,(B = C)), then the denotation of B would be the same as the denotation of C in the actual world, but not in all worlds which are compatible with what A knows, so the inference would not go through. Recalling our discussion in section 1.2 of McCarthy and Hayes's approach to this problem, we can see that we wouldn't need to have different expressions to refer to someone's idea of the combination of a safe and the actual combination as they propose. We would simply regard the denotation of the expression for the combination of the safe as depending on which possible world it is evaluated in. The denotation of that expression could well be different in the actual world than in the worlds which are compatible with what someone believes.

The introduction of quantifiers also causes problems. Suppose John is trying to repair a radio. Consider the sentence "John knows a transistor is burned out." This sentence has at least two interpretations. The first is that John knows that some transistor is burned out, but he does not necessarily know which one. The second interpretation is that there is a particular transistor which John knows is burned out. Sentences such as this were first studied by Russell (1905). He explained the ambiguity by analyzing sentences of the form "A P is Q," as "'P(x) and Q(x)' is sometimes true." In modern notation, we would write this as $\exists x(P(x) \wedge Q(x))$. So a sentence of the form "A transistor is Q," would be formally represented as $\exists x(\text{Transistor}(x) \wedge Q(x))$.

Russell goes on to point out that in sentences of the form "John knows a P is Q," the rule for eliminating the phrase "a P" can be applied either to the whole sentence, or only to the subordinate clause, "a P is Q." Applying this observation to "John knows a transistor is burned out," gives us the following two formal representations:

(1) Know(John,∃x(Transistor(x) ∧ Burned-out(x)))

(2) ∃x(Transistor(x) ∧ Know(John,Burned-out(x)))

The most natural English paraphrases of these formulas are "John knows that there is a burned out transistor," and "There is a transistor which John knows is burned out." These seem to correspond pretty well to the two interpretations which we identified for the original sentence. So, the ambiguity in the original sentence is mapped into an uncertainty as to the scope of the operator Know.

There is another possible interpretation of "John knows a transistor is burned out," which isn't accounted for by Russell's theory of how English sentences of the form "A P is Q." are expanded, but which can be represented in this notation, namely:

(3) ∃x(Know(John,(Transistor(x) ∧ Burned-out(x)))).

(3) can be read as "There is something that John knows to be burned out and (knows) to be a transistor." The difference between (2) and (3) is that (3) asserts that John knows that the thing which he knows is burned out is a transistor, while (2) simply asserts that it *is* is a transistor without asserting whether John knows that it is. Thus (2) is weaker than (3) in that it would be true if John knew that a particular transistor was burned out without knowing that it was a transistor. He might know nothing about electronic components but see smoke rising from a certain object and say to himself, "That thing (whatever it is) is burned out." In this case, it would be correct to say that he knows of a particular transistor

that it is burned out, and he knows that something is burned out, but he does not know that it is a transistor that is burned out.

Following a suggestion of Hintikka (1962), we can use a formula similar to (2) or (3) to express the fact that someone knows who or what something is. He points out that a sentence of the form "A knows who (or what) B is," intuitively seems to be equivalent to "there is someone (or something) that A knows to be B. But this can be represented formally as $\exists x(Know(a,(B = x)))$. To take a specific example, "John knows who the President is," can be paraphrased as "There is someone whom John knows to be the President," which can be represented by:

(4) $\exists x(Know(John,(President = x))$

In (1), Know may still be regarded as a purely propositional operator, although the proposition to which it is applied now has a quantifier in it. Put another way, Know still is used simply as a relation between a knower and the proposition he knows. (2) and (4) are not so simple. In these formulas there is a quantified variable that is bound outside the scope of the Know operator, but has an occurrence inside. This situation is usually called "quantifying in", and it creates problems for the formal interpretation of Know as a relation between a knower and a proposition.

Consider trying to apply the usual Tarskian notion of satisfiability (Rogers, 1971) to (2). This is of the form $\exists x(P)$, so we must bind x to an individual that makes P true. In this case P is a conjunction, so the value of x must satisfy both conjuncts of P. The first conjunct is Transistor(x), which we chose to represent the fact that the value of x is a transistor, a physical object. For the second conjunct, Know(John,Burned-out(x)), to be true, Burned-out(x) must denote a proposition. The value of x has to be a physical object to satisfy the first conjunct, but Frege's argument shows that Burned-out(x) does not determine

a proposition unless the value of x tells us how the object is identified. The analysis we have does not supply us with any such description, so we are stuck; the Tarskian definition of satisfiability doesn't work here.

The possible-world analysis, however, provides us with a very natural interpretation of quantifying in. We keep the standard interpretation that ∃x(P) is true just in case there is some value for x that satisfies P. If P is Know(A,Q) then a value for x satisfies P just in case that value satisfies Q in every world that is compatible with what A knows. So (2) is satisfied if there is a particular transistor which is burned out in every world that is compatible with what John knows. That is, in every such world, the same transistor is burned out. On the other hand, (1) is satisfied if in every world compatible with what John knows there is some burned out transistor, but it doesn't have to be the same one in every case. In either situation, there is no problem about determining a proposition from a physical object, because we do not speak of propositions. We simply talk about various possible worlds and which transistors are burned out in those worlds.

This analysis does require us to talk about the same individual existing in several different possible worlds, which may seem unintuitive, but as Kripke (1972) has pointed out this is a common feature of ordinary discourse. When we say that Humphrey would have won the 1968 Presidential election if he had only done such-and-such, we are really asserting that there is some other course of events (i.e., another possible world) in which Humphrey would have done such-and-such and therefore have won the election. Furthermore, we really do mean Humphrey, the very same individual who in fact lost the election, when we talk about this other possible world. Of course, there are other possible worlds in which Humphrey does not exist. The best way to think about this is in terms of one universal domain of possible individuals with the domains of particular possible worlds being subsets of that domain.

Notice that the difference between (1) and (2) has been transformed from a difference in the relative scopes of an existential quantifier and the operator Know to a difference in relative scopes of an existential and a universal quantifier (the "every" in "every possible world compatible with..."). Recall from ordinary first-order logic that $\exists x(\forall y(P(x,y)))$ entails $\forall y(\exists x(P(x,y)))$ but not vice versa. The possible-world analysis, then, implies that we should be able to infer "John knows that something is burned out," from "There is a transistor that John knows is burned out," as indeed we can.

When we look at how this analysis applies to our representation for "knowing who" we get a particularly nice picture. We said that A knows who B is means that there is someone whom A knows to be B. If we analyze this we conclude that there is a particular individual who is B in every world that is compatible with what A knows. Suppose this were not the case, and in some of the worlds compatible with what A knows one person is B and in the others, some other person is B. In other words, for all that A knows, either of these two people might be B. But this is exactly what we mean when we say A *doesn't* know who B is! Basically, the possible-world view gives us the very natural picture that A knows who B is if A has the possibilities for B narrowed down to a single individual.

There is at least one more consequence of this analysis that is worth noting. Suppose that A knows who B is and who C is. Then the denotation of B is the same in all the worlds which are compatible with what A knows, and the same is true for C. Since in all these worlds, B and C each have only one denotation, they either have the same denotation everywhere or different denotations everywhere. Thus, either $(B = C)$ is true in every world compatible with what A knows or $(B \neq C)$ is. From this we can infer that either A knows that B and C are the same individual or that they are not. The end conclusion is that if A knows who both B and C are, he must know whether they are the same person, another very intuitive result.

We now have a coherent account of quantifying in that does not talk about knowing particular propositions. Still, in many cases there will be a certain proposition such that knowing that proposition counts as knowing something which we would express by quantifying in. For instance, the proposition that John knows that Bill's telephone number is 321-1234 might be represented as:

(5) Know(John,(Phone-num(Bill) = 321-1234)),

which does not involve quantifying in. We want to be able to infer from this, however, that John knows what Bill's telephone number is, which would be represented as:

(6) ∃x(Know(John,(Phone-num(Bill) = x))).

It might seem that (6) can be derived from (5) simply by the logical principle of existential generalization (EG), but the situation is more complicated than that. Suppose that (5) were not true, but instead, John simply knew that Bill and Mary had the same telephone number. We could represent this as:

(7) Know(John,(Phone-num(Bill) = Phone-num(Mary))).

It is clear that we would not want to infer from (7) that John knows Bill's telephone number, yet if we can get (6) from (5) by EG, then we ought to be able to get (6) from (7) by the same process.

To take another example, suppose there is a collection of blocks that John knows something about. If John knows that the number of cubes is greater than ten, then there is a number such that John knows that the number of cubes is greater than that number. If, on the other hand, all that John knows is that the number of cubes is greater than the number of pyramids, then there may not be any number such that John knows the number of cubes is greater than that number.

It seems then that EG can be applied to occurrences in knowledge contexts of the terms which represent "321-1234" and "ten" but not the terms which represent "Mary's telephone number" and "the number of pyramids". What is the difference in these cases? The difference seems to be that "321-1234" and "ten" are *standard names* for the things they refer to, whereas "Mary's telephone number" and "the number of pyramids" are not. A standard name can be thought of as a name such that knowing what the name denotes is part of knowing the language that the name occurs in. Thus, not to know which number is the number of cubes in a certain collection is to be ignorant of a certain feature of the world. Not to know which number is ten is to be ignorant of part of English, namely the meaning of the word "ten".

Now we can show formally why EG works for standard names in knowledge contexts even though it doesn't work in general. Suppose John knows that P(B) is true, where B is a standard name:

(8) Know(John,P(B))

Since B is a standard name, it is part of knowing the language to know what B is; so we will assume that everyone, including John, knows what B is:

(9) $\exists x$(Know(John,(B = x)))

By ordinary first-order logic, we can conjoin (8) and (9), bringing (8) inside the scope of the quantifier in (9). This is valid because (8) does not contain any free occurrences of the variable in (9):

(10) $\exists x$(Know(John,(B = x) $\wedge$ Know(John,P(B)))

Now, using the results on equality substitution that we developed in the first part of this section, we can substitute x for B in P(B):

(11) ∃x(Know(John,P(x)))

It should be noted that we are not claiming that the only way of knowing who or knowing what is to know a proposition that contains a standard name. For instance if John picks up an unusual rock and puts it in his pocket, we would not want to claim that John doesn't know what is in his pocket just because there is no standard name in the language for that particular rock. To say exactly what the other ways of knowing who or knowing what are is one of the basic problems of epistomology, and is, therefore, beyond the scope of this thesis. As with the concept of knowledge itself, we have a formalism that makes some intuitively plausible predictions, and any epistemological theory that explains these prediction would be acceptable.

In terms of possible worlds, standard names have a very straightforward interpretation. Standard names are simply terms that have the same denotation in every possible world. If the denotation of a standard name is fixed by the language alone, then no matter what possible world we are talking about, the name must have that denotation. Following Kripke (1972), we will call terms that have the same denotation in every possible world *rigid designators*. The conclusion that standard names are rigid designators seems inescapable. How could any expression be a standard name, a canonical identifier, of an individual if under some circumstances that expression refers to something else?

The validity of EG for standard names follows immediately from this definition. The possible world analysis of Know(John,P(B)) is that in every world which is compatible with what John knows, the denotation of B in that world is in the extension of P in that world. EG fails because we are unable to conclude that there is any particular individual which is in the extension of P in all the relevant worlds. If B is a rigid designator, however, the denotation of B is the same in every world, so it is the same in every world compatible with what John knows, and that denotation is an individual which is in the extension of P in all those worlds.

In addition to rigid designators which are simple constants, we will also need to have terms built up using *rigid functions*. Rigid functions will have the property that if the arguments to the function are rigid designators, then the term consisting of the function applied to the arguments is also a rigid designator. We will make considerable use of this notion in section 3.2 when we formalize the notion of an action having knowledge preconditions. If $Act(x_1,...x_n)$ represents a general action that we assume anyone can perform, then we will treat Act as a rigid function. In our possible-world semantics for actions, this will have the consequence that an agent knows how to perform some specific instance of Act, just in case he knows what individuals the arguments of Act refer to.

Saying that a standard name is a term whose denotation is determined by the language it occurs in leaves open the question, "What language?" It would be possible to have two languages that were identical in syntax and semantics, except that some of the terms which were standard names in one language were not standard names in the other. In fact, this happens all the time. The linguistic community comprising the users of any natural language will contain subcommunities who use certain terms as standard names that are not shared by the larger community. This is done more or less formally in professional and scientific disciplines, and informally in other contexts.

This is an important point for AI systems, because they usually assume that there is a common vocabulary shared by the system and the user that goes beyond the bare essentials of the basic language. For example, in systems for analysis of electronic circuits [e.g., (Stallman and Sussman, 1976), (Brown, 1977)], individual components and nodes are typically assigned identifiers that function as standard names. If Q301 is such an identifier and the transistor it refers to is burned-out, then the system knows which transistor is burned out only if it knows that Q301 is burned out. We will make use of this notion of specialized vocabularies of standard names in our examples. Standard names for objects

mentioned by an example will be freely introduced whenever the identity of that object is not directly relevant to the point which we are using the example to illustrate. It should be noted, however, that this is never essential for making the examples work. We could always eliminate the assumption that the identifiers are standard names by adding explicit assertions that the agent in the example knows what objects the identifiers refer to.

There are a few more observations to be made about standard names and rigid designators. First, in describing standard names we assumed that everyone knew what they referred to. Identifying them with rigid designators makes the stronger claim that what they refer to is common knowledge. That is, not only does everyone know what a particular standard name denotes, but everyone knows that everyone knows, etc. Second, although it is natural to think of any individual having a unique standard name, this is not required by our theory. What the theory does require is that if there are two standard names for the same individual, it will be common knowledge that they name the same individual.

Finally, there is a question about how a rigid designator can refer to the same individual in *all* possible worlds, when that individual may not exist in some of those worlds. At first glance, the idea that the denotation of a term in a possible world could be something that does not exist in that world seems paradoxical, but an examination of ordinary discourse shows that we very often say things which are most naturally analyzed in this way. We can talk about things like "the largest tower that could be built out of these six blocks," and if there is only one way of arranging the blocks to fit this description, it will denote a well-defined possible, though nonexistent, individual. In fact, we frequently quantify over possible individuals as well, as when we ask how many possible towers could be constructed with a certain group of blocks, or whether any of them would be over ten inches high.

If we let the quantifiers in our formalism range over possible individuals, then we can

allow rigid designators for possible individuals that do not actually exist and still allow existential generalization over rigid designators, without "precipitating an ontological crisis". If we want to treat Santa-Claus as a rigid designator, then we can infer:

(12) ∃x(Believes(John,Lives-at(x,North-Pole)))

from:

(13) Believes(John,Lives-at(Santa-Claus,North-Pole))

without claiming that Santa Claus actually exists. We are merely claiming that Santa Claus might have existed.

In such a system, since existential quantifiers indicate possible existence, to talk about actual existence we would need a predicate whose extension in each possible world is the subset of all the possible individuals who actually exist in that world. Most universally quantified statements, however, would not need modification so long as ordinary predicates are restricted to actual individuals. Thus, "All men are mortal," could still be true even if there are possible men who are immortal, so long as the predicate "men" picks out just the men who actually exist in the possible world that the sentence is evaluated in.

2.6 Other Work on Reasoning about Knowledge

Although the work cited in section 1.1 seems to be the only previous work in AI to consider the interaction of knowledge and action, there has been slightly more done on reasoning about knowledge alone. Most of this work is due to John McCarthy and his students, much of it unpublished. As we mentioned in section 1.2, in addition to their unsatisfactory attempt to formalize knowledge prerequisites for actions, McCarthy and Hayes (1969) review Hintikka's work on the logic of knowledge and make the point that the

possible-world semantics for this logic could be formalized in first-order logic. This seems to be the first time that this idea appears in the AI literature, but they do not pursue the approach. They do make a tantalizing reference to another idea that is crucial to our efforts to integrate knowledge and action, namely identifying possible worlds in the semantics for knowledge with situations in their formalism for actions. They give no examples that make use of this identification, however, so it is not clear that they have in mind the same interpretation that we will use. Subsequently McCarthy (1978) recalled that they abandoned this idea, because they could not see how to express that someone knows the effect of an action if possible worlds and situations are identified. As we will see in chapter 3, our approach does not suffer from this problem.

Sato (1976) uses some techniques due to Gentzen to prove some very general results about the soundness, completeness, and decidability of various propositional modal logics with respect to Kripke-style possible-world semantics. He then applies these results to some puzzles in the logic of knowledge, using an axiomatization due to McCarthy. Since we will be using the model theory directly in our system, his results on the properties of the modal logics themselves do not seem directly relevant to our work. Sato's work is interesting, however, in that he does integrate a simple logic of time into his logic of knowledge. Because he does not identify possible worlds with possible situations, though, all formulas in his system are required to be definite as to time. In chapter 3 we will explain why this is the case and what problems it creates.

In an unpublished note, McCarthy (1975) uses the possible-world semantics developed by Sato to construct a first-order axiomatization of knowledge in exactly the same way that we will use Hintikka's. He then uses this axiomatization to give a formal proof of the solution to a puzzle involving reasoning about knowledge. Goad (1976) uses the same ideas to formalize some interesting problems in reasoning about *lack* of knowledge which we will

discuss in chapter 8. As we have mentioned, the main limitation of this work is that it deals only with the propositional part of the logic of knowledge, while handling quantifiers and equality will be essential for the problems we wish to solve.

More recently McCarthy (1979) (summarized in McCarthy (1977b)) uses a completely different approach to handle the problem of referential transparency in knowledge contexts. He proposes that the notion of knowing a phone number, for instance, be regarded as a relation between the knower and the *concept* of the phone number, rather than as a relation between the knower and the phone number itself. This allows McCarthy to account for the fact that although Mary and Bill may have the same phone number, it does not follow from the fact that John knows Mary's phone number that John knows Bill's phone number. This point is that while the phone numbers are the same, the concept of Mary's phone number is distinct from the concept of Bill's phone number. Thus knowing one of them does not imply knowing the other.

This is just the problem described by Frege (1892) which we discussed in section 2.5, and McCarthy's concept/object distinction seems to be essentially the same as the sense/denotation distinction which Frege proposed as a solution to the problem. McCarthy's proposal is interesting, though, because he formalizes these ideas entirely within first-order logic by treating concepts as individuals. McCarthy gives many examples of representations of various types of statements, but since he axiomatizes few of the properties of knowing and presents few deductions in the system, it is difficult to evaluate these methods.

None of the work cited above deals with the problem of automatically generating deductions about knowledge. The only previous work which comes close to this problem is by Morgan (1976). Morgan deals with purely abstract modal logics, rather than the logic of knowledge specifically, but the general issues are the same in either case. He presents two methods for using standard theorem proving techniques to prove theorems in modal logic.

One of these is to axiomatize the possible-world semantics of the logic in exactly the same way as is done here and in McCarthy's formalism. The other method, which he calls the syntactic method, is to make sentences of the modal logic into terms in a first-order logic, and introduce the predicate PR(P) which means that P is provable in the modal logic. The only axioms necessary are one axiom of the form PR(P) for each axiom or axiom schema P in the modal logic, and one axiom for each rule of inference in the modal logic. For instance, $(PR(implies(p_1,p_2)) \supset (PR(p_1) \supset PR(p_2)))$ would represent modus ponens. Morgan then feeds these axioms to a simple resolution theorem prover, and for any formula P which he wishes to prove in the modal logic, he tries to prove PR(P) in the resolution system.

The trouble with this approach is that it suffers the same problems of efficiency that we saw in section 2.2 when we explored the consequences of adding modal logic axioms to a standard deduction system. Morgan's idea runs into the same difficulty that most of the *control* of the deductive process actually resides in the axioms and rules of the modal logic rather than in the theorem prover, and, as we pointed out before, these formalisms are usually designed to be concisely stated, rather than to be efficient in generating proofs. To see what can happen in this type of situation, consider using the axiom which describes modus ponens to try to prove an arbitrary goal P. Resolving this goal against that axiom would produce a new goal of trying to find some other formula Q, such that Q and $(Q \supset P)$ can be proved. Whichever of these goals we attack first, the modus ponens axiom applies again, producing still more complicated goals. It is possible to continue in this way to an arbitrary depth without ever touching ground, so to speak.

Morgan does not seem to be aware of the possibilities for this sort of behavior, although he does note somewhat innocently that he was able to prove certain theorems using the possible-world approach that he had not been able to prove using the syntactic approach. What is really disappointing about Morgan's work, however, is that he gives no suggestions

for efficient use of either approach other than to simply turn loose a uniform resolution theorem prover on them. The main point of chapters 6 and 7 of this thesis will be to try to improve on that idea.

3. An Integrated Theory of Knowledge and Action

3.1 Possible-World Semantics for Actions

In the preceding sections, we presented a theory for talking about knowledge in terms of possible worlds. If we are to capture the interactions between knowledge and action, we need a theory of actions in these same terms. Happily, the standard AI way of looking at actions gives us exactly that. Most AI programs that reason about actions view the world as a set of possible situations, where each action determines a binary relation on situations - one situation being the outcome of performing the action in the other situation. We will integrate knowledge and action by identifying the possible worlds that are used to describe knowledge with the possible situations that are used to describe actions.

The identification of possible worlds with situations is somewhat nonstandard in possible-world semantics. Usually a possible world is thought of as including an entire course of events. For example, we might say that in some possible worlds the European discovery of America occurs 100 years later than it actually did. It might seem that taking possible worlds to be situations, and therefore not extended in time, might make it difficult to talk about what someone knows about the past or future. That is not the case, however. Knowledge about the past and future can be handled by modal tense operators which have corresponding accessibility relations on possible situation/worlds. We could have a tense operator Future, such that Future(P) means that P will be true at some time to come. If we let F be an accessibility relation such that $F(W_1,W_2)$ means that the situation/world W_2 lies in the future of the situation/world W_1, then we can define Future(P) to be true in W_1 just in case there is some W_2 such that $F(W_1,W_2)$ holds and P is true in W_2.

This much is standard tense logic, as in Rescher and Urquhart (1971). The interesting

point is that statements about someone's knowledge of the future work out exactly right, even though knowledge is analyzed in terms of alternatives to a situation, rather than alternatives to a course of events. The proposition that John knows that P will be true is represented simply by Know(John,Future(P)). The analysis of this is that Future(P) is true in every situation which is compatible with what John knows, from which it follows that, for each situation which is compatible with what John knows, P is true in some future alternative to that situation. An important point to note here is that two situations can be "internally" similar (that is, they agree in truth value for all nonmodal statements), but be distinct because they differ in their accessibility relations to other possible situations. So although we treat a possible world as a situation rather than a course of events, it is a situation in the particular course of events defined by its relationships to other situations.

It turns out that treating possible worlds as situations is actually a more flexible way of handling time than treating possible worlds as courses of events. If, following Sato (1976), we formalize possible worlds as extending over time and identify a person's knowledge with the set of propositions which are true in every possible world which is compatible with what he knows, then all propositions will have to be specific as to what times they refer to. Thus, "A is on B," would not be a proposition because it does not have a unique truth value in each possible world. In some worlds it will be true at some points in time and false in others. So, only sentences like "A is on B at time T," would express definite propositions. As a result, we could not say that at time T (or in situation W) John knows that A is on B. Instead we would have to say that at time T John knows that at time T A is on B. Furthermore, since a person can know something at one time that he did not know at an earlier time, the accessibility relation for knowledge, K, will have to have an extra argument position for the time in question.

For a planning system this distinction is extremely important. Suppose the system has

the goal of bringing about P, and it knows that if Q is true, performing Act will cause P to be true. If its knowledge about the truth of Q is limited to statements of the form Q is true at time T, there is a problem, because the system has no way to represent what time T is with respect to the time at which the system is doing its planning. What it needs to know is that Q is true *now*, i.e., Q is simply true, but this is not representable in the logic. The logic cannot have a formula such as T = Now either, since this is not a statement that can be true at all times in a given possible world. Of course, the effect of having such a fact can be built into the planner, but it is necessary to go outside of the logic to do it.

For reasoning about actions, instead of a tense operator like Future, which simply says what will be true, we need an operator that talks about what would be true if a certain action were performed. Our approach will be to recast McCarthy's situation calculus (McCarthy, 1963), (McCarthy and Hayes, 1969), to mesh with our possible-world approach to reasoning about knowledge. The situation calculus is a first-order language in which predicates which can vary in truth value over time are given an extra argument to say what situations they hold in, and there is a function Result that maps an agent, an action, and a situation into the situation which results from the agent performing the action in the first situation. Statements about the effects of actions are then expressed by formulas like P(Result(A,Act,S)), which means that P is true in the situation that results from A performing Act in situation S.

In order to integrate these ideas into our logic of knowledge, we will redefine the situation calculus as a modal logic. We will introduce a modal operator Res for talking about the results of actions, parallel to the modal operator Know for talking about knowledge. Situations will not be referred to explicitly in this language, but they will reappear when we specify the possible-world semantics for Res and formalize that semantics in first-order logic. We will let Res take as its arguments a description of an event and a

formula, such that Res(Ev,P) means that if the event described by Ev occurs, the formula P will then be true. The possible-world semantics for Res will be specified in terms of an accessiblity relation R, parallel to K, such that $R(:Ev,W_1,W_2)$ means that W_2 is the situation/world that would result from the event :Ev happening in W_1. We need to distinguish between expressions that represent event descriptions (e.g., Ev) and expressions that represent events (e.g., :Ev), because the same event description may refer to different events in different possible worlds. (Generally, if X is a symbol in the modal language, :X will be the corresponding symbol in the possible-world language.) For example, $Dial(Combination(Sf_1))$ will refer to different sequences of dial twisting in worlds where the combination of Sf_1 differs.

We will assume that if it is impossible for :Ev to occur in W_1 (i.e., the preconditions of :Ev are not satisfied), then there is no W_2 such that $R(:Ev,W_1,W_2)$ holds. Otherwise, we assume that there is exactly one W_2 such that $(:Ev,W_1,W_2)$ holds. Formally, this amounts to an assumption that all events are deterministic, which might seem to be an unnecessary limitation. Pragmatically, however, it doesn't matter whether we say that a given event is nondeterministic, or that it is deterministic, but no one knows precisely what the outcome will be. If we treated events as being deterministic, we could say that someone knows exactly what situation he is in, but doesn't know what situation would result if :Ev occurs, because :Ev is nondeterministic. It would be completely equivalent, however, to say that :Ev is deterministic, and that this person *doesn't* know exactly what situation he is in because he doesn't know what the result of :Ev would be in that situation.

Giving a possible-world semantics for Res requires specifying which possible worlds a formula of the form Res(Ev,P) is true in. (If we want to say that Res(Ev,P) is simply true, we have to say that it is true in the actual world.) With the assumptions we have just made,

we can say that Res(Ev,P) is true in W_1 just in case there is some W_2 which is the situation/world that results from the event described by Ev happening in W_1, and in which P is true. Because this definition involves an existential quantifier, we get a strong interpretation of Res in that it must be possible for the event described by Ev to occur for Res(Ev,P) to be true. There is a corresponding weak interpretation which is noncommital as to whether it is possible for Ev to occur, but asserts that if it did, P would be true in the resulting situation. This weaker interpretation is obtained by saying that P must be true in every situation that results from the occurence of the event described by Ev. We will give this weaker interpretation to the operator Resl.

With this definition, we can develop a theory based on Res and Resl parallel to our theory of Know, and then convert the theory into first-order logic. An instance of Res(Ev,P) whose truth is to be determined relative to W_1 will be replaced by the corresponding instance of:

$$\exists w_2(R(:Ev,W_1,w_2) \wedge T(w_2,P)),$$

and an instance of Resl(Ev,P) whose truth is to be determined relative to W_1 will be replaced by the corresponding instance of:

$$\forall w_2(R(:Ev,W_1,w_2) \supset T(w_2,P)).$$

Both of these can be handled in the same way as the possible-world transformations of formulas containing Know.

The type of event we will normally be concerned with is an agent performing an action. We will let Do(A,Act) be a description of the event consisting in the agent named by A performing the action named by Act. We will assume that the set of possible agents is the same as the set of possible knowers. Do will be a rigid function, so Do(A,Act) will be the

standard name of an event if A is the standard name of an agent and Act is the standard name of an action.

It would be more precise to say that Do(A,Act) names a type of event rather than an individual event, since an agent can perform the same action on different occasions. We would then say that Res and R are relations on event types. We will let the present usage stand, however, since we will not need to distinguish individual events in this thesis.

Most actions can be thought of as a general procedure applied to some specific objects. These general procedures will be represented by functions which map the objects the procedure is applied to into the action of applying the procedure to those objects. For instance, if Dial represents the general procedure of dialing combinations of safes, C_1 represents a combination, and Sf_1 represents a safe, then $Dial(C_1,Sf_1)$ represents the action of dialing the combination C_1 on the safe Sf_1.

This formalism gives us the *ability to talk about someone's knowing* about the effects of an action. In the modal logic, we can express the assertion that A_1 knows that P would result from A_2 doing Act as $Know(A_1,Res(Do(A_2,Act),P))$. The possible-world analysis of this statement is that in every world that is compatible with what A_1 knows, there is a world which is the result of A_2 doing Act and in which P is true (see figure 3.1). Formally, this is expressed by:

$$\forall w_1 (K(:A_1,W_0,w_1) \supset \exists w_2 (R(:Do(:A_2,:Act),w_1,w_2) \wedge T(w_2,P))),$$

assuming that A_1, A_2, and Act are rigid designators. As with event descriptions and events, we distingush between terms in the modal logic such as A_1, A_2, and Act and terms in the possible-world notation such as $:A_1$, $:A_2$, and :Act. In general, terms in the modal logic may correspond to different terms in the possible-world notation depending on what possible world they are evaluated in. This is discussed in more detail in section 4.3.

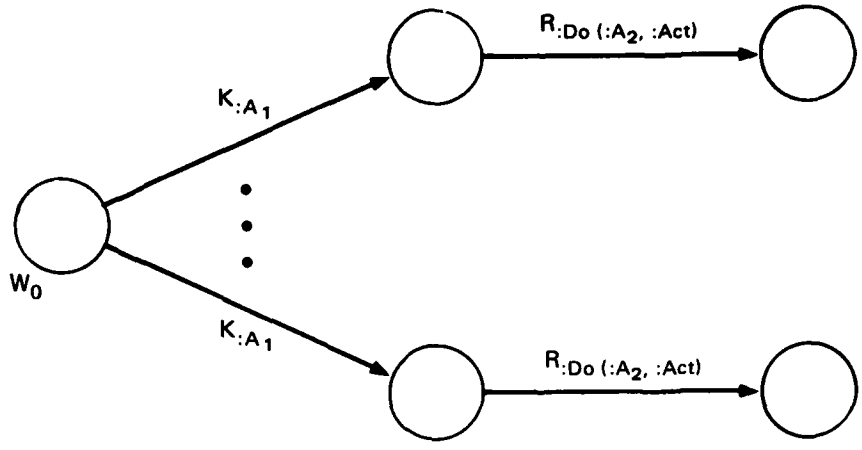


Figure 3.1 Know $(A_1, Res(Do(A_2, Act), P))$

$$=$$

$$\forall w_1 (K(:A_1, W_0, w_1) \supset \exists w_2 (R(:Do(:A_2, :Act), w_1, w_2) \wedge T(w_2, P)))$$

McCarthy and Hayes ran into difficulty with identifying possible worlds with situations because they wanted to express knowledge about the effects of actions in terms of knowing formulas of the situation calculus (McCarthy, 1978). This requires allowing occurrences of terms that denote situations inside the modal operator Know. The problem is how to relate these terms to the references to possible worlds that are introduced when an occurrence of Know is eliminated. From our point of view, this problem is the result of confusing two different levels of language. In the modal notation we do not have terms denoting situations. At this level, all talk about the effects of actions is in terms of the modal operators Res and Resl. All references to situation/worlds are introduced in transforming the modal notation into possible-world notation. Thus, we never have the problem of introducing references to possible worlds in the analysis of a formula that already refers to situations.

In addition to simple, one-step actions, we will want to talk about complex combinations of actions. To facilitate this, we will introduce sequences, conditionals, and iterations. If P

is a formula, and Act_1 and Act_2 name actions, then $(Act_1; Act_2)$, $If(P,Act_1,Act_2)$, and $While(P,Act_1)$ also name actions. The result of A doing $(Act_1; Act_2)$ in W_1 will be W_2 just in case there is some situation W_3 such that W_3 is the result of A doing Act_1 in W_1 and W_2 is the result of his doing Act_2 in W_3. That is, doing $(Act_1; Act_2)$ is equivalent to doing Act_1 and then doing Act_2. The result of A doing $If(P,Act_1,Act_2)$ in W_1 will be W_2 just in case P is true in W_1 and the result of A doing Act_1 in W_1 is W_2, or P is false in W_1 and the result of A doing Act_2 in W_1 is W_2. This means that doing $If(P,Act_1,Act_2)$ is equivalent to doing Act_1 or Act_2 depending on P. The result of A doing $While(P,Act_1)$ in W_1 will be W_2 just in case the result of A doing $If(P,(Act_1; While(P,Act_1)),Nil)$ in W_1 is W_2, where the result of A doing Nil in W_1 is W_1. In other words, doing $While(P,Act_1)$ is equivalent to doing Act_1 followed by $While(P,Act_1)$ if P is true, otherwise doing nothing, i.e., doing Act_1 as long as P remains true.

The choice of programming language constructs for sequences, conditionals, and iterations is more than coincidental. If the references to agents are eliminated and possible situations/worlds are identified with machine states, then these rules amount to a partial specification of the semantics of an imperative (Algol-like) programming language. In fact, this approach to formalizing the semantics of complex actions is based on some ideas of V. R. Pratt and this author for formalizing the semantics of programs in modal logic. Pratt and his associates have used this approach to develop a powerful formalism for talking about the semantics of programs which they call *dynamic logic* (Pratt, 1976), (Harel, Meyer, and Pratt, 1977).

To continue the analogy between actions and programs, we can think of the distinction between our strong and weak operators, Res and Res1 in terms of the program-verification notions of total and partial correctness. Since Res(Ev,P) is defined to be true just in case P is

true in at least one possible outcome of Ev and we are assuming determinism, Res expresses both termination and correctness. On the other hand, since Resl(Ev,P) is true when P is true in every possible outcome of Ev, Resl should be satisfied when, due to nontermination, there are no possible outcomes. This corresponds to the notion of partial correctness in the semantics of programs.

The rules that we have given so far are not sufficient to prove that there is no situation which is the result of a nonterminating iterative action, but we can remedy this by adding a possible-world version of Hoare's (1969) rule for partial correctness of While loops. In our terms, the rule is that if Q is true in every situation that might result from A doing Act_1 in a situation where P and Q are true, then Q is true and P is false in every situation that might result from A doing While(P,Act_1) in a situation where Q is true. Intuitively, While(P,Act_1) will not terminate if it is executed in a situation where some condition Q holds that always implies P and is never changed by doing Act_1. What we want to show is that in such a situation, there is no situation that is the result of carrying out While(P,Act_1). This follows immediately from the rule we have just stated, because if Q is invariant with respect to Act_1 then Q will be true and P will be false in every situation that might result from doing While(P,Act_1), but since Q always implies P, P would have to both true and false in any such situation. Therefore, no such situation can exist.

Our possible-world semantics for actions leads us to say that a primitive action description Act denotes some primitive action in each possible world. (If Act is a rigid designator it will name the same action in every possible world.) This leads us to ask what the denotation of complex action descriptions such as (Act_1; Act_2), If(P,Act_1,Act_2), and While(P,Act_1) might be. The most natural answer consistent with the treatment of primitive actions is that the denotation of a complex action description is the sequence of primitive

actions that would be performed in carrying out the complex action. Such a sequence seems to be a natural interpretation of the term *process* as it is used in computer science. If we regard procedures as action descriptions, then we conclude that the relation between procedures and processes is that in a given environment, a procedure denotes the process that would result from executing the procedure in that environment. Pratt (1979) has independently proposed a similar approach, but with a slightly different notion of what a process is. This type of approach seems to be definitely preferable to the notion that a procedure denotes the function it computes, since that idea ignores questions of efficiency and does not seem to handle programs such as operating systems which do not in any interesting sense compute a value.

3.2 The Dependence of Action on Knowledge

As we pointed out in section 1.1, knowledge and action interact in two principal ways. First, knowledge is often required prior to taking action, and second, actions can change what is known. In the first area, we need to consider knowledge preconditions as well as physical preconditions for actions. Our main thesis is that almost all knowledge preconditions for actions can be analyzed as a matter of knowing what action to take. Recall from chapter 1 our example of of trying to open a locked safe. Why is it that for an agent to achieve this goal by using the plan "Dial the combination of the safe," he must know the combination? The reason is that an agent could know that dialing the combination of the safe would result in the safe being open, but still not know *what to do*, because he does not know what the combination of the safe is. A similar analysis applies to knowing a telephone number in order to call someone on the telephone or knowing a password in order to gain access to a computer system.

It is important to realize that even mundane actions that are not usually thought of as

requiring any special knowledge are no different from the examples just cited. For instance none of the AI problem solving systems that dealt with the blocks world tried to take into account whether the robot had sufficient knowledge to be able to move block A to point B. Yet if a command were phrased as "Move my favorite block back to its original position," the system could be just as much in the dark as with "Dial the combination of the safe." If the system does not know what actions satisfy the description, it will not be able to carry out the command. The only reason that that the question of knowledge seems more salient in the case of dialing combinations and telephone numbers is that, in the contexts where these goals naturally arise, usually there is no presumption that the agent knows what action fits the description.

An important consequence of this view is that the specification of an action will not need to include anything about knowledge preconditions. These will always be supplied by our general theory of using actions to achieve goals. What we will need to specify are criteria for knowing what action is referred to by an action description. As we will see, though, this can often be done implicitly.

In terms of our possible-world semantics for knowing, the usual way of knowing what entity is referred to by a description B is by having some description C that is a rigid designator, and knowing that B = C. (Note that if B itself is a rigid designator, it can be used for C.) In particular, then, knowing what action is referred to by an action description means having a rigid designator for that action. But if this is all the knowledge that is required for carrying out the action, then a rigid designator for an action must be an *executable description* of the action in the same sense that a computer program is an executable description of a computation for the interpreter of the language in which the program is written. Note that by "executable description", we mean that the description can be executed provided the physical preconditions of the action are satisfied. This is true of

programs as well. If the preconditions of a program are not satisfied, it may not be possible to execute the program because of run-time errors.

Often the actions we want to talk about are mundane general procedures that we would be willing to assume that everyone knows how to perform. Dialing a telephone number or the combination of a safe are likely examples. In many of these cases, assuming an agent knows the general procedure, if he knows what objects the procedure is to be applied to then he knows everything that is relevant to the problem. In such cases the function which represents the general procedure will be a rigid function so that if the arguments of the function are rigid designators, the term consisting of the function applied to the arguments will be a rigid designator. Hence knowing what objects the arguments are amounts to knowing what action the term refers to. We will treat dialing the combination of a safe as this type of procedure. That is, we assume that anyone who knows what combination he is to dial and what safe he is to dial it on knows what action he is to perform.

There are other procedures which we might also wish to assume that anyone could perform, but which cannot be represented as rigid functions. Consider the blocks world action Puton(B,C). Even though we would not want to question anyone's ability to perform Puton in general, knowing what objects B and C are will not be sufficient to perform Puton(B,C) without knowing *where* they are. We could have a special axiom stating that knowing what action Puton(B,C) is requires knowing where B and C are, but this will be unneccessary if we simply assume that everyone knows the definition of Puton in terms of more primitive actions. If we define Puton(x_1,x_2) as something like:

(Movehand(Location(x_1)); Grasp; Movehand(Location(Top(x_2))); Ungrasp)

then we can treat Movehand, Grasp, and Ungrasp as rigid functions, and we can see that executing Puton requires knowing the location of the two objects because the locations are

mentioned in the definition. So, although Puton itself is not a rigid function, we can avoid having a special axiom saying what the knowledge preconditions of Puton are, by defining Puton as a sequence of actions which are represented by rigid functions. Of course, in a practical system we would probably want to "compile" this information, rather than going back to the definition each time we reason about Puton. The compiled information would be in the form of theorems that can be derived from the basic axioms of the system and the definition of Puton.

In the preceding discussion, we have been assuming that knowing what to do amounts to knowing what single specific action to perform; e.g., we have assumed that $Dial(C_1,Sf_1)$ names only one action. Obviously, there are many different sequences of movements that would constitute dialing a particular combination on a particular safe, so we really ought to regard $Dial(C_1,Sf_1)$ as naming a class of actions rather than a single action. It would be completely straightforward to modify our theory to take this into account, but none of the interesting problems we want to look at turn on this point. Therefore we will merely note the fact and let it pass. It should be realized, though, that this is different from the distinction between individual events and types of events that we made in the previous section. Even if in a strict sense there were only one way to dial the combination of a safe, so $Dial(C_1,Sf_1)$ referred to a definite action, performing this action on different occasions would still constitute different individual events.

The picture we have presented seems to be an adequate account of knowing how to perform the sort of action that one can be *told* how to perform, but many skills do not appear to fit well in this theory. For instance, knowing how to ride a bicycle, play the piano, or speak a language does not seem to consist entirely in being in possession of the right factual knowledge. For instance, someone could be an expert on piano playing, knowing all about such things as the notation of piano music and the theory of technique

(e.g., what fingerings to use for various chords), but not be able to play because he had never practiced. The difference between such a person and a concert pianist, though, would not be a matter of knowledge (in the sense we have been using) at all. It seems probable that any analysis of the difference would have to be at the level of physiology. Therefore, in our theory, whenever we consider an action that can only be performed by someone possessing a specific skill, we will not treat the skill as a matte. of knowledge, but rather as one of the "physical" preconditions for performing the action. For example, in the specification of the action Read we will require that the agent of the action satisfy the condition Reads (i.e, "is able to read").

To formalize the theory we have developed, we will introduce a new modal operator Can. Can(A,Act,P) will mean that A can achieve P by performing Act, in the sense that A knows how to achieve P by performing Act. This notion could be used in a planning system to achieve a goal P by finding some plan Act such that the system can deduce Can(A,Act,P), where A is the system's name for itself. We will not give a possible-world semantics for Can directly; instead, we will give a definition of Can in terms of Know and Res, which we can use in reasoning about Can to transform a problem into terms of possible worlds. For a simple action that cannot be decomposed into more primitive actions, the definition of Can(A,Act,P) will be that A knows what action Act describes, and he knows that his performing Act will result in P being true. The "will result" in the second condition must be interpreted in the strong sense that it is possible for A to perform Act (i.e., Res). This forces the planner to check that the preconditions of his plan are fulfilled.

This definition of Can is adequate for simple actions, but it is too stringent for complex plans. The reason is that it requires the agent to know ahead of time exactly what he is going to do. In a complex plan, however, he may take some action that results in his acquiring knowledge about what to do in later stages of the plan. All that is required when

he starts executing the plan is that he knows what to do first and he knows that at each subsequent step he will know what to do next. So, the definition of Can for sequences of actions is that A can achieve P by doing (Act_1; Act_2) just in case by doing Act_1 he can bring it about that by doing Act_2 he can achieve P. If Act_1 is a simple action, then the two rules we have given require that for Can(A,(Act_1; Act_2),P) to be true, A must know what action Act_1 describes, and he must know that after performing Act_1 he can achieve P by performing Act_2.

Finally, we will define Can for conditional and iterative plans. The rule for conditionals is that Can(A,If(P,Act_1,Act_2),Q) is true just in case A knows that P is true and A can achieve Q by doing Act_1, or A knows that P is false and A can achieve Q by doing Act_2. In other words, for an agent to know how to achieve a goal using a conditional plan, he must know whether the condition is true and know how to achieve the goal using the appropriate branch of the conditional. The rule for iterative plans is quite simple; it just specifies one level of expansion of the loop. The rule is that Can(A,While(P,Act_1),Q) is true just in case Can(A,If(P,(Act_1; While(P,Act_1)),Nil),Q) is true. Since Nil is a primitive action, it is covered by the first rule, but we will note that the result of applying this rule to Nil is that Can(A,Nil,P) is true just in case A knows that P is true. This may seem to be a trivial point, but it is important for a planner to realize that if its goal is already true then it doesn't have to do anything.

3.3 The Effects of Action on Knowledge

In reasoning about the effects of an action on the knowledge of the agent, our chief concern will be whether the action gives the agent any new information. Those actions that provide the agent with new information will be called *knowledge-producing* actions. We will

say that an action is knowledge-producing just in case after performing the action the agent would know more about the resulting situation than he did before performing the action. In the blocks world, looking inside a box could be a knowledge-producing action, while moving a block probably would not. Even if after moving the block the agent could see what configuration the blocks are in, the action would not be considered a knowledge-producing action, provided the agent could have predicted before hand what configuration would result. In the real world there are probably no actions which are never knowledge-producing, because all physical processes are subject to errors. Nevertheless, it seems clear that we do and should treat many actions as not being knowledge-producing to simplify the process of planning.

Even if an action is not knowledge-producing in the sense that we have just defined, performing the action will still alter the state of knowledge of the agent. The reason for this is that, assuming the agent is aware of his action, he will then know that the action has been performed. As a result, the tense and modality of many of the things he knows will change. For example, if before performing the action he knows that P is true, then after performing the action he will know that P was true before he performed the action. Similarly, if before performing the action he knew that P would be true after performing the action, then after performing the action he will know that P is true.

We can represent this very elegantly in terms of possible worlds. Suppose Act describes an action which is not knowledge-producing and A names an agent. Then let :A be the agent described by A, and let $:Ev_1$ be the event described by Do(A,Act), i.e, the event which consists in :A performing the action described by Act. Then for any possible worlds W_1 and W_2 such that W_2 is the result of $:Ev_1$ happening in W_1, the worlds which are compatible with what $:A_1$ knows in W_2 are exactly those worlds which are the result of $:Ev_1$ happening in some world which is compatible with what $:A_1$ knows in W_1. This tells us exactly how

what $:A_1$ knows after $:Ev_1$ happens (i.e. after :A performs the action described by Act) is related to what $:A_1$ knows before $:Ev_1$ happens.

We can try to get some insight into this analysis by studying figure 3.2. Sequences of possible situations connected by events can be thought of as possible courses of events. If W_1 is an actual situation and $:Ev_1$ happens producing W_2, then W_1 and W_2 form a subsequence of the actual course of events. Now we can ask what other courses of events are compatible with what :A knows in W_1 and in W_2. Suppose W_4 and W_3 are connected by $:Ev_1$ in a course of events that is compatible with what :A knows in W_1. Since $:Ev_1$ is not knowledge-producing for :A, the only sense in which his knowledge is increased by $:Ev_1$ is that he knows that $:Ev_1$ has happened. Since $:Ev_1$ happens at the corresponding place in the course of events that includes W_4 and W_3, this course of events will still be compatible with every thing :A knows in W_2. However, the appropriate "tense shift" takes place. In W_1, W_4 is a possible alternative present for :A, and W_3 is a possible alternative future. In W_2, W_3 is a possible alternative present for :A, and W_4 is a possible alternative past.

Next consider a different course of events that includes W_5 and W_6 connected by a different event $:Ev_2$. This course of events might be compatible with what :A knows in W_1 if he is not certain what he will do next, but after $:Ev_1$ has happened and he knows that it has happened, this course of events is no longer compatible with what he knows. Thus, W_6 is not compatible with what :A knows in W_2. We can see then that even actions which provide the agent no new information from the outside world still filter out for him those courses of events where he might perform actions other than those which he actually performs.

The idea of a filter on possible courses of events also provides a good picture of

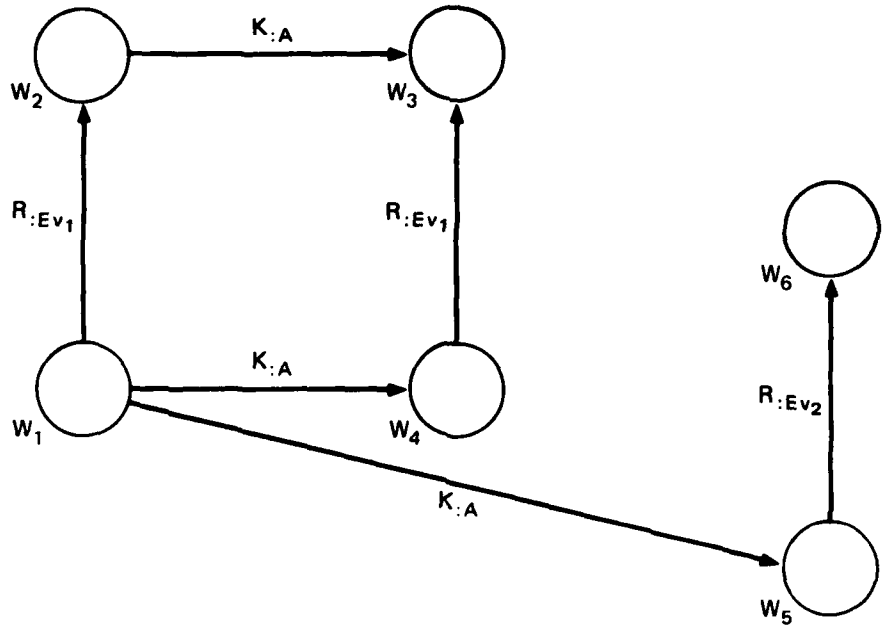


Figure 3.2 The effect of performing an action that is not knowledge-producing on the knowledge of the agent.

knowledge-producing actions. With these actions, though, the filter is even stronger, since they not only filter out courses of events that differ from the actual course of events as to what happens, but they also filter out courses of events which are incompatible with the information the action produces. Suppose Act describes a knowledge-producing action, where the knowledge that the agent gains is whether P is true. If :Ev is the event which consists in :A performing the action described by Act, then for any possible worlds W_1 and W_2 such that W_2 is the result of :Ev happening in W_1, the worlds which are compatible with what $:A_1$ knows in W_2 are exactly those worlds which are the result of :Ev happening in some world which is compatible with what $:A_1$ knows in W_1 *and in which* P *has the same truth value as in* W_2. It is this final condition that distinguishes actions that are knowledge-producing from those that are not.

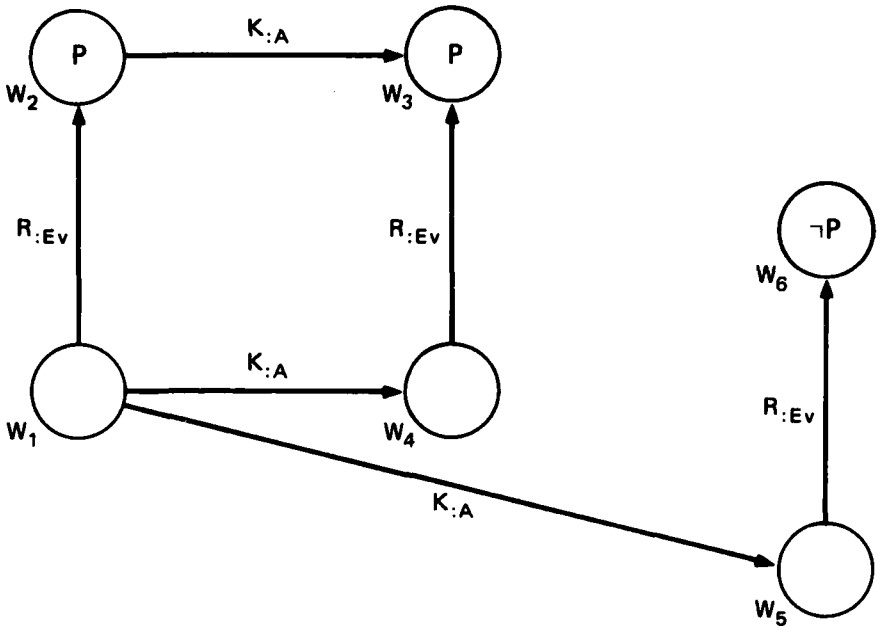


Figure 3.3 The effect of performing a knowledge-producing action on the knowledge of the agent.

Figure 3.3 illustrates this analysis. Suppose W_1 and W_2 are connected by :Ev and are part of the actual course of events. Suppose further that P is true in W_2. Let W_4 and W_3 also be connected by :Ev, and let them be part of a course of events that is compatible with what :A knows in W_1. If P is true in W_3, then if the only thing :A learns about the world from :Ev (other than that it has happened) is whether P is true, this course of events will still be compatible with what :A knows after :Ev happens. That is, W_3 will be compatible with what :A knows in W_2. Suppose, on the other hand, that W_5 and W_6 form part of a similar course of events, except that P is false in W_6. If :A does not know in W_1 whether P would be true after :Ev happened, then this course of events will be compatible with what he knows in W_1. After :Ev has happened, however, he will know that P is true, so this course of events will no longer be compatible with what he knows. That is, W_6 will not be compatible with what :A knows in W_2.

One major advantage of this approach to describing how an action affects what the agent knows is that, not only have we specified what he learns from the action, but also what he does not learn. Our analysis gives us not only sufficient conditions for inferring that :A knows that P after event :Ev, but also necessary conditions. In the case of an action which is not knowledge-producing, we can infer that unless :A knew before performing the action whether P would be true, he does not know afterwards either. In the case of a knowledge-producing action where what is learned is whether Q is true, he will not know whether P is true unless he already knows, or he knows how P depends on Q.

This possible-world analysis of the effects of action on knowledge gives us everything we need to formalize the notion of a test that we presented in section 1.1. Recall that a test was defined to be an action that has a directly observable result that depends conditionally on an unobservable precondition. In the terminology of this chapter we would say that a test is a knowledge-producing action where the observable result is part of the information provided by the action. In chapter 1 we identified three conditions on an action being usable as a test for P:

(1) The agent knows that Q will be true after he performs the action just in case P is true before he performs the action.

(2) After the agent performs the action, he knows that he has just performed the action.

(3) After the agent performs the action, he knows whether Q is true.

Conditions (2) and (3) will be satisfied if Act describes a knowledge-producing action, where the knowledge provided includes whether Q is true. So, any such action can be used as a test for P just in case (1) is also satisfied. Using the theory that we have presented, we can show that this is the case, as illustrated by figure 3.4.

Suppose the action described by Act satisfies (1) - (3), and let :Ev be the event which

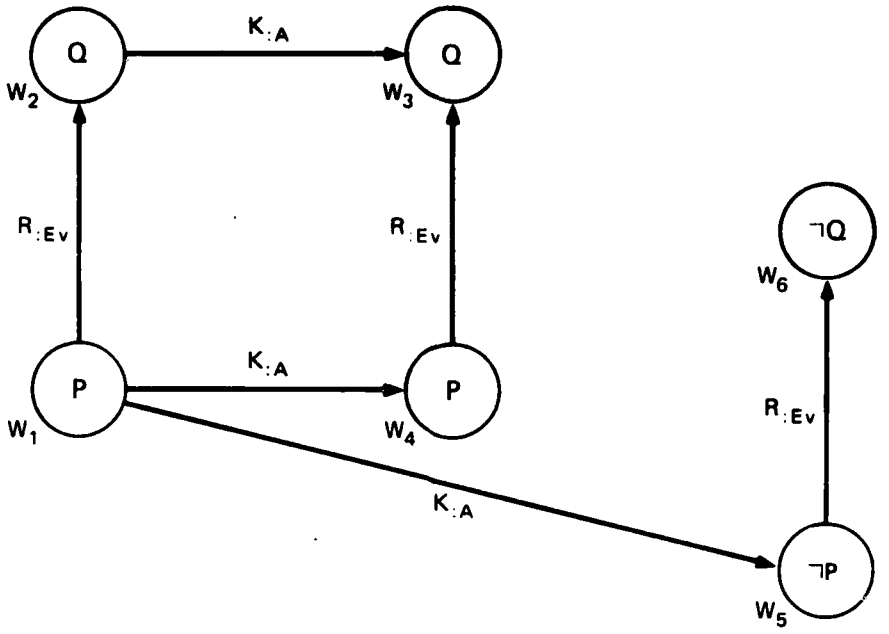


Figure 3.4 The effect of a test on the knowledge of the agent.

consists in :A performing the action. Suppose that P is true in W_1, but :A does not know whether P is true. Under these conditions, there will be at least one situation which is compatible with what he knows in W_1 in which P is true (W_4) and at least one such situation in which P is false (W_5). Since :A knows how the truth of Q after :Ev depends on the truth of P before :Ev, if W_3 is the result of :Ev happening in W_4, then Q must be true in W_3. Similarly, if W_6 is the result of :Ev happening in W_5, then Q must be false in W_6. Now, since P is in fact true in W_1, Q must be true in W_2. By condition (3), :A knows this fact, so W_3 will be compatible with what :A knows in W_2, but W_6 will not. This argument shows that after :Ev actually happens no possible course of events in which P is false before :Ev happens will be compatible with what :A knows. Thus we conclude that after :Ev happens, :A will know that P was true. It should be easy to see that if we assume that :A knows

whether ;Ev changes the truth value of P we could also show that ;A knows whether P is true after ;Ev occurs. An exactly parallel argument would apply if P were false in W_1, so we can see that our theory completely captures the reasoning about tests that we described in chapter 1, based on the general principles that govern reasoning about knowledge and action.

4. Formalizing the Possible-World Semantics for Knowledge

4.1 Object Language and Meta-Language

We have now presented all the basic theory that we need to construct a formalism for reasoning about knowledge and action. The essence of our approach is to define a logical language that contains modal operators for stating facts about knowledge and action, specify a possible-world semantics for the modal operators, and formalize that semantics in an ordinary first-order theory to which standard automatic deduction techniques can be applied. A major question that we have not answered, though, is what *formal* role, if any, the modal language will play in this formalism. In most, and perhaps all, cases it would be possible to frame the problem entirely within the concepts of the possible-world semantics, by-passing the modal operators completely. If we did this, the modal language would simply be a heuristic device for us to use in formulating problems in the possible-world formalism.

Rather than follow this approach, we will incorporate the modal language directly into our formalism. We will do this by encoding expressions of the modal language (which we will henceforth call the *object language*) as terms in a first-order language that talks about possible worlds (which we will call the *meta-language*). Then we can axiomatize the interpretation of modal expressions in terms of possible worlds using the relation T that we introduced in chapter 2, where T(W,P) is a meta-language formula which means that the object-language formula P is true in the world W. This is an idea adopted from McCarthy (1975).

There are several reasons for doing things in this way. First of all, even where the translation from modal notation to possible-world notation is quite direct, the modal notation is much more concise. Recall from section 3.1 the example of saying that A_1 knows that A_2 performing Act will result in P being true. In the modal notation this is expressed simply as Know(A_1,Res(Do(A_2,Act),P)). In the possible-world notation, however, it becomes:

$$\forall w_1 (K(:A_1, W_0, w_1) \supset \exists w_2 (R(:Do(:A_2, :Act), w_1, w_2) \wedge T(w_2, P))),$$

As a language for problem specification, then, the modal notation is clearly preferable to the possible-world notation.

There is a deeper problem than this, however, which seems not to have been previously noted. The possible-world framework is, in a sense, conceptually impoverished compared to the modal framework. Even if we can represent the same states of affairs within either framework, it does not follow that for every concept we can express in the modal language there will be a corresponding concept in the possible-world language. This seems to be the case with the modal operator Can. Notice that while there is a simple correspondence between the modal operator Know and the accessibility relation K, and between the modal operators Res and Resl and the accessibility relation R, we have no accessibility relation that corresponds directly to Can. Nevertheless, any formula of the form Can(A,Act,P) has a corresponding formula in the possible-world language. That formula can be obtained by expanding the definition of Can in terms of Know and Res (see section 5.2) and then transforming the occurrences of these operators into their possible-world counterparts. The problem is that the axioms that describe this transformation appear not to be formulatable completely within the possible-worlds framework.

It is important to remember that Can is not simply a relation among an agent, an action, and a goal; it is a relation among an agent, an action described in a particular way, and a goal described in a particular way. Furthermore, if the action is a described by a complex sequence of subactions, then the requirement that the agent know what subactions are being described is distributed over the execution of the whole sequence. The agent does not have to know exactly what action a particular step of the sequence describes until he actually has to do it. This seems to require that Can be defined recursively over action *descriptions*. In particular, the definition of Can has to allow for the fact that $Can(A,Act_1,P)$ can be true and

Can(A,Act_2,P) can be false even if Act_1 and Act_2 both describe the same action. We can accomodate this in the modal framework, because it allows such intensional constructions. The possible-world framework, however, is extensional, so there is a problem.

This sort of difficulty does not arise with Know because there is no need for a recursive definition to capture the meaning of Know. Res(Do(A,Act),P) and Resl(Do(A,Act),P), like Can may be defined recursively over Act (indeed, we will find it convenient to do so), but the recursion can pushed into the possible-world framework, because Act occupies an extensional position in these formulas. That is, Res(Do(A,Act_1),P) and Res(Do(A,Act_2),P) must have the same truth value if Act_1 and Act_2 describe the same action. Therefore we can give the definitions of Res and Resl as simple formulas in terms of the whatever event is denoted by its first argument. The recursion is then introduced in a natural way in determining what action is denoted by a complex action description. The problem with Can is that it not only requires a recursive definition, but the argument that definition recurs on is interpreted intensionally. There may be some way to make such a definition fit naturally into a pure possible-world framework, but I have not been able to find it.

The result of these considerations is that although any particular formula invoving Can will have a possible-world equivalent, there is no single concept in the possible-world framework which corresponds directly to Can in the way that the accessibility relations K and R correspond to the other modal operators. We can draw an analogy here with various levels of programming languages. We know that in theory any program can be translated into any universal basis for computation no matter how primitive, for example Turing machines or combinatory logic. The constructs that are used in one system may have no analogues in another system, however. Program variables are a good example of such a construct. Almost every practical programming language has some notion of variable in it, yet there are bases for computation, like combinatory logic (Curry and Feys, 1958), in which

there are no variables. So even though any program in a language with variables can be translated into combinatory logic, the variables will disappear in the process, the same way the concept Can disappears in translating from the modal notation to the possible-world notation.

Thus it is not merely clumsiness of syntax that leads us to prefer the modal language for purposes of problem specification, as one might be led to prefer PASCAL to FORTRAN. It is a fundamental difference in conceptual power, like the difference between both of these languages and assembly language.

Given that the modal language in some ways has greater conceptual power than the possible-world language because of its ability to express intensional concepts, we might ask how it is possible to axiomatize the interpretation of the modal language in an extensional first-order logic. Again the analogy with programming languages is helpful. Even though one programming language may be more powerful conceptually than another, it is always possible to write an interpreter for the first language in the second. Thus, we can write a LISP interpreter in assembly language, even though LISP has recursion and assembly language does not. We can do this because the interpreter treats LISP programs as data. We have the same sort of situation with respect to logical languages. Our extensional meta-language can interpret intensional object-language formulas, because those formulas are treated as logical individuals (i.e., data) in the meta-language.

4.2 A First-Order Treatment of the Propositional Logic of Knowledge

We will develop our formalism for reasoning about knowledge and action in a staged fashion. In this section we will deal with the propositional logic of knowledge, stating the necessary axioms and illustrating their use in some sample deductions. In the next section, we will introduce quantifiers and predicates, and in chapter 5 we will extend the formalism to handle our integrated theory of knowledge and action.

As we discussed in the previous section, the modal object language will be encoded as term expressions in a first-order meta-language. Typically this sort of thing is done using string operations like concatenation, so that the conjunction of P and Q would be represented by something like '('|P|'∧'|Q|')'. This would be interpreted as the string consisting of a left parenthesis followed by P followed by the conjunction symbol followed by Q followed by a right parenthesis. Thus the meta-language expression '('|P|'∧'|Q|')' would denote the object-language expression (P∧Q).

There is a much more elegant way to do the encoding, however, which is due to McCarthy (1962). For purposes of semantic interpretation of the object language, which is what we want to do, the details of the syntax of that language are largely irrelevant. In particular, the only thing we need to know about the syntax of conjunctions is that there is *some* way of taking P and Q and producing the conjunction of P and Q. We can represent this by having a function And such that And(P,Q) denotes the conjunction of P and Q. To use McCarthy's term, And(P,Q) is an *abstract syntax* for representing the conjunction of P and Q. We will represent all the logical operators of the object language by functions in an abstract syntax.

The object language will contain the usual logical operators and quantifiers with equality, and the modal operators Know, Res, Res1, and Can. The predicates, functions, and constants will vary from example to example, but will include the function Do discussed in chapter 3, and the composition functions for sequences, conditionals, and iterations of actions. The meta-language will include all the well-formed expressions of the object language as terms, the truth predicate T, the accessibility relations R and K, and analogues of all the nonintensional constructs of the object-language. The meta-language will also contain some additional constructs which will be introduced later. The domain of discourse of the meta-language includes the domain of discourse of the object language, plus object-language expressions and possible situation/worlds.

Since the axioms of our formalism will be introduced gradually over this chapter and the next with a large amount of intervening material, we list all of them in appendix A, indicating where in the text they are introduced. The axioms that specify the interpretation of object-language expressions in the meta-language constitute a recursive definition of the truth predicate T. Recall that T(W,P) means that the object language formula denoted by P is true in the possible world denoted by W. Often we will want to say that a formula is simply true, i.e., true in the actual world. We will therefore introduce a special constant symbol W_0 to denote the actual world and a monadic truth predicate True to mean true in the actual world. True is defined in terms of T and W_0 by the following axiom:

L1. $\forall p_1 (True(p_1) \equiv T(W_0,p_1))$

In the our formalism (i.e., the first-order meta-language) all predicates, functions, and constants will begin with upper-case letters, while variables will be in lower-case letters. We will be using a many-sorted logic, with different sorts assigned to differents sets of variables. For instance, the variables w_1, w_2,... will range over possible worlds, the variables p_1, p_2,... will range over object-language formulas, and the variables a_1, a_2,... will range over agents. For the sake of clarity, we once again note that object-language expressions are not formulas from the point of view of the logic. They are merely terms in the meta-language which we interpret as representing formulas of another language outside the formal system.

The recursive definition of T for the propositional part of the object language is as follows:

L2. $\forall w_1,p_1,p_2 (T(w_1,And(p_1,p_2)) \equiv (T(w_1,p_1) \wedge T(w_1,p_2)))$

L3. $\forall w_1,p_1,p_2 (T(w_1,Or(p_1,p_2)) \equiv (T(w_1,p_1) \vee T(w_1,p_2)))$

L4. $\forall w_1,p_1,p_2 (T(w_1,(p_1 \Rightarrow p_2)) \equiv (T(w_1,p_1) \supset T(w_1,p_2)))$

L5. $\forall w_1,p_1,p_2(T(w_1,(p_1 \langle\circ\rangle p_2)) \equiv (T(w_1,p_1) \circ T(w_1,p_2)))$

L6. $\forall w_1,p_1(T(w_1,Not(p_1)) \equiv \neg T(w_1,p_1))$

Axioms L1 - L6 just translate the logical connectives from the object language to the meta-language, using the ordinary Tarskian definition of truth. For instance, according to L2, And(P,Q) is true in a world if and only if P is true in the world and Q is true in the world. The other axioms state that all the truth-functional connectives are "transparent" to T in exactly the same way. As we pointed out in section 2.3, the major advantage of analyzing the object language in this way is that by lifting the logical connectives directly into the meta-language, we avoid having to formalize object-language axioms or rules of inference for them, and thereby avoid the problem of such axioms and rules taking control of the deductive process.

We should note that although we are axiomatizing truth for the object language, the well-known results of Tarski (see Rogers (1971), pp. 210 - 215) on the impossibility of axiomatizing truth do not apply here. What Tarski proved was that it is impossible for any language rich enough to contain arithmetic to consistently axiomatize its *own* truth conditions. But in our system the meta-language axiomatizes not its own truth conditions, but rather the truth conditions of the object-language. In particular, Tarski showed how to construct a term S which denotes the sentence "S is not true." This immediately gives rise to the classical liar paradox. This construction cannot be carried out in our system, because there is no representation of the predicates T or True in the object-language.

To get the propositional logic of knowledge, we need add only the following three axioms:

K1. $\forall w_1,trm.a_1,p_1(T(w_1,Know(trm.a_1,p_1)) \equiv \forall w_2(K(D(w_1,trm.a_1),w_1,w_2) \supset T(w_2,p_1)))$

K2. $\forall a_1,w_1(K(a_1,w_1,w_1))$

K3. $\forall a_1,w_1,w_2(K(a_1,w_1,w_2) \supset \forall w_3(K(a_1,w_2,w_3) \supset K(a_1,w_1,w_3)))$

K1 gives the possible-world analysis for object-language formulas of the form Know(A,P). The interpretation is that Know(A,P) is true in world W_1 just in case P is true in every world which is compatible with what the agent denoted by A in W_1 knows in W_1. Since an object language term may denote different individuals in different possible worlds, we introduce the function D, such that D(W,A) is a meta-language term that refers to the individual denoted by the object-language term A in world W. K represents the accessibility relation associated with Know, so $K(D(W_1,A),W_1,W_2)$ is how we represent W_2 being compatible with what the agent denoted by A in W_1 knows in W_1.

As we pointed out in section 2.3, the principle embodied in K1 is what we use to infer that an agent knows what is implied by his knowledge. Since this is not strictly true, in a more thorough analysis we would regard the inference from the right side of K1 to the left side as being a plausible implication. K2 and K3 state constraints on the accessibility relation K that we use to capture other properties of knowledge. Together, they require that for a fixed agent A, $K(A,w_1,w_2)$ must be a partial ordering on possible worlds. We have already shown in section 2.3 that this entails the principles that anything that anyone knows must be true, and that if someone knows something he knows that he knows it. Below we show how to derive these principles formally. Finally, the fact that K1 - K3 are asserted to hold for all possible worlds entails that everyone knows the principles they embody, and everyone knows that everyone knows, etc. In other words, these principles are common knowledge.

One of the features of our formalism illustrated by K1 is the method of writing meta-language variables for object-language terms. As we mentioned above, we are formalizing the meta-language as a many-sorted first-order logic. Since the domain of discourse of the

meta-language includes various types of well-formed object-language expressions, we will need meta-language variables to range over these expressions. We have already introduced the variables p_1, p_2,... which range over object-language formulas. Now we introduce the variables trm_1, trm_2,... to range over object-language terms. However, we also want to consider the object language to be a many-sorted logic, so we will need meta-language variables to range over object-language terms of a particular sort. Since the domain of discourse of the meta-language includes the domain of discourse of the object language, we will construct these variables so that if s_1, s_2,... are meta-language variables of sort s and this sort is also in the domain of discourse of the object language, then $trm.s_1$, $trm.s_2$,... will be meta-language variables that range over object-language terms of sort s. In K1, for instance, $trm.a_1$ ranges over object-language terms which refer to possible agents. Notice that this requires us to allow sorts to be hierarchically organized since the range of $trm.a_1$ is a subset of the range of trm_1.

To illustrate the use of these axioms, we will show how to derive some simple results in the propositional logic of knowledge. To simplify the formulas in these examples, we will assume that the object-language term A is a rigid designator for an individual who is also denoted by the meta-language term :A. This allows us to substitute :A for the more complicated D(W,A). Our proofs will be in natural deduction form. The axioms and preceding lines which justify each step will be given to the right of the step. Subordinate proofs will be indicated by indented sections, and Ass will mark the assumptions on which these subordinate proofs are based. Dis(n,m) will indicate the discharge of the assumption on line n with respect to the conclusion on line m. The general pattern of proofs in this system will be to assert the object-language premises of the problem, transform them into their meta-language equivalents using axioms L1-L6 and K1, then derive the meta-language

expression of the conclusion using first-order logic and purely meta-language axioms such as K2 and K3, and finally transform the conclusion back into the object language, again using L1-L6 and K1. Our first example will be to show that axiom M4 in the modal logic of knowledge follows from K1.

Prove: $True(Know(A,(P \Rightarrow Q)) \Rightarrow (Know(A,P) \Rightarrow Know(A,Q)))$

1.	$T(W_0,Know(A,(P \Rightarrow Q)))$	Ass
2.	$K(:A,W_0,w_1) \supset T(w_1,(P \Rightarrow Q))$	K1,1
3.	$T(W_0,Know(A,P))$	Ass
4.	$K(:A,W_0,w_1) \supset T(w_1,P)$	K1,3
5.	$K(:A,W_0,w_1)$	Ass
6.	$T(w_1,(P \Rightarrow Q))$	2,5
7.	$T(w_1,P) \supset T(w_1,Q)$	L4,6
8.	$T(w_1,P)$	4,5
9.	$T(w_1,Q)$	7,8
10.	$K(:A,W_0,w_1) \supset T(w_1,Q)$	Dis(5,10)
11.	$T(W_0,Know(A,Q))$	K1,10
12.	$T(W_0,Know(A,P)) \supset T(W_0,Know(A,Q))$	Dis(3,11)
13.	$T(W_0,(Know(A,P) \Rightarrow Know(A,Q)))$	L4,12
14.	$T(W_0,Know(A,(P \Rightarrow Q))) \supset T(W_0,(Know(A,P) \Rightarrow Know(A,Q)))$	Dis(1,13)
15.	$T(W_0,(Know(A,(P \Rightarrow Q)) \Rightarrow (Know(A,P) \Rightarrow Know(A,Q))))$	L4,14
16.	$True(Know(A,(P \Rightarrow Q)) \Rightarrow (Know(A,P) \Rightarrow Know(A,Q)))$	L1,15

This proof is completely straight-forward. Lines 1 - 4 assume the two antecedent conditions and then express them in possible-worlds notation. Then we pick w_1 as a typical world which is possible according to what A knows. In lines 6 - 9, we do the inference that we want to attribute to A. Since this inference can be done in an arbitrarily chosen member of the set of worlds which are possible for A, it must be valid in all of them (line 10). From this we conclude that A can probably do the inference also (line 11). We then discharge our assumptions and express the result in the same form as M4.

Another interesting example is the inference that proved to be so troublesome for the data-base approach - concluding $\neg Know(A,P)$ from $Know(A,(P \supset Q))$ and $\neg Know(A,Q)$:

Given: True(Know(A,(P => Q)))
True(Not(Know(A,Q)))

Prove: True(Not(Know(A,P)))

1. True(Know(A,(P => Q)))	Given
2. $T(W_0,Know(A,(P => Q)))$	L1,1
3. $K(:A,W_0,w_1) \supset T(w_1,(P => Q))$	K1,1
4. True(Not(Know(A,Q)))	Given
5. $T(W_0,Not(Know(A,Q)))$	L1,4
6. $\neg T(W_0,Know(A,Q))$	L6,5
7. $K(:A,W_0,W_1)$	K1,6
8. $\neg T(W_1,Q)$	K1,6
9. $T(W_1,(P => Q))$	3,7
10. $T(W_1,P) \supset T(W_1,Q)$	L4,9
11. $\neg T(W_1,P)$	10,8
12. $\neg T(W_0,Know(A,P))$	K1,7,11
13. $T(W_0,Not(Know(A,P))$	L6,12
14. True(Not(Know(A,P))	L1,13

In this proof, like the preceding one, most of the steps simply translate between the modal notation and the possible-world notation. Lines 1 - 3 express the first premise in possible-world notation. Lines 4 - 8 do the same for the second premise. The key step is concluding from the fact that A doesn't know Q (line 6), that there is a world, W_1, which is compatible with everything that A knows and in which Q false (lines 6 and 8). From this and the fact that in every world compatible with what A knows, if P is true then Q is true, it follows by modus tollens that P must be false in W_1 (line 11). Translating back into modal notation, we get that A doesn't know P.

For our final two examples in this section, we will show formally that K2 and K3 entail M2 and M3:

Prove: True(Know(A,P) => P)

1.	$T(W_0,Know(A,P))$	Ass
2.	$K(:A,W_0,w_1) \supset T(w_1,P)$	K1,1
3.	$K(:A,W_0,W_0)$	K2
4.	$T(W_0,P)$	2,3

5. $T(W_0,Know(A,P)) \supset T(W_0,P)$	Dis(1,4)
6. $T(W_0,(Know(A,P) \Rightarrow P))$	L4,5
7. $True(Know(A,P) \Rightarrow P)$	L1,6

We assume that A knows that P (line 1), so P must be true in every world which is compatible with what A knows (line 2). By K2, the actual world, W_0, must be compatible with what A knows (line 3), so P must be true in W_0 (line 4).

Prove: $True(Know(A,P) \Rightarrow Know(A,Know(A,P)))$

1. $T(W_0,Know(A,P))$	Ass
2. $K(:A,W_0,w_1) \supset T(w_1,P)$	K1,1
3. $K(:A,W_0,w_1) \supset (K(:A,w_1,w_2) \supset K(:A,W_0,w_2))$	K3
4. $K(:A,W_0,w_1)$	Ass
5. $K(:A,w_1,w_2) \supset K(:A,W_0,w_2)$	3,4
6. $K(:A,w_1,w_2)$	Ass
7. $K(:A,W_0,w_2)$	5,6
8. $T(w_2,P)$	2,7
9. $K(:A,w_1,w_2) \supset T(w_2,P)$	Dis(6,8)
10. $T(w_1,Know(A,P))$	K1,9
11. $K(:A,W_0,w_1) \supset T(w_1,Know(A,P))$	Dis(4,10)
12. $T(W_0,Know(A,Know(A,P)))$	K1,11
13. $T(W_0,Know(A,P)) \supset T(W_0,Know(A,Know(A,P)))$	Dis(1,12)
14. $T(W_0,(Know(A,P) \Rightarrow Know(A,Know(A,P))))$	L4,13
15. $True(Know(A,P) \Rightarrow Know(A,Know(A,P)))$	L1,14

Again we assume that A knows that P (line 1), and again we conclude that P is true in every world which is compatible with what A knows (line 2). We let w_1 be a typical world compatible with what A actually knows (line 4), and we let w_2 be a typical world compatible with what A knows in w_1 (line 6). By K3, w_2 must also be compatible with what A actually knows (line 7), so P must be true in w_2 (line 8). Therefore, A knows that P in w_1 (line 10), and A knows that A knows that P in the actual world (line 12).

Actually, this last proof contains something of a cheat. When we made the assumption that A was a rigid desigator, we did so mainly to simplify the formulas that we had to work

with. In the proofs before this one, nothing depends on that assumption. In those proofs, everything still goes through if $D(W_0,A)$ is used instead of ιA. Here that is not the case, and the proof depends on A being a rigid designator. The formalism that we are developing in this chapter is correct, though. The problem is with the propositional logic of knowledge that we presented back in section 2.1. Recall that the intent of M3 was to represent the fact that if someone knows something, he knows that he knows it. It seems very natural to assume that this entitles us to infer Know(A,Know(A,P)) from Know(A,P). This is not quite right, however, because there is no guarantee that the person who is described by A will know that he is the person described by A.

Suppose the richest man in the world knows that he has less than ten billion dollars. If we apply M3, we will infer that the richest man in the world knows that the richest man in the world knows that he has less than ten billion dollars. This might not be true, however, if he does not know that he is the richest man in the world. He might think that someone else is the richest man in the world and has more than ten billion dollars. What we really want to infer is that the richest man in the world knows that *he* knows that he has less than ten billion dollars. This is in fact the principle that is captured by K3. One way of making the M3 version valid is to restrict the term that denotes the knower to be a rigid designator. That way we can be sure that he recognizes it as a description of himself. That is what we have done in this proof.

In this section we have seen how to axiomatize the possible-world semantics of the propositional modal logic of knowledge, so that its inferences can be captured in first order logic. In the next section, we will extend this approach to handle the quantified logic of knowledge.

4.3 Introducing Quantifiers, Predicates, and Equality

We will represent object-language quantifiers in our formalism by two functions in the abstract syntax, Exist and All. These functions will take two arguments, a term denoting an object-language variable and a term denoting an object-language formula, presumably containing at least one free occurrence of the variable. The terms that denote object-language variables will be meta-language constants beginning with a "?". The scheme for indicating what sort the variable belongs to will be the same as in the meta-language, but since formally these symbols are constants we will use upper-case rather than lower-case letters. Thus, corresponding to the meta-language variable s_1, we will have the object-language variable $?S_1$. Exist($?S_1$,P) will denote the object-language formula which means there is an individual of sort S such that the open formula P is true of that individual. Similarly, All($?S_1$,P) will mean that P is true of every individual of sort S.

Axiomatizing the interpretation of quantified object-language formulas presents some minor technical problems. We would like to say something like Exist($?S_1$,P) is true in W just in case there is some individual such that the open formula P is true of that individual in W. We don't have a way of saying that an open formula is true of an individual in a world, however; we just have the predicate T which simply says that a formula is true in a world. One way of solving the problem would be to introduce a new predicate, or perhaps redefine T, to express the Tarskian notion of satisfiability rather than truth. An elegant way to do this is to borrow the computer science notion of a *closure* (Sussman and Steele, 1975), which can be defined as an ordered pair consisting of a formula containing free variables and a set of bindings for those variables. If we used this notion we would talk about closures being true in a world rather than formulas. In interpreting a closure, whenever we came across a free variable we would use the binding specified by the closure

to interpret the variable. A closure whose body was Exist($?S_1$,P) would be true in W if there is some individual of sort S such that the closure of P in which that individual is bound to $?S_1$ is true in W.

While this approach is semantically elegant, it is syntactically clumsy, as it requires a complicated syntax to describe closures, and even purely propositional formulas have to be represented as closures with empty sets of bindings. We will take the simpler approach of finding substitutions for the free variables of a formula such that the resulting formula has the same truth value in every world as the equivalent closure. Using this approach to interpret quantified formulas, for every individual that satisfies the open formula P we need to be able to find a term that can be substituted for the free variable to make the resulting closed formula true. In an extensional object language any term that denotes the individual would do. Since our object-language is intensional, however, we have to take into account whether the term that we substitute will be evaluated with respect to a different possible world where it might denote a different individual. Therefore, we must use a rigid designator for the individual in question to insure that all occurrences of the subtituted expression will denote the intended referent.

This approach is semantically unattractive since it requires us to assume that there is at least one rigid designator for every individual in the domain of discourse. Our theory does not require this, and we even pointed out in section 2.5 that not requiring all individuals to have names was a desirable feature of the theory in respect to interpreting certain statements about knowing what something is. Therefore, we will adopt the substitutional approach for its syntactic simplicity, but we will refrain from making use of it in any way that would be incompatible with the closure approach.

In order to formalize the substitutional approach to the interpretation of quantified object-language formulas, we need to be able to construct a rigid designator in the object

language for any arbitrary individual. Since our representation of the object language is in the form of an abstract syntax, we can simply stipulate that there is a function @ such that for any individual in the domain of discourse of the object language, if :X is a meta-language term refering to that individual, then @(:X) denotes an object-language rigid designator for that individual. (We can read @(:X) as "the standard name of :X".) We can now state our interpretation rules for object language quantifiers. In the following axiom schemas P may be any object-language formula, $?S_i$ may be any object-language variable, s_i is the corresponding meta-language variable, and the notation $P[Trm_1/Trm_2]$ indicates the expression which results from substituting Trm_1 for every free occurrence of Trm_2 in P:

L7. $\forall w_1 (T(w_1, Exist(?S_i, P)) \equiv \exists s_i (T(w_1, P[@(s_i)/?S_i])))$

L8. $\forall w_1 (T(w_1, All(?S_i, P)) \equiv \forall s_i (T(w_1, P[@(s_i)/?S_i])))$

L7 says that an existentially quantified formula is true in a world W if there is some individual of the sort indicated by the bound variable, such that the formula which results from substituting a rigid designator for that individual for the bound variable in the body of the formula is true in W. L8 says that a universally quantified formula is true in W if every individual of the sort indicated by the bound variable is such that the formula which results from substituting a rigid designator for that individual for the bound variable in the body of the formula is true in W.

Note that one of the instances of L7 asserts the equivalence of the analysis of "knowing who" in terms of quantifying-in with the analysis in terms of rigid designators:

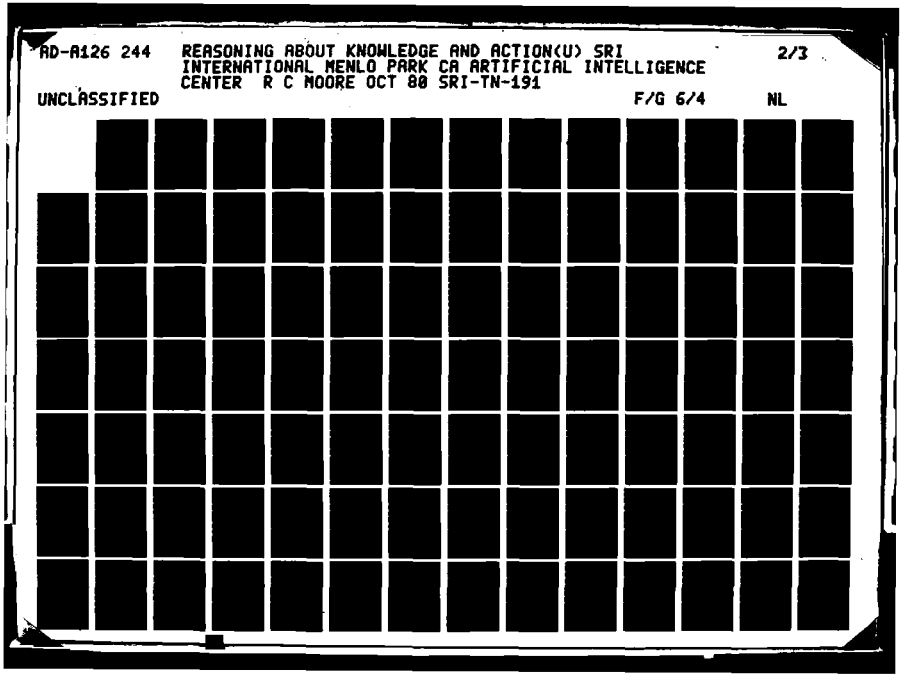
$T(w_1, Exist(?X_1, Know(A, Eq(?X_1, B)))) \equiv \exists x_1 (T(w_1, Know(A, Eq(@(x_1), B))))$

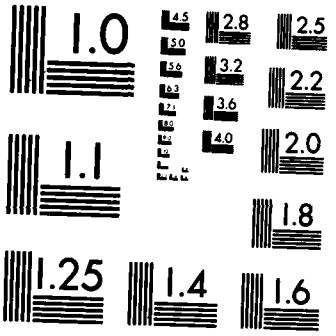
This says that there is something which A knows to be B (i.e. A knows who B is), just in case there is some individual x_1 such that A knows the proposition which asserts the equality of a rigid designator for x_1 and B.

Up to this point we have left nonintensional atomic formulas unanalyzed, reasoning in terms of meta-language expressions of the form T(W,P). In order to analyze object-language predicates and terms we will need a few new tools. We have already introduced one of these tools, the function D which maps a possible world and an object-language term into the denotation of that term in that world. Using this function, the simplest method of interpreting object-language atomic formulas would be to introduce, for each n-ary object language predicate, an n+1-ary meta-language predicate that took as its arguments the possible world in which the object-language formula is to be evaluated and the denotations in that world of the arguments of the object-language predicate. If we did this, then T(W,P(A)) would be analyzed as something like :P(W,D(W,A)). This treatment of predicates is similar to that used by McCarthy (1963) (McCarthy and Hayes, 1969) in his situation calculus.

This approach, however, creates problems when we try to formalize the effects of actions. If we do things this way, when we axiomatize a particular action we will have to say explicitly for each predicate, function, and constant how the action changes its extension. That is, for each predicate we will have to say how what the predicate is true of after the action is performed depends on what is true before the action is performed. Similarly, for each function or constant we must say how the action might change the referent of any terms that mention that function or constant. This problem was first pointed out by McCarthy and Hayes (1969) and was called by them the *frame problem*. We will call axioms that describe the effects of actions *frame axioms*.

It has often been noted that most actions affect relatively few aspects of a situation, so that the most concise formulation of the frame axioms for an action would be to explicitly state what things do change and then add that "everything else stays the same." If we follow the approach of translating object-language predicates into meta-language predicates,


AD-A126 244
REASONING ABOUT KNOWLEDGE AND ACTION(U) SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER R C MOORE OCT 80 SRI-TN-191
2/3
UNCLASSIFIED
F/G 6/4
NL



however, we cannot express the notion of everything else staying the same, because our meta-language is first-order, and we would have to quantify over those predicates.

So that we can state frame axioms more easily, we will adopt a notation where the meta-language analogue of an object-language predicate is a function rather than a predicate. We will analyze T(W,P(A)) as H(W,:P(D(W,A))). This formula can be read as ":P holds in W for the denotation of A in W." The difference between P and :P is that the argument of P is an object-language description of an individual which has to be interpreted relative to a possible world; the argument of :P is a meta-language description of the individual which is independent of possible worlds. (This is similar to the difference between Eval and Apply in LISP. When a function is Apply'ed its arguments have already been evaluated with respect to the relevant environment.) We can regard :P as function which maps an individual into an intensional object which may or may not hold in a given possible world. The interpretation is that :P(:A) holds in W, just in case :A has the property P in world W. H is the meta-language predicate which expresses this relationship. The difference between H and T is that T(W,P(A)) means that the object-language formula P(A) is true in the world W, while H(W,:P(:A)) means that the individual :A has the property :P in W. That is, the latter expression gives the semantic interpretation of the former.

If we use this notation the nonintensional atomic formulas of the object language are analyzed in the meta-language as term expressions such as :P(:A)). Since these are terms, we can quantify over them in our first-order meta-language and express the "everything else remains the same" clause in our frame axioms. This will be explained in more detail in the next chapter. The basic idea here was first suggested by Kowalski (1974) as a modification of McCarthy's situation calculus. The integration into the possible-worlds framework is original.

Now we can state the interpretation axioms for object-language predicates. These

Up to this point we have left nonintensional atomic formulas unanalyzed, reasoning in terms of meta-language expressions of the form T(W,P). In order to analyze object-language predicates and terms we will need a few new tools. We have already introduced one of these tools, the function D which maps a possible world and an object-language term into the denotation of that term in that world. Using this function, the simplest method of *interpreting* object-language atomic formulas would be to introduce, for each n-ary object language predicate, an n+1-ary meta-language predicate that took as its arguments the possible world in which the object-language formula is to be evaluated and the denotations in that world of the arguments of the object-language predicate. If we did this, then T(W,P(A)) would be analyzed as something like :P(W,D(W,A)). This treatment of predicates is similar to that used by McCarthy (1963) (McCarthy and Hayes, 1969) in his situation calculus.

This approach, however, creates problems when we try to formalize the effects of actions. If we do things this way, when we axiomatize a particular action we will have to say explicitly for each predicate, function, and constant how the action changes its extension. That is, for each predicate we will have to say how what the predicate is true of after the action is performed depends on what is true before the action is performed. Similarly, for each function or constant we must say how the action might change the *referent* of any terms that *mention* that function or constant. This problem was first pointed out by McCarthy and Hayes (1969) and was called by them the *frame problem*. We will call axioms that describe the effects of actions *frame axioms*.

It has often been noted that most actions affect relatively few aspects of a situation, so that the most concise formulation of the frame axioms for an action would be to explicitly state what things do change and then add that "everything else stays the same." If we follow the approach of translating object-language predicates into meta-language predicates,

axioms and the ones that follow formalize the convention that for most predicates, functions, or constants in the object language, the corresponding construct in the meta-language uses the same symbol preceded by a colon. L9a and L9b are axiom schemata, where P can be any nonintensional atomic predicate in the object language.

L9a. $\forall w_1,trm_1,...,trm_n(T(w_1,P(trm_1,...,trm_n)) \equiv H(w_1,:P(D(w_1,trm_1),...,D(w_1,trm_n))))$
if P is not an essential property of the things it is true of.

L9b. $\forall w_1,trm_1,...,trm_n(T(w_1,P(trm_1,...,trm_n)) \equiv :P(D(w_1,trm_1),...,D(w_1,trm_n)))$
if P is an essential property of the things it is true of.

L9a and L9b both say that an atomic formula is true if the corresponding relationship holds among the referents of the terms in the formula. The distinction between an essential and a non-essential property is that if an individual (or tuple of individuals) has an essential property, it has that property in all possible worlds. A good example is the fact that numbers are essentially numbers. That is, if some individual is a number, it is a number regardless of what possible world it is in. It would make no sense to talk about a possible world where the number five were, say, a chair. So the translation of object-language expressions for essential properties into the meta-language requires no reference to possible worlds. In this special case, the meta-language construct corresponding to an object-language predicate is in fact a predicate. This is the reason for the difference between L9a and L9b.

Whether there are in fact such things as essential properties is a very controversial issue in philosophy. The idea of essential properties originated with Aristotle, but by this century the idea had fallen into general disrepute. For instance, Quine (1953) bases his attack on quantified modal logic on the argument that such logics contain sentences whose interpretation presupposes the existence of essential properties, which Quine takes to be an incomprehensible notion. With the advances in formal semantics for modal logic, however,

essentialism has regained a certain degree of respectability, with Kripke (1972) arguing rather persuasively in its favor.

Whether or not we take essentialism seriously as a philosophical doctrine, it will be convenient to identify those properties in any problem domain which are unchangeable. We will sometimes treat certain predicates as being essential properties even if they are not, when we are interested only in a subset of possible worlds where they do not change. In the blocks world, we will consider being a block to be an essential property of all blocks, so long as we don't consider any actions which change blocks into non-blocks or vice-versa, and we are willing to assume that the robot knows of all blocks that they are blocks.

This last point is particularly important. If one of the knowers we are considering does not know that some object exists, then that object cannot have any essential properties. This is because the object would have to have those properties in all the worlds compatible with what that knower knows. The knower then would know that the object had these properties, and would, therefore, know that the object exists. An alternative formulation of the notion of essential properties would be to say that P is an essential property of A if A has P in every world where A exists. This would require more complicated axioms, however, and would not help in any of the examples we will consider, so we will stay with the simpler formulation.

The next set of axioms specifies the translation of object-language terms into the meta-language. L11a and L11b are axiom schemata where Cnst may be any object-language constant. L12a and L12b are axiom schemata where F may be any object language function.

L10. $\forall w_1,x_1 (D(w_1,@(x_1)) = x_1)$

L11a. $\forall w_1 (D(w_1,Cnst) = V(w_1,:Cnst))$ if Cnst is not a rigid designator.

L11b. $\forall w_1 (D(w_1,Cnst) = :Cnst)$ if Cnst is a rigid designator.

L12a. $\forall w_1,trm_1,...,trm_n(D(w_1,F(trm_1,...,trm_n)) = V(w_1,{:}F(D(w_1,trm_1),...,D(w_1,trm_n)))$
if F is not a rigid function.

L12b. $\forall w_1,trm_1,...,trm_n(D(w_1,F(trm_1,...,trm_n)) = {:}F(D(w_1,trm_1),...,D(w_1,trm_n)))$
if F is not a rigid function.

Since $@(x_1)$ is a rigid designator for x_1, its value is x_1 in every possible world. An object-language constant which is not a rigid designator translates into an intensional object (like the intensional objects corresponding to predicates), which determines an individual in each possible world. The function V maps a possible world and one of these intensional objects into the corresponding individual. The reason for interpreting object-language constants this way, as in the case of object-language predicates, is to be able to state frame axioms more easily. The referent of a rigid designator does not depend on which possible world it is evaluated in, so its translation into the meta-language is simply a constant. Similarly, non-rigid object-language functions translate into meta-language functions which map tuples of individuals into intensional objects. Rigid object-language functions translate into meta-language functions from tuples of individuals to individuals.

The final logical axiom in our system deals with equality:

L13. $\forall w_1,trm_1,trm_2(T(w_1,Eq(trm_1,trm_2)) \equiv (D(w_1,trm_1) = D(w_1,trm_2)))$

L13 is a special case of L9b, where the meta-language interpretation of the object-language predicate Eq is the known meta-language predicate =, rather than an undefined predicate :Eq. Two object-language terms are equal in a possible world if they name the same individual in that possible world. Note that since = does not depend on what possible world it is applied in, we are assuming that being identical to oneself is an essential property of every individual.

It may be instructive to see how translation into the meta-language distinguishes quantifiers which have different scopes with respect to the operator Know. In our current

formalism, the examples from section 2.5 of differing quantifier scopes would be expressed as follows:

(1) True(Know(John,Exist($?X_1$,And(Transistor($?X_1$),Burned-out($?X_1$)))))

(2) True(Exist($?X_1$,And(Transistor($?X_1$),Know(John,Burned-out($?X_1$)))))

Recall that (1) says that John knows there is a burned out transistor, while (2) says that there is a transistor which John knows is burned out.

Applying the axioms we have just given will produce the following meta-language translations for these two formulas:

(3) $\forall w_1 (K(:John,W_0,w_1) \supset \exists x_1 (H(w_1,:Transistor(x_1)) \wedge H(w_1,:Burned\text{-}out(x_1))))$

(4) $\exists x_1 (H(W_0,:Transistor(x_1)) \wedge \forall w_1 (K(:John,W_0,w_1) \supset H(w_1,:Burned\text{-}out(x_1))))$

(3) says that in every world which is compatible with what John knows in the real world, there is some transistor which is burned out. (4) says that there is some particular transistor which is burned out in every world which is compatible with what John knows in the real world.

With these axioms we can also show formally how the fact that A knows that P(C) can be derived from the fact that A knows that P(B) and A knows that B = C.

Given: True(Know(A,P(B)))
True(Know(A,Eq(B,C)))

Prove: True(Know(A,P(C)))

1. True(Know(A,P(B)))	Given
2. $T(W_0,Know(A,P(B)))$	L1,1
3. $K(D(W_0,A),W_0,w_1) \supset T(w_1,P(B))$	K1,2
4. $K(V(W_0,:A),W_0,w_1) \supset T(w_1,P(B))$	L11a,3
5. True(Know(A,Eq(B,C)))	Given
6. $T(W_0,Know(A,Eq(B,C)))$	L1,5
7. $K(D(W_0,A),W_0,w_1) \supset T(w_1,Eq(B,C))$	K1,6

8. $K(V(W_0,:A),W_0,w_1) \supset T(w_1,Eq(B,C))$	L11a,7
9. $K(V(W_0,:A),W_0,w_1)$	Ass
10. $T(w_1,P(B))$	4,9
11. $H(w_1,:P(D(w_1,B)))$	L9a,10
12. $H(w_1,:P(V(w_1,:B)))$	L11a,11
13. $T(w_1,Eq(B,C))$	8,9
14. $D(w_1,B) = D(w_1,C)$	L13,13
15. $V(w_1,:B) = V(w_1,:C)$	L11a,14
16. $H(w_1,:P(V(w_1,:C)))$	12,15
17. $H(w_1,:P(D(w_1,C)))$	L11a,16
18. $T(w_1,P(C))$	L9a,17
19. $K(V(W_0,:A),W_0,w_1) \supset T(w_1,P(C))$	Dis(9,18)
20. $K(D(W_0,A),W_0,w_1) \supset T(w_1,P(C))$	L11a,19
21. $T(W_0,Know(A,P(C)))$	K1,20
22. $True(Know(A,P(C)))$	L1,21

A knows that P(B) (line 1), so P(B) is true in every world compatible with what A knows (line 4). Similarly, since A knows that B = C (line 5), B = C is true in every world compatible with what A knows (line 8). Let w_1 be one of these worlds (line 9). P(B) and B = C must be true in w_1 (lines 12 and 15), hence P(C) must be true in w_1 (line 16). Therefore, P(C) is true in every world compatible with what A knows (line 19), so A knows that P(C) (line 22). If True(Eq(B,C)) were given instead of True(Know(A,Eq(B,C))), we would have had B = C true in W_0 instead of w_1. In that case, the substitution of C for B in P(B) (line 16) would not have been valid, and we could not have concluded that A knows that P(C). This proof seems long because we have made each routine step a separate line. This is worth doing once to illustrate all the formal details, but in subsequent proofs, we will combine some of the routine steps to shorten the length of the derivation.

Another good example of reasoning about equality and quantification in knowledge contexts is to show formally that if A knows who B is and A knows who C is, then A must know whether B = C. Recall that the modal formula that represents A knowing who B is, is $\exists x(Know(A,(x = B)))$.

Given: True(Exist(?X_1,Know(A,Eq(?X_1,B))))
True(Exist(?X_1,Know(A,Eq(?X_1,C))))

Prove: True(And((Eq(B,C) => Know(A,Eq(B,C))),(Not(Eq(B,C)) => Know(A,Not(Eq(B,C))))))

1. True(Exist(?X_1,Know(A,Eq(?X_1,B))))	Given
2. T(W_0,Exist(?X_1,Know(A,Eq(?X_1,B))))	L1,1
3. $\exists x_1$ (T(W_0,Know(A,Eq(@(x_1),B))))	L7,2
4. T(W_0,Know(A,Eq(@(:B'),B)))	3
5. K(V(W_0,:A),W_0,w_1) ⊃ T(w_1,Eq(@(:B'),B))	K1,L11a,4
6. True(Exist(?X_1,Know(A,Eq(?X_1,C))))	Given
7. K(V(W_0,:A),W_0,w_1) ⊃ T(w_1,Eq(@(:C'),C))	L1,L7,K1,L11a,6

Lines 1 - 7 translate the premises from the object language into the meta-language, letting :B' be the individual whom A knows to be the thing referred to by B, and letting :C' be the individual whom A knows to be the thing referred to by C. Some of the intermediate steps have been suppressed.

8. T(W_0,Eq(@(:B'),B))	K2,5
9. D(W_0,@(:B')) = D(W_0,B)	L13,8
10. :B' = D(W_0,B)	L10,9
11. :B' = V(W_0,:B)	L11a,10
12. T(W_0,Eq(@(:C'),C))	K2,7
13. :C' = V(W_0,:C)	L13,L10,L11a,12

According to K2, the actual world must be compatible with what A knows, so B must denote :B' in W_0 (line 11). A similar argument applies to C (lines 12 and 13).

The rest of the proof is divided into two cases; we show that if B = C, then A knows that B = C, and if B ≠ C, then A knows that B ≠ C.

14.	T(W_0,Eq(B,C))	Ass
15.	V(W_0,:B) = V(W_0,:C)	L13,L11a,14
16.	:B' = :C'	11,13,15

First we assume that B = C is true in the actual world (line 14). According to lines 11 and 13, this means that :B' and :C' must be the same individual (line 16).

17.	$K(V(W_0,:A),W_0,w_1)$	Ass
18.	$T(w_1,Eq(@(:B'),B))$	5,17
19.	$:B' = V(w_1,:B)$	L13,L10,L11a,18
20.	$T(w_1,Eq(@(:C'),C))$	7,17
21.	$:C' = V(w_1,:C)$	L13,L10,L11a,20
22.	$V(w_1,:B) = V(w_1,:C)$	16,19,21
23.	$K(V(W_0,:A),W_0,w_1) \supset (V(w_1,:B) = V(w_1,:C))$	Dis(17,22)
24.	$T(W_0,Know(A,Eq(B,C)))$	L11a,L13,K1,23
25.	$T(W_0,Eq(B,C)) \supset T(W_0,Know(A,Eq(B,C)))$	Dis(14,24)
26.	$T(W_0,(Eq(B,C) \Rightarrow Know(A,Eq(B,C))))$	L4,25

We let w_1 be a typical world which is compatible with what A knows (line 17). Therefore, it must be true in w_1 that B denotes :B' (line 19) and C denotes :C' (line 21). Since, :B' and :C' are the same individual, B and C have the same denotation in w_1 (line 22), so A must know that B = C (line 24). Discharging the assumption, if B = C, then A knows that B = C (line 26).

27.	$T(W_0,Not(Eq(B,C)))$	Ass
28.	$V(W_0,:B) \neq V(W_0,:C)$	L13,L11a,14
29.	$:B' \neq :C'$	11,13,28
30.	$K(V(W_0,:A),W_0,w_1)$	Ass
31.	$T(w_1,Eq(@(:B'),B))$	5,30
32.	$:B' = V(w_1,:B)$	,L13,L10,L11a,31
33.	$T(w_1,Eq(@(:C'),C))$	7,30
34.	$:C' = V(w_1,:C)$	L13,L10,L11a,33
35.	$V(w_1,:B) \neq V(w_1,:C)$	29,32,34
36.	$K(V(W_0,:A),W_0,w_1) \supset (V(w_1,:B) \quad V(w_1,:C))$	Dis(30,35)
37.	$T(W_0,Know(A,Not(Eq(B,C))))$	L11a,L13,L6,K1,36
38.	$T(W_0,Not(Eq(B,C))) \supset T(W_0,Know(A,Not(Eq(B,C))))$	Dis(27,37)
39.	$T(W_0,(Not(Eq(B,C)) \Rightarrow Know(A,Not(Eq(B,C)))))$	L4,38
40.	$T(W_0,And((Eq(B,C) \Rightarrow Know(A,Eq(B,C))),$ $(Not(Eq(B,C)) \Rightarrow Know(A,Not(Eq(B,C))))))$	L2,26,39
41.	$True(And((Eq(B,C) \Rightarrow Know(A,Eq(B,C))),$ $(Not(Eq(B,C)) \Rightarrow Know(A,Not(Eq(B,C))))))$	L1,40

In the second case, we assume that B ≠ C (line 27). This means that :B' and :C' are not the same individual (line 29). By an argument completely parallel to the first case, we conclude

that if B $\not=$ C, then A knows that B $\not=$ C (line 39). Combining the two cases gives the desired final result (line 41).

5. A First-Order Theory of Knowledge and Action

5.1 Formalizing the Possible-World Semantics for Actions

In the preceding chapter we showed how to formalize in first-order logic the possible-world semantics for knowledge. In this chapter we will extend that formalism to encompass our integrated theory of knowledge and action. We will begin by presenting a first-or treatment of the possible-world semantics for actions. In the rest of the chapter we bring in the ideas about the interaction of knowledge and action presented in chapter 3.

In chapter 3 we introduced the object-language modal operator Res which takes arguments a description of an event and a formula. The interpretation of Res(Ev,P) being true in W was that it is possible for the event described by Ev to occur in W and if it did, P would be true in the resulting situation. By assuming that all events are deterministic, we could express this in terms of possible worlds by saying that there is some world which is the result of the event described by Ev happening in W and in which P is true. In our first-order formalism this is represented as follows:

R1. $\forall w_1, trm.ev_1, p_1$
$(T(w_1, Res(trm.ev_1, p_1)) \equiv \exists w_2 (R(D(w_1, trm.ev_1), w_1, w_2) \wedge T(w_2, p_1)))$

The only new notation introduced in this axiom is the variable $trm.ev_1$ which ranges over object-language terms that denote events.

In chapter 3 we also noted that the events that we are interested in consist of agents performing actions. We introduced an object-language function Do such that Do(A,Act) names the event in which the agent described by A performs the action described by Act. We decided to let Do be a rigid function, so that Do(A,Act) will be a rigid designator of an event if A is a rigid designator of an agent and Act is a rigid designator of an action. Hence, by axiom L12b, $D(W,Do(A,Act)) \equiv :Do(D(W,A),D(W,Act))$.

We also introduced several operators to construct complex actions out of simpler ones, ; for sequences, If for conditionals, and While for iterations. In chapter 3 we informally described how a possible-world semantics could be given for these complex actions directly. Here, however, we will take a slightly different approach. The problem is that to apply an axiom like R1 to a formula containing a complex action description, we would have to axiomatize what these action descriptions denote. That is, we would have to define how the function D behaves with respect to these operators. We cannot simply apply the L12 axioms because these complex action descriptions must be interpreted intensionally. In particular, if we have an action described as a sequence, any expression mentioned in a step of the sequence must be interpreted relative to the situation in which that step of the sequence is executed. For example, if we execute the sequence "(chop down the tallest tree; chop down the tallest tree)", the trees refered to by the two instances of "the tallest tree" will be different. The same sort of thing occurs in the interpretation of programming language expressions, e.g., (X <- X+1; X <- X+1). Here the interpretations of the two occurrences of X on the right side of the assignment statements will be different.

So, the interpretation of complex action descriptions will not be trivial. In general, the natural thing to take as the denotation of a complex action description in a situation W would seem to be the particular sequence of simple actions that would result from executing the complex description in W. Determining what that sequence is could require a complex series of deductions and, if loops are involved, it may be undecidable. All we really want to do for this thesis, however, is to be able to do some inferences about formulas in which a complex action description appears as an argument to Res, Res1, or Can. We have already argued that Can has to be defined recursively in the object-language. If we look back at section 3.2, we see that defining Can this way did not require talking about the denotation of complex action descriptions. This suggests that we can avoid the problem altogether by

defining Res and Resl for complex actions in a similar way. Of course, this does not make the theoretical problem go away. What it does do is allow us to confine our attention to the specific problem we want to address, deducing formulas containing Res and Resl, without having to deal with the general problem of what sequence of simple actions is denoted by a complex action description. We should note, though, that the conceptual framework that we have developed seems to be adequate for attacking this harder problem. It is the procedural difficulties of actually getting a system to do the deductions that we want to put off for further research.

Taking these considerations into account, we will work with the following recursive definition of Res for sequences, conditionals, and iterations:

R2. $\forall w_1,trm.a_1,trm.act_1,trm.act_2,p_1$
$(T(w_1,Res(Do(trm.a_1,(trm.act_1; trm.act_2)),p_1)) \equiv$
$T(w_1,Res(Do(trm.a_1,trm.act_1),Res(Do(@(D(w_1,trm.a_1)),trm.act_2),p_1))))$

R3. $\forall w_1,trm.a_1,trm.act_1,trm.act_2,p_1,p_2$
$(T(w_1,Res(Do(trm.a_1,If(p_1,trm.act_1,trm.act_2)),p_2)) \equiv$
$((T(w_1,p_1) \wedge T(w_1,Res(Do(trm.a_1,trm.act_1),p_2))) \vee$
$(\neg T(w_1,p_1) \wedge T(w_1,Res(Do(trm.a_1,trm.act_2),p_2)))))$

R4. $\forall w_1,trm.a_1,trm.act_1,p_1,p_2$
$(T(w_1,Res(Do(trm.a_1,While(p_1,trm.act_1)),p_2)) \equiv$
$T(w_1,Res(Do(trm.a_1,If(p_1,(trm.act_1; While(p_1,trm.act_1)),Nil)),p_2)))$

R5. $\forall trm.a_1,w_1,w_2(R(Do(trm.a_1,Nil),w_1,w_2) \equiv (w_1 = w_2))$

R2 defines Res for a sequence of actions. A proposition p_1 is true in the situation resulting from the agent $trm.a_1$ carrying out the sequence of actions $(trm.act_1; trm.act_2)$, just in case p_1 is true in the situation resulting from the agent $trm.a_1$ carrying out the action $trm.act_2$ in the situation resulting from the agent $trm.a_1$ carrying out the the action $trm.act_1$. The agent of the second action is more precisely specified by $@(D(w_1,trm.a_1))$. This

expression denotes a rigid designator for the referent of $trm.a_1$ in w_1. This makes sure that the agent of the second action is the same as the agent of the first action, in cases the referent of $trm.a_1$ is changed by the first action.

R3 defines Res for a conditional action. A proposition p_2 is true in the situation resulting from the agent $trm.a_1$ carrying out the conditional action $If(p_1,trm.act_1,trm.act_2)$, just in case p_1 is true and, in the situation resulting from the agent $trm.a_1$ carrying out the action $trm.act_1$, p_2 is true, or p_1 is false and, in the situation resulting from the agent $trm.a_1$ carrying out the action $trm.act_2$, p_2 is true.

R4 defines Res for an iterated action. A proposition p_2 is true in the situation resulting from the agent $trm.a_1$ carrying out the iterated action $While(p_1,trm.act_1)$, just in case p_2 is true in the situation resulting from the agent $trm.a_1$ carrying out the action $If(p_1,(trm.act_1; While(p_1,trm.act_1)),Nil)$. That is, to carry out $While(p_1,trm.act_1)$, an agent would repeat the action $trm.act_1$ as long as p_1 remained true.

R5 defines the execution of the null action as the event which maps every situation into itself. That is, the null action changes nothing. We introduce the action Nil merely to fill out the unused branch of conditionals, as in R4.

We can give a set of axioms for Resl that parallel those for Res. Recall that Resl is a weaker operator than Res in that to deduce Res(Ev,P) we must show that it is possible for Ev to occur, while to deduce Resl(Ev,P) we need only show that if Ev does occur P will be true in the resulting situation. We can express this as follows:

R6. $\forall w_1, trm.ev_1, p_1$
$(T(w_1, Resl(trm.ev_1, p_1)) \equiv \forall w_2(R(D(w_1, trm.ev_1), w_1, w_2) \supset T(w_2, p_1)))$

Resl(Ev,P) is true in W_1 if, assuming W_2 is the result of Ev happening in W_1, P is true in

W_2. We will not explicitly go through the axioms for sequences, conditionals, and iterations for Res1, but we will simply note that they would be identical to R2 - R4, with Res1 substituted for Res.

While we want our formalism to be able to handle the problems of reasoning about knowledge and action in as general a way as possible, there will obviously have to be special axioms for particular actions. After all, what effect an action has on the world is a question of physics, not logic. Still, our goal will be to put the minimum necessary amount of information into the axioms for specific actions and to use general principles as much as possible.

To illustrate how information about the physical effects of a specific action would be represented in our system we will work out an example from the blocks world. Suppose we have a table and a number of blocks. Any number of blocks can be on the table, but only one block can be on a given block. We will assume there is one agent in the world, called Hand, which can move a block using the action Puton, if the block being moved has nothing on it and the destination is either the table or another block with nothing on it. We will give three axioms for Puton:

P1. $\forall a_1,x_1,x_2,w_1,w_2$
$(\exists w_2(R(:Do(a_1,:Puton(x_1,x_2)),w_1,w_2)) \equiv$
$((:Block(x_1) \wedge \forall x_3(\neg H(w_1,:On(x_3,x_1))) \wedge$
$((x_1 \neq x_2) \wedge \forall x_3(\neg H(w_1,:On(x_3,x_2)))) \vee :Table(x_2))))$

P2. $\forall a_1,x_1,x_2,w_1,w_2$
$(R(:Do(a_1,:Puton(x_1,x_2)),w_1,w_2) \supset$
$(H(w_2,:On(x_1,x_2)) \wedge \forall x_3((x_2 \neq x_3) \supset \neg H(w_2,:On(x_1,x_3)))))$

P3. $\forall a_1,x_1,x_2,w_1,w_2$
$(R(:Do(a_1,:Puton(x_1,x_2)),w_1,w_2) \supset$
$(\forall int.trm_1(V(w_1,int.trm_1) \equiv V(w_2,int.trm_1)) \wedge$
$\forall int.p_1(\forall x_3(int.p_1 \neq :On(x_1,x_3)) \supset (H(w_1,int.p_1) \equiv H(w_2,int.p_1)))))$

P1 gives the prerequisites for Puton. It is possible to put x_1 on x_2 just in case x_1 is a block with nothing on it, and x_2 is either a different block with nothing on it or is the table. Since we have made the meta-language predicates :Block and :Table independent of any reference to possible worlds, we are treating being a block or a being a table as essential properties of blocks and tables. In the blocks world, this is enforced by the lack of any actions which transform blocks and tables into anything else.

P2 and P3 are frame axioms which describe the effect of of Puton. P2 says that in the world resulting from putting x_1 on x_2, x_1 is on x_2 and is not on anything else. P3 describes what Puton does not change. In P3 $int.trm_1$ is a variable which ranges over the intensional objects corresponding to object-language terms, and $int.p_1$ is a variable which ranges over the intensional objects corresponding to object-language propositions. What P3 asserts, then, is that Puton(x_1,x_2) does not change the extension of any of the basic functions or relations of the language except what x_1 is on.

P3 illustrates the advantages of mapping object-language formulas and terms into intensional objects in the meta-language as we discussed in section 4.3. To make effective use of this axiom, though, we will have to have some knowledge built into the system about what expressions for intensional objects are equal to each other. We will assume that two terms which denote intensional objects and begin with ":" are equal only if they are the same constant or are the same function with equal arguments. So :A will be implicitly unequal to :B, and :On(x_1,x_2) will be equal to :On(x_3,x_4) only if x_1 is equal to x_3 and x_2 is equal to x_4. This allows us to use axioms like P3 without having a large number of inequality axioms for intensional objects.

An action still may affect a great many relations or functions, but usually we can identify a relatively small number in terms of which the others can be defined. For instance, Above could be defined in terms of On:

ABV1. $\forall w_1, \mathrm{Irm}.x_1, \mathrm{Irm}.x_2$
$(T(w_1, \mathrm{Above}(\mathrm{Irm}.x_1, \mathrm{Irm}.x_2)) \equiv$
$(T(w_1, \mathrm{On}(\mathrm{Irm}.x_1, \mathrm{Irm}.x_2)) \vee$
$\exists x_3 (T(w_1, \mathrm{Above}(\mathrm{Irm}.x_1, @(x_3))) \wedge T(w_1, \mathrm{Above}(@(x_3), \mathrm{Irm}.x_2)))))$

If we do not have intensional objects corresponding to propositions like Above(A,B), then we will be forced to use the definition ABV1, and we will not be in danger of using P3 to infer that Above(A,B) is still true after we move B. Most AI problem solving systems have this technique embodied in their programs. What we have done here is to express it formally.

By asserting that P1 - P3 apply to all possible worlds, we are claiming that they apply in all situations in the actual course of events; i.e, they are true at all times. Furthermore, we are claiming that all agents knowthat P1 - P3 apply to all situations, and all agents know that all agents know that they apply to all situations, etc. In other words, the facts about Puton are assumed to be common-knowledge.

We can use these axioms to do "program verification" for the blocks world. For example, we can verify the solution to Sussman's (1973) "anomalous situation" problem. Suppose that block A and block B are on the table, block C is the only thing on A, and nothing is on B or C. We can achieve A on B and B on C by putting C on the table, putting B on C, and putting A on B. This is expressed formally by the following deduction (See figure 5.1 for a diagram of the relevant situations):

Given: True(Block(A))
True(Block(B))
True(Block(C))
True(Table(Tbl))
True(On(A,Tbl))
True(On(B,Tbl))
True(On(C,A))
True(All(X,(Not(Eq(X,C)) => Not(On(X,A)))))
True(All(X,Not(On(X,B))))
True(All(X,Not(On(X,C))))

Prove: True(Res(Do(Hand,(Puton(C,Tbl); (Puton(B,C); Puton(A,B)))),And(On(A,B),On(B,C))))

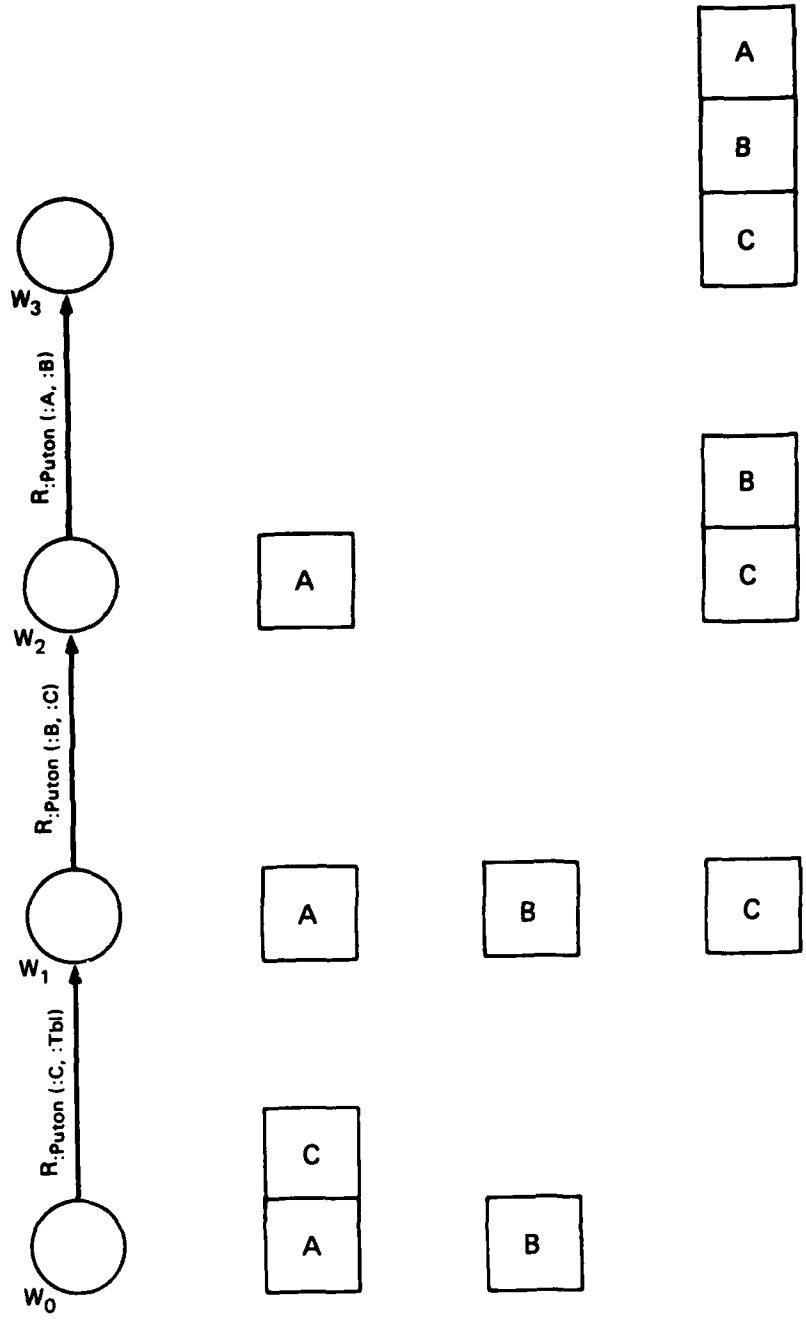


Figure 5.1 A typical blocks-world problem.

1. :Block(:A)	Given,L1,L9b,L11b
2. :Block(:B)	Given,L1,L9b,L11b
3. :Block(:C)	Given,L1,L9b,L11b
4. :Table(:Tbl)	Given,L1,L9b,L11b
5. $H(W_0,$:On(:A,:Tbl))	Given,L1,L9a,L11b
6. $H(W_0,$:On(:B,:Tbl))	Given,L1,L9a,L11b
7. $H(W_0,$:On(:C,:A))	Given,L1,L9a,L11b
8. $(x_1 \neq$:C) $\supset \neg H(W_0,$:On(x_1,:A))	Given,L1,L8,L4,L6,L13,L10,L11b,L9a
9. $\neg H(W_0,$:On(x_1,:B))	Given,L1,L8,L6,L9a,L10,L11b
10. $\neg H(W_0,$:On(x_1,:C))	Given,L1,L8,L6,L9a,L10,L11b

These first ten lines merely translate the premises of the problem from the object language to the meta-language. We are assuming that A, B, C, and Tbl are the standard names for the objects they refer to.

11. $R($:Do(:Hand,:Puton(:C,:Tbl)),$W_0,W_1)$	3,10,4,P1
12. $(x_1 \neq$:Tbl) $\supset \neg H(W_1,$:On(:C,x_1))	11,P2
13. $\forall x_3$(int.$p_1 \neq$:On(:C,x_3)) $\supset$ ($H(W_0$,int.p_1) $\equiv H(W_1$,int.p_1))	11,P3
14. $\neg H(W_1,$:On(:C,:A))	12
15. $\neg H(W_1,$:On(:C,:B))	12
16. $\neg H(W_1,$:On(:C,:C))	12
17. $x_1 =$:C	Ass
18. $\neg H(W_1,$:On(x_1,:A))	14,17
19. $\neg H(W_1,$:On(x_1,:B))	15,17
20. $\neg H(W_1,$:On(x_1,:C))	16,17
21. $(x_1 =$:C) $\supset \neg H(W_1,$:On(x_1,:A))	Dis(17,18)
22. $(x_1 =$:C) $\supset \neg H(W_1,$:On(x_1,:B))	Dis(17,19)
23. $(x_1 =$:C) $\supset \neg H(W_1,$:On(x_1,:C))	Dis(17,20)
24. $x_1 \neq$:C	Ass
25. $\neg H(W_0,$:On(x_1,:A))	8,24
26. $H(W_0,$:On(x_1,x_2)) $\equiv H(W_1,$:On(x_1,x_2))	13,24
27. $\neg H(W_1,$:On(x_1,:A))	25,26
28. $\neg H(W_1,$:On(x_1,:B))	9,26
29. $\neg H(W_1,$:On(x_1,:C))	10,26
30. $(x_1 \neq$:C) $\supset \neg H(W_1,$:On(x_1,:A))	Dis(24,27)
31. $(x_1 \neq$:C) $\supset \neg H(W_1,$:On(x_1,:B))	Dis(24,28)
32. $(x_1 \neq$:C) $\supset \neg H(W_1,$:On(x_1,:C))	Dis(24,29)
33. $\neg H(W_1,$:On(x_1,:A))	21,30
34. $\neg H(W_1,$:On(x_1,:B))	22,31
35. $\neg H(W_1,$:On(x_1,:C))	23,32

Lines 11 - 35 take us through the execution of the first step of the plan. Since nothing is on C and Tbl is a table, there is a possible situation which is the outcome of putting C on Tbl. We call this situation W_1 (line 11). In W_1, C is not on anything other than Tbl (line 12), and everything besides C is where it was in the original situation, W_0 (line 13). We conclude that C is on neither A, B, or C in W_1 (lines 14 - 15). In drawing these conclusions we use an implicit rule that two standard names (e.g. A and Tbl) which are not identical do not have the same referent.

We then do some reasoning by cases. First we suppose that the variable x_1 equals C (line 17). It follows immediately that x_1 is not on A, B, or C in W_1 (lines 21 - 23). Next we assume that x_1 is not equal to C (line 24). We can conclude that x_1 is not on A in W_0 (line 25), and that x_1 is the same place in W_1 as in W_0 (line 26). This means that x_1 is not on A, B, or C in W_1 (lines 30 - 32). Therefore, nothing is on A, B, or C in W_1 (lines 33 - 35).

36. $R(:Do(:Hand,:Puton(:B,:C)),W_1,W_2)$	2,3,34,35,P1
37. $H(W_2,:On(:B,:C))$	36,P2
38. $(x_1 \neq :C) \supset \neg H(W_2,:On(:B,x_1))$	36,P2
39. $\forall x_3(int.p_1 \neq :On(:B,x_3)) \supset (H(W_1,int.p_1) \equiv H(W_2,int.p_1))$	36,P3
40. $\neg H(W_2,:On(:B,:A))$	38
41. $\neg H(W_2,:On(:B,:B))$	38
42. $x_1 = :B$	Ass
43. $\neg H(W_2,:On(x_1,:A))$	40,42
44. $\neg H(W_2,:On(x_1,:B))$	41,42
45. $(x_1 = :B) \supset \neg H(W_2,:On(x_1,:A))$	Dis(42,43)
46. $(x_1 = :B) \supset \neg H(W_2,:On(x_1,:B))$	Dis(42,44)
47. $x_1 \neq :B$	Ass
48. $H(W_1,:On(x_1,x_2)) \equiv H(W_2,:On(x_1,x_2))$	39,47
49. $\neg H(W_2,:On(x_1,:A))$	33,48
50. $\neg H(W_2,:On(x_1,:B))$	34,48
51. $(x_1 \neq :B) \supset \neg H(W_2,:On(x_1,:A))$	Dis(47,49)
52. $(x_1 \neq :B) \supset \neg H(W_2,:On(x_1,:B))$	Dis(47,50)
53. $\neg H(W_2,:On(x_1,:A))$	45,51
54. $\neg H(W_2,:On(x_1,:B))$	45,52

Lines 36 - 54 describe the execution of the second step. Since nothing is on B or C in W_1, it is possible to put B on C (line 36). In the outcome of this action, W_2, B is on only C (lines 37 - 38), and everything besides B is where it was in W_1 (line 39). In particular, B is not on A or B (lines 40 - 41). As we did for C in W_1, we reason by cases that nothing, whether or not it is B, is on A or B in W_2 (lines 42 - 54).

55. R(:Do(:Hand,:Puton(:A,:B)),W_2,W_3)	1,2,53,54,P1
56. H(W_3,:On(:A,:B))	55,P2
57. $\forall x_3$(int.p_1 ≠ :On(:A,x_3)) ⊃ (H(W_2,int.p_1) ≡ H(W_3,int.p_1))	55,P3
58. H(W_2,:On(:B,x_2)) ≡ H(W_3,:On(:B,x_2)))	57
59. H(W_3,:On(:B,:C))	37,58
60. T(W_3,And(On(A,B),On(B,C)))	56,59,L2,L9a,L11b
61. R(D(W_2,Do(Hand,Puton(A,B))),W_2,W_3)	55,L11b,L12b
62. T(W_2,Res(Do(Hand,Puton(A,B)),And(On(A,B),On(B,C))))	55,61,R1
63. R(D(W_1,Do(Hand,Puton(B,C))),W_1,W_2)	36,L11b,L12b
64. T(W_2,Res(Do(Hand,Puton(B,C)), Res(Do(Hand,Puton(A,B)),And(On(A,B),On(B,C)))))	62,63,R1
65. T(W_2,Res(Do(Hand,(Puton(B,C); Puton(A,B))), And(On(A,B),On(B,C))))	64,R2
66. R(D(W_0,Do(Hand,Puton(C,Tbl))),W_0,W_1)	11,L11b,L12b
67. T(W_0,Res(Do(Hand,Puton(C,Tbl)), Res(Do(Hand,(Puton(B,C); Puton(A,B))),And(On(A,B),On(B,C)))))	65,66,R1
68. T(W_0,Res(Do(Hand,(Puton(C,Tbl); (Puton(B,C); Puton(A,B)))), And(On(A,B),On(B,C))))	67,R2
69. True(Res(Do(Hand,(Puton(C,Tbl); (Puton(B,C); Puton(A,B)))), And(On(A,B),On(B,C))))	68,L1

Now we consider the final step. Since nothing is on A or B in W_2, it is possible to put A on B, bringing about W_3 (line 55). We know that in this situation, A is on B (line 56), and everything else is where it was before (line 57). In particular, B is where it was before (line 58), on C (line 59). At this point we are essentially done. All that remains is to translate our results back into the object language (lines 60 - 69).

This example shows that we can express the usual AI approach to actions in a rigorous possible-world formalism. In the rest of this chapter we will show how to formalize the interactions between knowledge and action within the same framework.

5.2 Formalizing the Dependence of Action on Knowledge

In section 3.2 we discussed the ways in which being able to act effectively depends on knowledge. The conclusion we reached was that in general it is not neccessary to regard particular actions as having knowledge preconditions, but that using any action to achieve a goal requires knowing what action to take. To formalize this idea we introduced the modal operator Can(A,Act,P) to mean that the agent denoted by A can use the action described by Act to achieve P, in the sense that A knows how to achieve P by performing Act. In section 4.1 we argued that the most natural way to specify Can formally is by a recursive definition in terms of object-language expressions. That definition is given by axioms C1 - C4:

C1. $\forall w_1, trm.a_1, trm.act_1, p_1$
$(T(w_1, Know(trm.a_1, And(Eq(@(D(w_1, trm.act_1)), trm.act_1),$
$Res(Do(@(D(w_1, trm.a_1)), trm.act_1), p_1)))) \supset$
$T(w_1, Can(trm.a_1, trm.act_1, p_1)))$

C2. $\forall w_1, trm.a_1, trm.act_1, trm.act_2, p_1$
$(T(w_1, Can(trm.a_1, (trm.act_1; trm.act_2), p_1)) \equiv$
$T(w_1, Can(trm.a_1, trm.act_1, Can(@(D(w_1, trm.a_1)), trm.act_2, p_1))))$

C3. $\forall w_1, trm.a_1, trm.act_1, trm.act_2, p_1, p_2$
$(T(w_1, Can(trm.a_1, If(p_1, trm.act_1, trm.act_2), p_2)) \equiv$
$((T(w_1, Know(trm.a_1, p_1)) \wedge T(w_1, Can(trm.a_1, trm.act_1, p_2))) \vee$
$(T(w_1, Know(trm.a_1, Not(p_1))) \wedge T(w_1, Can(trm.a_1, trm.act_2, p_2)))))$

C4. $\forall w_1, trm.a_1, trm.act_1, p_1, p_2$
$(T(w_1, Can(trm.a_1, While(p_1, trm.act_1), p_2)) \equiv$
$T(w_1, Can(trm.a_1, If(p_1, (trm.act_1; While(p_1, tr.act_1)), Nil), p_2)))$

C1 says that the agent named by $trm.a_1$ can achieve p_1 by doing the action named by $trm.act_1$, if he knows what action $trm.act_1$ names and knows that if he does the action named by $trm.act_1$, P will result. The first of these conditions is expressed by saying that there is a rigid designator (an executable description) for an action which the agent knows

describes the same action as trm.act$_1$. C2 says that the agent named by trm.a$_1$ can achieve p_1 by doing the sequence of actions (trm.act$_1$; trm.act$_2$), just in case by doing trm.act$_1$, he can achieve a state where by doing trm.act$_2$, he can achieve P.

C1 and C2 together imply that in performing a sequence of actions, an agent is not required to know precisely what is to be done in the second part of the sequence until the first part has been carried out. He does have to know *some* description of the second part of the sequence, but not an executable description. This will allow for sequences of actions in which the early stages are actions that gather information to find out what to do in the later stages. In both these axioms trm.a$_1$ is converted to a rigid designator to guarantee that the agent knows that *he* is the one who is able to achieve the result described.

C3 says that the agent trm.a$_1$ can achieve p_2 by doing If(p_1,trm.act$_1$,trm.act$_2$), just in case he knows that p_1 is true and he can achieve p_2 by doing trm.act$_1$, or he knows that p_1 is false and he can achieve p_2 by doing trm.act$_2$. C4 says that the agent trm.a$_1$ can achieve p_2 by doing While(p_1,trm.act$_1$), just in case he can achieve p_2 by doing trm.act$_1$ as long as p_1 remains true.

We will illustrate the use of the operator Can with the sample problems from chapter 1 about opening safes. First, we need some facts about dialing combinations:

D1. $\forall a_1,x_1,x_2,w_1$
$(\exists w_2(R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \equiv$
$(\exists w_3(x_1 = V(w_3,:Comb(x_2))) \wedge :Safe(x_2) \wedge H(w_1,:At(a_1,X_2))))$

D2. $\forall a_1,x_1,x_2,w_1,w_2$
$R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \supset$
$(((x_1 = V(w_1,:Comb(x_2))) \supset H(w_2,:Open(x_2))) \wedge$
$(((x_1 \neq V(w_1,:Comb(x_2))) \wedge \neg H(w_1,:Open(x_2))) \supset \neg H(w_2,:Open(x_2))) \wedge$
$(H(w_1,:Open(x_2)) \supset H(w_2,:Open(x_2)))))$

We will let the action Dial refer to the entire sequence of turning the dial of the safe and

then attempting to turn the handle and open the safe. D1 says that an agent can dial x_1 on x_2 if it is possible for x_1 to be the combination of x_2, and x_2 is a safe, and the agent is at the same place as the safe. D2 tells how dialing a combination affects whether the safe is open: if the combination is the combination of the safe, then the safe will be open; if it is not the combination of the safe and the safe was locked, the safe stays locked; if the safe was already open, it stays open. Notice that we have asserted that these facts are true in all possible worlds. This is lets us infer that they are always true, everyone knows that they are always true, everyone knows that everyone knows that they are always true, etc.

Besides the axioms for Dial, we will need one additional fact in order to work out our examples:

A1. $\forall w_1,a_1,x_1 (H(w_1,:At(a_1,x_1)) \supset \forall w_2(K(a_1,w_1,w_2) \supset H(w_2,:At(a_1,x_1))))$

A1 says that when an agent is at the same place as some object, he knows that he is at the same place as the object. This is not really true, of course; the object may be hidden so that the person doesn't know that it is there. We justify our use of A1 in our examples by the observation that if a person were asked under what conditions it is possible to open a safe, he probably would not consider the possibility that the agent might be at the location of the safe and not know it. Actually, dealing with all the unlikely ways in which a plan might fail (dubbed the *qualification problem* by McCarthy (1977)) is a very serious problem in AI for which no one seems to have a good solution, and it is beyond the scope of this thesis to find one.

Our main example to illustrate the use of Can is to show that if John knows the combination to the safe Sf_1, and he is in the same place as Sf_1, then he can open the safe by dialing the combination. The interesting point is that knowing the combination of the safe comes in, not as a specific precondition of the action, but as a way of satisfying the

general conditions on Can. The possible-world structure for this proof is pictured in figure 5.2.

Given: True(Safe(Sf_1))
True(At(John,Sf_1))
True(Exist(?X_1,Know(John,Eq(?X_1,Comb(Sf_1)))))

Prove: True(Can(John,Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))

1.	:Safe(Sf_1)	Given,L1,L9b,L11b
2.	H(W_0,:At(:John,:Sf_1))	Given,L1,L9a,L11b
3.	$\exists x_1$ (T(W_0,Know(John,Eq(@(x_1),Comb(Sf_1)))))	Given,L1,L7
4.	T(W_0,Know(John,Eq(@(:C),Comb(Sf_1))))	3
5.	K(D(W_0,John),W_0,w_1) ⊃ T(w_1,Eq(@(:C),Comb(Sf_1)))	4,K1
6.	K(:John,W_0,w_1) ⊃ (D(w_1,@(:C)) = D(w_1,Comb(Sf_1))	5,L11b,L13
7.	K(:John,W_0,w_1) ⊃ (:C = V(w_1,:Comb(:Sf_1)))	6,L10,L12a,L11b
8.	K(:John,W_0,w_1) ⊃ H(w_1,:At(:John,:Sf_1))	2,A1
9.	:C = V(W_0,:Comb(:Sf_1))	7,K2
10.	K(:John,W_0,w_1)	Ass
11.	:C = V(w_1,:Comb(:Sf_1))	7,10
12.	V(W_0,:Comb(:Sf_1)) = V(w_1,:Comb(:Sf_1))	9,11
13.	:Dial(V(W_0,:Comb(:Sf_1)),:Sf_1) = :Dial(V(w_1,:Comb(:Sf_1)),:Sf_1)	12
14.	D(W_0,Dial(Comb(Sf_1),Sf_1)) = D(w_1,Dial(Comb(Sf_1),Sf_1))	13,L11b,L12a
15.	D(w_1,@(D(W_0,Dial(Comb(Sf_1),Sf_1)))) = D(w_1,Dial(Comb(Sf_1),Sf_1))	14,L10
16.	T(w_1,Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)))	15,L13
17.	H(w_1,:At(:John,:Sf_1))	8,10
18.	R(:Do(:John,:Dial(V(w_1,:Comb(:Sf_1)),:Sf_1)),w_1,W_2)	1,17,D1
19.	H(W_2,:Open(:Sf_1))	18,D2
20.	T(W_2,Open(Sf_1))	19,L11b,L9a
21.	R(:Do(D(W_0,John),:Dial(V(w_1,:Comb(:Sf_1)),:Sf_1)),w_1,W_2)	18,L11b
22.	R(:Do(D(w_1,@(D(W_0,John))),:Dial(V(w_1,:Comb(:Sf_1)),:Sf_1)),w_1,W_2)	21,L10
23.	T(w_1,Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))	22,L11b,L12a,12b,R1
24.	T(w_1,And(Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))	16,23,L2
25.	K(:John,W_0,w_1) ⊃ T(w_1,And(Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))	Dis(10,24)
26.	T(W_0,Know(John, And(Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))))	25,L11b,K1
27.	T(W_0,Can(John,Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))	26,C1
28.	True(Can(John,Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))	27,L1

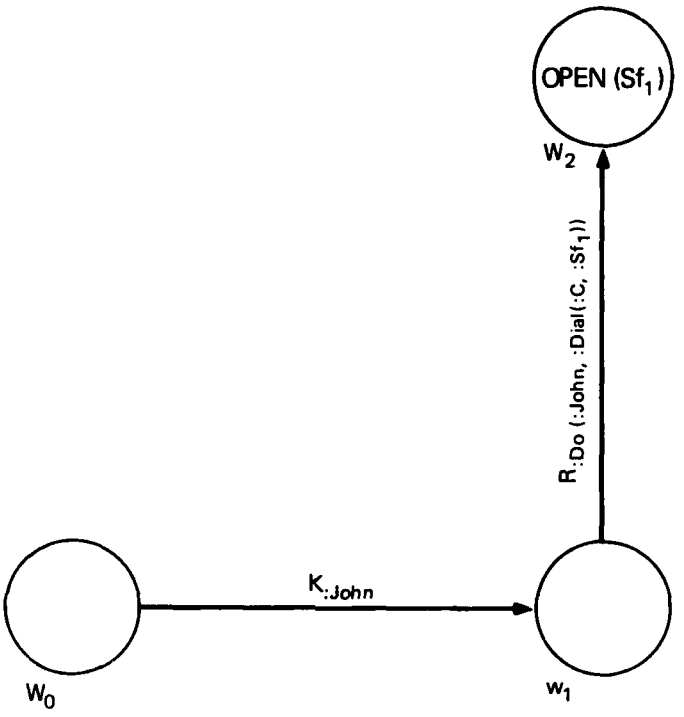


Figure 5.2 "John can open Sf_1 by dialing Comb(Sf_1)."

Line 1 translates into the meta-language the premise that Sf_1 is a safe, and line 2 translates the premise that John is at the same place as the safe. The third premise, that John knows the combination to the safe, is handled by lines 3 - 7. We have stretched out the translation of this premise into the meta-language to expose the details of how the existential quantifier is handled. There is something which John knows to be the combination of the safe, and we choose to call that thing :C (line 4). Since John is at the same place as the safe, we conclude that he knows he is at the same place as the safe (line 8). Since John knows :C to be the combination of the safe, :C must, in fact, be the combination to the safe (line 9).

To make deductions about what John knows, we let w_1 be a typical world which is possible according to what John knows (line 10). Since John knows that :C is the

combination of the safe, :C is the combination of the safe in w_1 (line 11). Therefore, the combination of the safe in w_1 is the same as the combination of the safe in the actual world, W_0 (line 12), and the action of dialing the combination of the safe in w_1 is the same as the action of dialing the combination of the safe in W_0 (line 16).

Since John is at the same place as the safe in w_1 (line 17), and since the combination of the safe is a is a possible combination, it is possible for John to dial the combination of the safe in w_1 (line 18). We will call the resulting situation W_2. Since the combination dialed is, in fact, the combination of the safe in w_1, the safe is open in W_2 (line 19). So, John dialing the combination of the safe in w_1 would result in the safe being open (lines 20 - 23). Translating everything back into the object language, John knows which action dialing the combination of the safe is, and he knows that dialing the combination of the safe will result in the safe being open (lines 24 - 26). Therefore, John can open the safe by dialing the combination (lines 27 and 28).

The major point of this example is that in deducing that John can open the safe, we did not have as an explicit piece of knowledge the fact that knowing the combination is one of the requirements for opening a safe. Instead we used a much more general piece of knowledge, the fact that in order to achieve any goal it is necessary to have an executable description of a procedure that causes the goal to be satisfied. In this case, the combination of the safe is part of the executable description of the procedure for opening safes.

In this section we have looked at how the possibility of taking effective action depends on having the right knowledge. In the next section we will examine how actions can affect what an agent knows.

5.3 Formalizing the Effects of Action on Knowledge

In section 3.3 we explained how our theory handles the effects of an action on the knowledge of the agent. The basic idea was to represent these effects by a pattern of accessibility relationships among various possible worlds. We distinguished actions on the basis of whether or not they are knowledge-producing, a knowledge-producing action being one where after performing the action the agent would know more about the resulting situation than he did before performing the action. We showed how to account for this distinction in terms of the possible-world semantics for knowledge and action.

The formalism that we have now developed is adequate to capture the effects on knowledge of both types of actions. As an example of an action which is not knowledge-producing, we can use the blocks world operation Puton that we formalized in section 5.1. We can extend our formalization of Puton to include its effect on the knowledge of the agent by adding the following axiom to those we have already given:

P4. $\forall a_1,x_1,x_2,w_1,w_2$
$(R(:Do(a_1,:Puton(x_1,x_2)),w_1,w_2) \supset$
$\forall w_3(K(a_1,w_2,w_3) \equiv \exists w_4(K(a_1,w_1,w_4) \wedge R(:Do(a_1,:Puton(x_1,x_2)),w_4,w_3))))$

P4 says that if w_2 is the situation which results from a_1 putting x_1 on x_2 in w_1, then the worlds which are compatible with what a_1 knows in w_2 are exactly those worlds (w_3) which are the result of a_1 puting x_1 on x_2 in one of the worlds (w_4) which are compatible with what a_1 knows in w_1. This is exactly the situation that was illustrated by figure 3.2. We can think of a_1 puting x_1 on x_2 as having the effect of filtering out from the courses of events compatible with what a_1 knows in w_1 all those in which some other event occurs at that point in time.

Using P4, we can show that after performing Puton(A,B), an agent would know that A is

on B. In this example we will not be interested in showing that the agent is able to put A on B, so we will use the weak modal operator for actions, Resl.

Prove: True(Resl(Do(Hand,Puton(A,B)),Know(Hand,On(A,B))))

1.	$R(:Do(:Hand,:Puton(:A,:B)),W_0,w_1)$	Ass
2.	$K(:Hand,w_1,w_2)$	Ass
3.	$\exists w_3(K(:Hand,W_0,w_3) \wedge R(:Do(:Hand,:Puton(:A,:B)),w_3,w_2))$	1,2,P4
4.	$R(:Do(:Hand,:Puton(:A,:B)),W_3,w_2)$	3
5.	$H(w_2,:On(:A,:B))$	4,P2
6.	$T(w_2,On(A,B))$	5,L9a,L11b,L12b
7.	$K(:Hand,w_1,w_2) \supset T(w_2,On(A,B))$	Dis(2,6)
8.	$T(W_1,Know(Hand,On(A,B)))$	7,L11b,K1
9.	$R(:Do(:Hand,:Puton(:A,:B)),W_0,w_1) \supset T(W_1,Know(Hand,On(A,B)))$	Dis(1,8)
10.	$T(W_0,Resl(Do(Hand,Puton(A,B)),Know(Hand,On(A,B))))$	9,L11b,L12b,R6
11.	True(Resl(Do(Hand,Puton(A,B)),Know(Hand,On(A,B))))	10,L1

We start by assuming that w_1 is the world which results from Hand putting A on B in W_0 (line 1). Notice that we are assuming that Hand, A, and B are all rigid designators. The assumption that Hand is a rigid designator is merely a convenience. A and B, on the other hand, must be rigid designators for the conclusion to be valid in the exact form that it is stated. If A and B were not rigid designators, Hand might not recognize them as referring to the objects he acted upon, so we would have to substitute something like:

$$Exist(?X_1,And(Eq(?X_1,A),Exist(?X_2,And(Eq(?X_2,B),Know(Hand(On(?X_1,?X_2)))))))$$

for Know(Hand,On(A,B)) in the conclusion. This would make the proof longer, but not really any harder.

We let w_2 be a typical world which is possible according to what Hand knows in w_1. P4 implies that w_2 must be the result of Hand putting A on B in some other world, say W_3 (line 4). So, A must be on B in w_2 (lines 5 - 6); hence in w_1, Hand knows that A is on B (lines 7 - 8). This leads to the conclusion that in the result of putting A on B, Hand knows that A is

on B (lines 9 - 11). At a very general level, the argument of this proof is that Hand knows that he has put A on B, and he understands the effects of putting A on B, so he must know that A is on B.

The analysis we presented for knowledge-producing actions is similar to that for actions that are not knowledge-producing except that we also take into account the knowledge gained. We can use the action Dial from the previous section as an example of knowledge-producing action if we assume that after trying to open a safe by dialing a combination the agent knows whether he has succeeded. This can be a genuine increase in his knowledge, since he might not know beforehand whether he would succeed. We can express this fact by adding D3 to our axioms for Dial:

D3. $\forall a_1,x_1,x_2,w_1,w_2$
$(R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \supset$
$\forall w_3(K(a_1,w_2,w_3) \equiv ((H(w_2,:Open(x_2)) \equiv H(w_3,:Open(x_2))) \wedge$
$\exists w_4(K(a_1,w_1,w_4) \wedge R(:Do(a_1,:Dial(x_1,x_2)),w_4,w_3)))))$

D3 describes how dialing affects the knowledge of the dialer. Roughly it says that the agent knows he has done the dialing, and he now knows whether the safe is open. More precisely, it says that the worlds that are now possible as far as he knows are exactly those which are the result of doing the action in some world which was previously possible according to what he knew, and which agree with the world which actually results from trying to open the safe as to whether the safe is open. This is the type of situation that was pictured in figure 3.3.

We can show that this axiom implies that after trying to open a safe an agent would know whether the safe were open:

Given: $True(Res(Do(John,Dial(C_1,Sf_1)),Open(Sf_1)))$

Prove: $True(Res(Do(John,Dial(C_1,Sf_1)),Know(John,Open(Sf_1))))$

1. $T(W_0,Res(Do(John,Dial(C_1,Sf_1)),Open(Sf_1)))$	Given,L1
2. $\exists w_1 (R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,w_1) \wedge H(w_1,:Open(:Sf_1))$	1,R1,L11b,L12b,L9a
3. $R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,W_1)$	2
4. $H(W_1,:Open(:Sf_1))$	2
5. $K(:John,W_1,w_2)$	Ass
6. $H(W_1,:Open(:Sf_1)) \equiv H(w_2,:Open(:Sf_1))$	3,5,D3
7. $H(w_2,:Open(:Sf_1))$	4,6
8. $T(w_2,Open(Sf_1))$	7,L11b,L9a
9. $K(:John,W_1,w_2) \supset T(w_2,Open(Sf_1))$	Dis(5,8)
10. $T(W_1,Know(John,Open(Sf_1)))$	9,L11b,K1
11. $T(W_0,Res(Do(John,Dial(C_1,Sf_1)),Know(John,Open(Sf_1))))$	3,L11b,L12b,10,R1
12. $True(Res(Do(John,Dial(C_1,Sf_1)),Know(John,Open(Sf_1))))$	11,L1

In the first case, we assume that John dialing the combination C_1 on the safe Sf_1 results in the safe being open. Line 2 translates this premise into the meta-language. For the same reasons as in the previous example, we assume that John, C_1, and Sf_1 are rigid designators. We let W_1 be the world which results from John trying to open Sf_1 (line 3), so the safe is open in W_1 (line 4). We let w_2 be a typical world which is possible according to what John knows in W_1 (line 5). D3 implies that w_2 must agree with W_1 as to whether the safe is open (line 6), so the safe must be open in w_2 (lines 7 - 8). Therefore, in W_1 John knows that the safe is open (lines 9 - 10), so after trying to open the safe, John knows that the safe is open (lines 11 - 12).

Given: $True(Res(Do(John,Dial(C_1,Sf_1)),Not(Open(Sf_1))))$

Prove: $True(Res(Do(John,Dial(C_1,Sf_1)),Know(John,Not(Open(Sf_1)))))$

1. $T(W_0,Res(Do(John,Dial(C_1,Sf_1)),Not(Open(Sf_1))))$	Given,L1
2. $\exists w_1 (R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,w_1) \wedge \neg H(w_1,:Open(:Sf_1))$	1,R1,L11b,L12b,L6,L9a
3. $R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,W_1)$	2
4. $\neg H(W_1,:Open(:Sf_1))$	2
5. $K(:John,W_1,w_2)$	Ass
6. $H(W_1,:Open(:Sf_1)) \equiv H(w_2,:Open(:Sf_1))$	3,5,D3
7. $\neg H(w_2,:Open(:Sf_1))$	4,6
8. $T(w_2,Not(Open(Sf_1)))$	7,L11b,L9a,L6

9. $K(:John,W_1,w_2) \supset T(w_2,Not(Open(Sf_1)))$	Dis(5,8)
10. $T(W_1,Know(John,Not(Open(Sf_1))))$	9,L11b,K1
11. $T(W_0,Res(Do(John,Dial(C_1,Sf_1)),Know(John,Not(Open(Sf_1)))))$	3,L11b,L12b,10,R1
12. $True(Res(Do(John,Dial(C_1,Sf_1)),Know(John,Not(Open(Sf_1)))))$	11,L1

The proof of the second case is identical in form to the proof of the first case. This time we are given that the safe is not open in the result of John trying to open it, so the safe is not open in W_1 (line 4). This in turn implies that the safe is not open in w_2 (line 7), so after trying to open the safe, John knows that the safe is not open (line 12).

A more interesting example is the second of our benchmark problems from chapter 1: showing that after trying to open the safe Sf_1, which he knows to be locked, by dialing C_1, John would know whether C_1 is the combination of Sf_1. This proof depends on the facts that after trying to open the safe, John knows that he tried to open the safe, he knows whether the safe is open, and he understands how the safe being open depends on whether the combination he dialed is the combination of the safe. In addition, John must know that trying to open the safe does not change the combination. To show this we need an additional frame axiom for Dial:

D4. $\forall a_1,x_1,x_2,w_1,w_2$
$(R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \supset$
$(\forall int.trm_1 (V(w_1,int.trm_1) \equiv V(w_2,int.trm_1)) \wedge$
$\forall int.p_1 ((int.p_1 \neq :Open(x_2)) \supset (H(w_1,int.p_1) \equiv H(w_2,int.p_1)))))$

D4 says that Dial doesn't affect any basic function or relation other than whether the safe is open. Therefore Dial does not change the combination of the safe.

We will divide the proof into two cases, according to whether or not C_1 is the combination of Sf_1. The structure of the possible worlds mentioned in the proof of the first case is illustrated in figure 5.3.

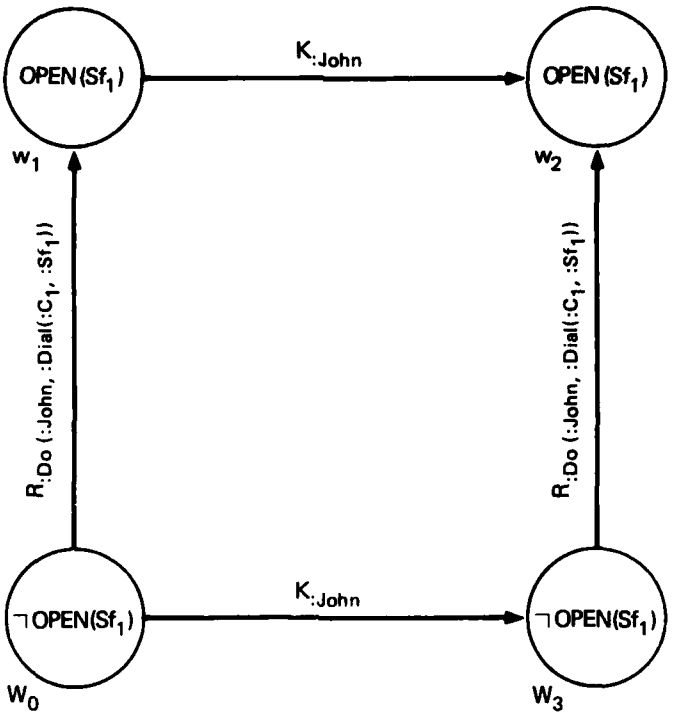


Figure 5.3 "C_1 is the combination of Sf_1."

Given: True(Know(John,Not(Open(Sf_1))))
True(Eq(C_1,Comb(Sf_1)))

Prove: True(Real(Do(John,Dial(C_1,Sf_1)),Know(John,Eq(C_1,Comb(Sf_1)))))

1.	K(:John,W_0,w_1) ⊃ ¬H(w_1,:Open(:Sf_1))	Given,L1,K1,L4,L9a,L11b
2.	:C_1 = V(W_0,:Comb(:Sf_1))	Given,L1,L13,L11b,L12b
3.	R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,w_1)	Ass
4.	H(w_1,:Open(:Sf_1))	3,2,D2
5.	K(:John,w_1,w_2)	Ass
6.	H(w_1,:Open(:Sf_1)) ≡ H(w_2,:Open(Sf_1))	3,5,D3
7.	H(w_2,:Open(:Sf_1))	4,6
8.	∃w_3(K(:John,W_0,w_3) ∧ R(Do(:John,:Dial(:C_1,:Sf_1)),w_3,w_2))	3,5,D3
9.	K(:John,W_0,W_3)	8
10.	¬H(W_3,:Open(:Sf_1))	1,9
11.	R(:Do(:John,:Dial(:C_1,:Sf_1)),W_3,w_2)	8

12.	$((:C_1 \neq V(W_3,:Comb(:Sf_1))) \wedge \neg H(W_3,:Open(Sf_1))) \supset \neg H(w_2,:Open(:Sf_1))$	11,D2
13.	$(:C_1 = V(W_3,:Comb(:Sf_1))) \vee H(W_3,:Open(Sf_1))$	7,12
14.	$:C_1 = V(W_3,:Comb(:Sf_1))$	10,13
15.	$V(W_3,:Comb(:Sf_1)) = V(w_2,:Comb(:Sf_1))$	11,D4
16.	$:C_1 = V(w_2,:Comb(:Sf_1))$	14,15
17.	$T(w_2,Eq(C_1,Comb(Sf_1)))$	16,L11b,L12b,L13
18.	$K(:John,w_1,w_2) \supset T(w_2,Eq(C_1,Comb(Sf_1)))$	Dis(5,17)
19.	$T(w_1,Know(John,Eq(C_1,Comb(Sf_1))))$	18,L11b,K1
20.	$R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,w_1) \supset T(w_1,Know(John,Eq(C_1,Comb(Sf_1))))$	Dis(3,19)
21.	$T(W_0,Resl(Do(John,Dial(C_1,Sf_1)),Know(John,Eq(C_1,Comb(Sf_1)))))$	20,L11b,L12b,R6
22.	$True(Resl(Do(John,Dial(C_1,Sf_1)),Know(John,Eq(C_1,Comb(Sf_1)))))$	21,L1

Lines 1 and 2 translate into the meta-language the premises that John knows the safe is locked and that C_1 is the combination to the safe. We let w_1 be the result of John trying to open the safe in W_0 (line 3). Since C_1 is the combination to the safe, the safe will be open in w_1 (line 4). We then let w_2 be a typical world which is possible according to what John knows in w_1 (line 5). D3 implies that w_2 must agree with W_1 as to whether the safe is open (line 6), so the safe must be open in w_2 (line 7). D3 also implies that w_2 must be the result John trying to open the safe in some world, say W_3, which is possible according to what John knows in W_0 (lines 8 - 9,11). Since in W_0, John knows that the safe is locked, the safe must be locked in W_3 (line 10). But according to D2, if C_1 were not the combination of the safe in W_3, since the safe is locked in W_3, the safe would still be locked in w_2 (line 12). Since the safe is open in w_2, C_1 must be the combination of the safe in W_3 (lines 13 - 14). Since trying to open a safe does not change the combination (line 15), C_1 is still the combination of the safe is w_2 (lines 16 - 17). Therefore, in w_1 John knows that C_1 is the combination of the safe (lines 18 - 19); i.e., after trying to open the safe John knows that C_1 is the combination of the safe (lines 20 - 22).

Given: True(Know(John,Not(Open(Sf_1))))
True(Not(Eq(C_1,Comb(Sf_1))))

Prove: True(Res1(Do(John,Dial(C_1,Sf_1)),Know(John,Not(Eq(C_1,Comb(Sf_1))))))

1.	$K(:John,W_0,w_1) \supset \neg H(w_1,:Open(:Sf_1))$	Given,L1,K1,L4,L9a,L11b
2.	$:C_1 \neq V(W_0,:Comb(:Sf_1))$	Given,L1,L6,L13,L11b,L12b
3.	$\neg H(W_0,:Open(:Sf_1))$	1,K2
4.	$R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,w_1)$	Ass
5.	$\neg H(w_1,:Open(:Sf_1))$	4,2,3,D2
6.	$K(:John,w_1,w_2)$	Ass
7.	$H(w_1,:Open(:Sf_1)) \equiv H(w_2,:Open(Sf_1))$	4,6,D3
8.	$\neg H(w_2,:Open(:Sf_1))$	5,7
9.	$\exists w_3(K(:John,W_0,w_3) \wedge R(Do(:John,:Dial(:C_1,:Sf_1)),w_3,w_2))$	4,6,D3
10.	$R(:Do(:John,:Dial(:C_1,:Sf_1)),W_3,w_2)$	9
11.	$(:C_1 \equiv V(W_3,:Comb(:Sf_1))) \supset H(w_2,:Open(:Sf_1))$	10,D2
12.	$:C_1 \neq V(W_3,:Comb(:Sf_1))$	8,11
13.	$V(W_3,:Comb(:Sf_1)) \equiv V(w_2,:Comb(:Sf_1))$	10,D4
14.	$:C_1 \neq V(w_2,:Comb(:Sf_1))$	12,13
15.	$T(w_2,Not(Eq(C_1,Comb(Sf_1))))$	14,L11b,L12b,L13,L6
16.	$K(:John,w_1,w_2) \supset T(w_2,Not(Eq(C_1,Comb(Sf_1))))$	Dis(6,15)
17.	$T(w_1,Know(John,Not(Eq(C_1,Comb(Sf_1)))))$	16,L11b,K1
18.	$R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,w_1) \supset T(w_1,Know(John,Not(Eq(C_1,Comb(Sf_1)))))$	Dis(4,17)
19.	$T(W_0,Res1(Do(John,Dial(C_1,Sf_1)),Know(John,Not(Eq(C_1,Comb(Sf_1))))))$	18,L11b,L12b,R6
20.	$True(Res1(Do(John,Dial(C_1,Sf_1)),Know(John,Not(Eq(C_1,Comb(Sf_1))))))$	19,L1

The second case is proved very much like the first. Figure 5.4 gives the possible-world structure for this case. Lines 1 and 2 translate into the meta-language the premises that John knows the safe is locked and that C_1 is not the combination to the safe. We note that since John knows the safe is locked, the safe must be locked (line 3). We let w_1 be the result of John trying to open the safe in W_0 (line 4). Since C_1 is not the combination to the safe, and the safe is locked in W_0, the safe will remain locked in w_1 (line 5). We then let w_2 be a typical world which is possible according to what John knows in w_1 (line 6). D3 implies that w_2 must agree with W_1 as to whether the safe is open (line 7), so the safe must

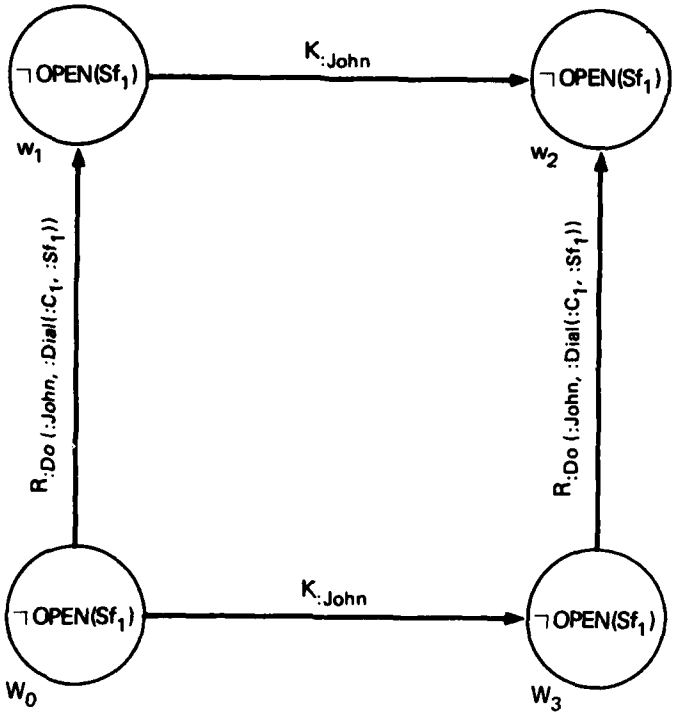


Figure 5.4 "C_1 is not the combination of Sf_1."

be locked in w_2 (line 8). D3 also implies that w_2 must be the result John trying to open the safe in some world, say W_3, which is possible according to what John knows in W_0 (lines 9 - 10). According to D2, if C_1 were the combination of the safe in W_3, the safe would be open in w_2 (line 11). Since the safe is still locked in w_2, C_1 must not be the combination of the safe in W_3 (line 12). Since trying to open a safe does not change the combination (line 13), C_1 is not the combination of the safe is w_2, either (lines 14 - 15). Therefore, in w_1 John knows that C_1 is not the combination of the safe (lines 16 - 17); i.e., after trying to open the safe John knows that C_1 is not the combination of the safe (lines 18 - 20).

This example is a good illustration of the power to be gained from using a rigorous logical formalism. The conclusion of this proof was not explicitly, or even consciously, built-

in to the axioms used in the proof. With the more ad hoc representation schemes that are frequently used in AI, it often seems that an additional fact is required for each new inference that is made. By making a thorough analysis of the problem domain and using a powerful deductive formalism, we have created a much more robust system.

5.4 An Example of Acquiring Knowledge Required for an Action

We conclude this chapter by considering the last of the sample problems from chapter 1. This is example shows how one step of a plan can produce knowledge which is necessary to carry out the rest of the plan. One way of obtaining such knowledge is to read it somewhere. To formalize this, we need a new action Read, a predicate Reads to say that an agent can read, and the operator Info, to say what information the thing being read contains.

INF1. $\forall w_1,trm.x_1,exp_1$
$(T(w_1,Info(trm.x_1,exp_1)) \equiv (exp_1 \equiv V(w_1,:Info(D(w_1,trm.x_1)))))$

RD1. $\forall a_1,x_1,w_1,w_2$
$(\exists w_2(R(:Do(a_1,:Read(x_1)),w_1,w_2)) \equiv$
$(H(w_1,:Reads(a_1)) \wedge H(w_1,:At(a_1,x_1))))$

RDS1. $\forall w_1,a_1(H(w_1,:Reads(a_1)) \supset \forall w_2(K(a_1,w_1,w_2) \supset H(w_2,:Reads(a_1))))$

RD2. $\forall a_1,x_1,w_1,w_2$
$(R(:Do(a_1,:Read(x_1)),w_1,w_2) \supset$
$\forall w_3(K(a_1,w_2,w_3) \equiv ((V(w_2,:Info(x_1)) \equiv V(w_3,:Info(x_1))) \wedge$
$\exists w_4(K(a_1,w_1,w_4) \wedge R(:Do(a_1,:Read(x_1)),w_4,w_3)))))$

RD3. $\forall a_1,x_1,w_1,w_2$
$(R(:Do(a_1,:Read(x_1)),w_1,w_2) \supset$
$(\forall int.trm_1(V(w_1,int.trm_1) \equiv V(w_2,int.trm_1)) \wedge$
$\forall int.p_1(H(w_1,int.p_1) \equiv H(w_2,int.p_1))))$

To represent that an object has information written on it, we introduce the operator Info into the object language. $Info(trm.x_1,exp_1)$ will mean that the object-language expression

exp_1 represents the information written on the referent of $trm.x_1$. exp_1 is a meta-language variable that ranges over all well-formed object-language expressions, both terms and formulas. It is important to realize that even though Info can take terms as arguments, it is not an ordinary predicate. For example, if Father(John) is a term denoting the father of John, then Info($Paper_1$,Father(John)) means that $Paper_1$ has written on it *some expression* (presumably in natural language) whose meaning is represented by the formal expression Father(John). If Info were an ordinary predicate, Info($Paper_1$,Father(John)) would have to assert some relation between the piece of paper and John's father, that is, between the denotations of the two argument expressions. Here, however, we have a relation between the denotation of the first argument expression and the second argument expression itself (*not* its denotation). It might be more intuitive to quote the second argument (e.g. Info($Paper_1$,Quote(Father(John)))), but we will want to quantify into the quoted context, and quotation is usually interpreted as blocking such quantifications.

One of the advantages of working with both a meta-language and an object language is that we can introduce an operator like Info whose semantics are unlike anything we have seen before, and we can define those semantics right in the meta-language. This is done in INF1. Notice that on the right side of INF1 the first argument of Info is evaluated, but the second is not. Another unusual feature of INF1 is that :Info, the meta-language correlate of Info, is the sort of expression usually associated with an object-language term. We might have expected the right side of to be $H(w_1,$:Info$(D(w_1,trm.x_1),exp_1))$. Instead, we let $V(w_1,$:Info$(D(w_1,trm.x_1))$ be a meta-language term which denotes the information contained in the referent of $trm.x_1$ in w_1. Info is treated this way because we want to imply that exp_1 represents all the information contained in x_1. :Info(x_1,p_1), however, would not do this, so extra axioms would be required. On the other hand, we don't want to have Info as an

object-language function, because its referent would itself be an object-language expression. We do not want to have object-language expressions as individuals in the object language, because that might allow the introduction of self-referential statements, leading to the familiar semantic paradoxes (e.g. statements like "This statement is false.").

RD1 says that a_1 can read x_1 if and only if a_1 can read, and a_1 is at the same place as x_1. RDS1 says that if a_1 can read, he knows that he can read. This fact is necessary for an agent to be able to reason about how he can acquire knowledge. RD2 tells how reading something true affects the knowledge of the reader. It says that if w_2 is the result of a_1 reading x_1 in w_1, then the worlds which are possible according to what a_1 knows in w_2 are exactly those worlds which satisfy both of the following conditions: First, they must agree with w_2 as to what information is contained in x_1. Second, they must be the result of a_1 reading x_1 in some world which is possible according to what a_1 knows in w_1. Informally, this means that after reading x_1, a_1 knows what information is contained in x_1, and he knows that he has read x_1.

We can use these axioms to show that if John has a piece of paper which he knows has the combination of the safe Sf_1 written on it, he can open the safe by reading the combination from the piece of paper and then dialing it on the safe. The premises are as follows:

Given: True(Safe(Sf_1))
True(At(John,Sf_1))
True(At(John,Ppr_1))
True(Reads(John))
True(Know(John,Exist(?X_1,And(Eq(?X_1,Comb(Sf_1)),Info(Ppr_1,?X_1)))))

The meanings of the first four premises should all be obvious. These are the conditions which ensure that John can physically perform the actions required. The last premise is

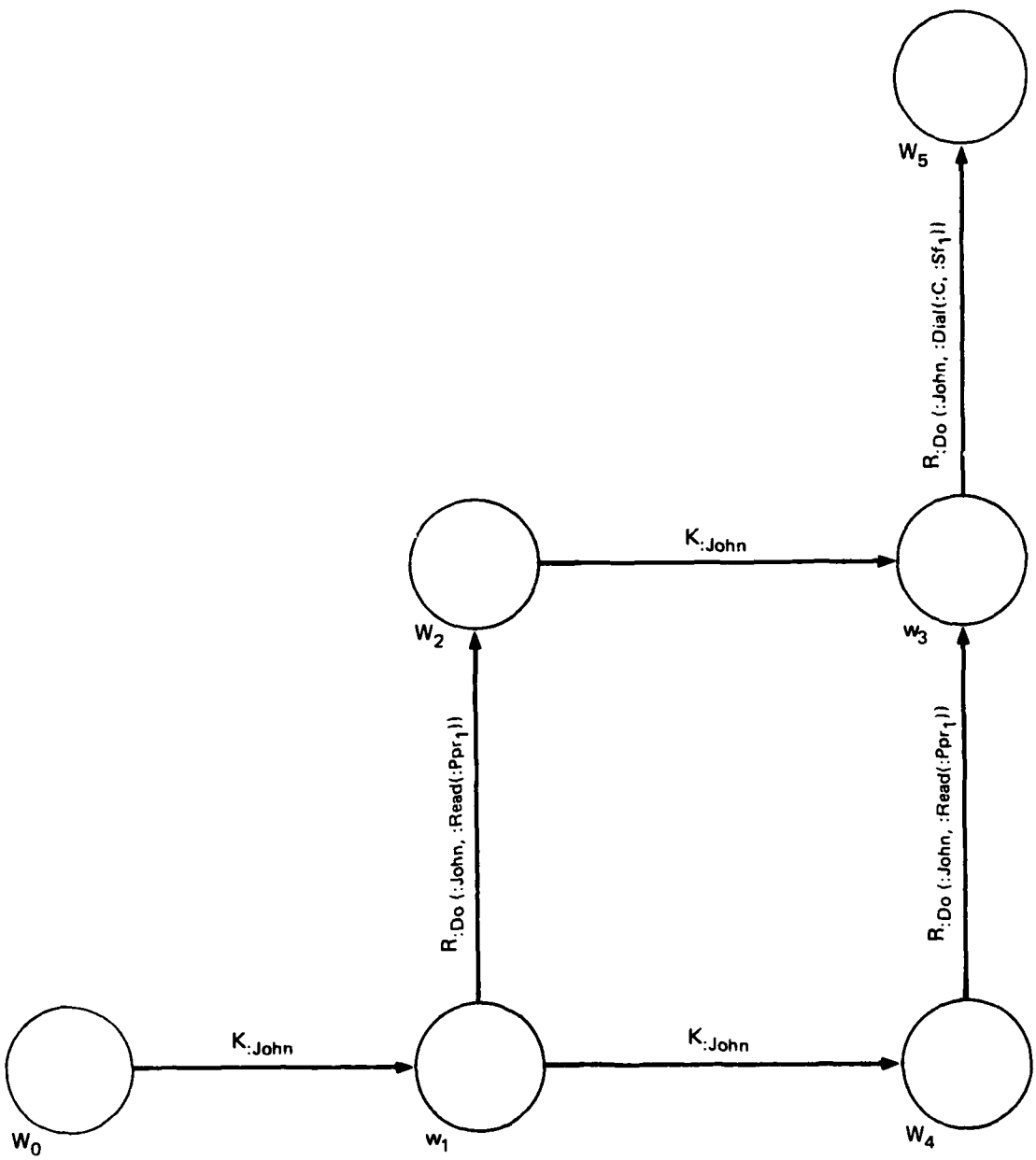


Figure 5.5 Acquiring knowledge required for an action.

the really interesting one. It says that John knows that the combination of the safe is written on the piece of paper Ppr_1. The interpretation of quantifiers in terms of substituting rigid designators enables us to make sense out of quantifying into the second argument position of Info. Technically, this violates our promise not to use the substitutional analysis of quantification in any way that goes beyond the analysis in terms of closures. In this case, however, the substitutional analysis is exactly what we want since there must be some linguistic expression written on the paper. If we were using the closure approach, we would need a special axiom for this case. In particular, we want to infer that it is the standard name of the combination of the safe that is written on the paper. We can describe this as $@(V(W_0,:Comb(:Sf_1)))$, which is equivalent to the expression we get by eliminating the quantifier (See lines 5 and 6).

The possible-world structure for the proof for this example is pictured in figure 5.5. This example provides an execellent illustration of the power of the possible-world approach to reasoning about knowledge and action. The possible worlds mentioned in the proof are related in a fairly complicated way by instances of K and R, and the whole pattern of interconnection is needed to produce the desired conclusion.

Prove: $True(Can(John,(Read(Ppr_1); Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))$

1. $:Safe(Sf_1)$	Given,L1,L9b,L11b
2. $H(W_0,:At(:John,:Sf_1))$	Given,L1,L9a,L11b
3. $H(W_0,:At(:John,:Ppr_1))$	Given,L1,L9a,L11b
4. $H(W_0,:Reads(:John))$	Given,L1,L9a,L11b
5. $K(:John,W_0,w_1) \supset \exists x_1((x_1 = V(w_1,:Comb(:Sf_1))) \wedge (@(x_1) = V(w_1,:Info(:Ppr_1))))$	Given,L1,K1,L11b,L7, L2,L13,L12b,INF1
6. $K(:John,W_0,w_1) \supset (@(V(w_1,:Comb(:Sf_1))) = V(w_1,:Info(:Ppr_1)))$	5
7. $K(:John,W_0,w_1) \supset H(w_1,:At(:John,:Sf_1))$	2,A1
8. $K(:John,W_0,w_1) \supset H(w_1,:At(:John,:Ppr_1))$	3,A1
9. $K(:John,W_0,w_1) \supset H(w_1,:Reads(:John))$	4,RDS1

Lines 1 - 5 translate the premises into the meta-language. Line 6 is a restatement of line

5. From the premises we conclude that John knows that he is at the same place as Sf_1 and Ppr_1 (lines 7 - 8), and that he knows that he can read (line 9).

10.	$K(:John,W_0,w_1)$	Ass
11.	$@(V(w_1,:Comb(:Sf_1))) = V(w_1,:Info(:Ppr_1))$	6,10
12.	$H(w_1,:At(:John,:Sf_1))$	7,10
13.	$H(w_1,:At(:John,:Ppr_1))$	8,10
14.	$H(w_1,:Reads(:John))$	9,10
15.	$R(:Do(:John,:Read(:Ppr_1)),w_1,W_2)$	13,14,RD1
16.	$@(V(W_2,:Comb(:Sf_1))) = V(W_2,:Info(:Ppr_1))$	11,15,D4
17.	$H(W_2,:At(:John,:Sf_1))$	12,15,D4
18.	$K(:John,W_2,w_3) \supset H(w_3,:At(:John,:Sf_1))$	17,A1
19.	$K(:John,W_2,w_3) \equiv ((V(w_2,:Info(Ppr_1)) = V(w_3,:Info(:Ppr_1))) \wedge \exists w_4(K(:John,w_1,w_4) \wedge R(:Do(:John,:Read(:Ppr_1)),w_4,w_3)))$	15,RD2

We let w_1 be a typical world which is possible according to what John knows in W_0 (line 10). In w_1, then, the information written on Ppr_1 is the standard description of the combination of the safe (line 11). John is at the same place as the safe and the piece of paper (lines 12 - 13), and John can read (line 14). Since John can read and he is at the same place as the piece of paper, John can read the piece of paper, resulting in situation W_2 (line 15). Since reading the piece of paper doesn't change either what is on the paper or the combination of the safe, the information written on the piece of paper in W_2 is the standard description of the combination of the safe in W_2 (line 16). Also, John is still at the same place as the safe in W_2, and he knows that this is true (lines 17 - 18). After reading the piece of paper, John knows what is on the piece of paper, and that he has read it (line 19).

20.	$K(:John,W_2,w_3)$	Ass
21.	$H(w_3,:At(:John,:Sf_1))$	18,20
22.	$V(W_2,:Info(Ppr_1)) = V(w_3,:Info(:Ppr_1))$	19,20
23.	$\exists w_4(K(:John,w_1,w_4) \wedge R(:Do(:John,:Read(:Ppr_1)),w_4,w_3))$	19,20
24.	$K(:John,w_1,W_4)$	23
25.	$K(:John,W_0,W_4)$	10,24,K3

26.	$@(V(W_4,:Comb(:Sf_1))) = V(W_4,:Info(:Ppr_1))$	6,25
27.	$R(:Do(:John,:Read(:Ppr_1)),W_4,w_3)$	23
28.	$@(V(w_3,:Comb(:Sf_1))) = V(w_3,:Info(:Ppr_1))$	26,27,D4
29.	$@(V(w_3,:Comb(:Sf_1))) = V(W_2,:Info(:Ppr_1))$	22,28
30.	$@(V(W_2,:Comb(:Sf_1))) = @(V(w_3,:Comb(:Sf_1)))$	16,29
31.	$V(W_2,:Comb(:Sf_1)) = V(w_3,:Comb(:Sf_1))$	30,L10
32.	$:Dial(:V(W_2,:Comb(:Sf_1)),:Sf_1) = :Dial(:V(w_3,:Comb(:Sf_1)),:Sf_1)$	31
33.	$D(W_2,Dial(Comb(Sf_1),Sf_1)) = D(w_3,Dial(Comb(Sf_1),Sf_1))$	32,L11b,L12a,L12b
34.	$D(w_3,@(D(W_2,Dial(Comb(Sf_1),Sf_1)))) = D(w_3,Dial(Comb(Sf_1),Sf_1))$	33,L10
35.	$T(w_3,Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)))$	34,L13

We let w_3 be a typical world which is possible according to what John knows in W_2 (line 20). We conclude that in w_3 John is at the same place as the safe (line 21) and the piece of paper has the same information on it as it does in W_2 (line 22), and that w_3 is the result of John reading the piece of paper in some world which is possible according to what John knows in w_1 (line 23). We call this world W_4 (line 24). Since in W_0 John knows whether he knows something, W_4 is also possible according to what John knows in W_0 (line 25). From this we conclude that the information written on the piece of paper in W_4 is the standard description of the combination of the safe (line 26). Since w_3 is the result of John reading the piece of paper in W_4 (line 27), the information written on the piece of paper in w_3 is the standard description of the combination of the safe (line 28). Since what is written on the piece of paper in w_3 is the same as in W_2, and in both cases that is the standard description of the of the combination of the safe in that world, the two descriptions must be the same (lines 29 - 30), so the combination of the safe must be the same in w_3 as it is in W_2 (line 31). This allows us to conclude that dialing the combination of the safe in w_3 is the same action as dialing the combination of the safe in W_2 (lines 32 - 35).

36. $R(:Do(:John,:Dial(V(w_3,:Comb(:Sf_1)),:Sf_1)),w_3,W_5)$ 1,21,D1
37. $H(W_5,:Open(:Sf_1))$ 36,D2
38. $T(W_5,Open(Sf_1))$ 37,L11b,L9a
39. $R(:Do(D(W_2,John),:Dial(V(w_3,:Comb(:Sf_1)),:Sf_1)),w_3,W_5)$ 36,L11b
40. $R(:Do(D(w_3,@(D(W_2,John))),:Dial(V(w_3,:Comb(:Sf_1)),:Sf_1)),w_3,W_5)$ 39,L10
41. $R(D(w_3,Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1))),w_3,W_5)$ 40,L11b,L12a,L12b
42. $T(w_3,Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))$ 38,41,R1
43. $T(w_3,And(Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)),$ 35,42,L2
$Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))$

Since in w_3 John is at the same place as the safe, it is possible for him to dial the combination of the safe, and we call the resulting situation W_5 (line 36). Since the combination John dials is the combination of the safe, the safe is open in W_5 (lines 37 - 38). Hence, it is physically possible in W_3 for John to open the safe by dialing the combination (lines 39 - 42). Line 43 conjoins this fact with the previous conclusion that dialing the combination of the safe is the same action in w_3 as it is in W_2.

44. $K(:John,W_2,w_3) \supset$ Dis(20,43)
$T(w_3,And(Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)),$
$Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))$
45. $K(D(W_2,@(D(W_0,John))),W_2,w_3) \supset$ 44,L11b,L10
$T(w_3,And(Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)),$
$Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))$
46. $T(W_2,Know(@(D(W_0,John))$ 45,K1
$T(w_3,And(Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)),$
$Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))$
47. $T(W_2,Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))$ 46,C1

Since w_3 is an arbitrarily chosen world which is possible according to what John knows in W_2, we conclude that in W_2 John knows what action dialing the combination of the safe is, and he knows that dialing the combination of the safe will result in the safe being open (lines 44 - 46). So in W_2 John can open the safe by dialing the combination of the safe (line 47).

48. $D(W_0,Read(Ppr_1)) = D(w_1,Read(Ppr_1))$ — L11b,L12b
49. $D(w_1,@(D(W_0,Read(Ppr_1)))) = D(w_1,Read(Ppr_1))$ — 48,L10
50. $T(w_1,Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1))$ — 49,L13
51. $R(:Do(D(w_1,@(D(W_0,John))),:Read(:Ppr_1)),w_1,W_2)$ — 15,L11b,L10
52. $R(D(w_1,Do(@(D(W_0,John)),Read(Ppr_1))),w_1,W_2)$ — 51,L11b,L12b
53. $T(w_1,Res(Do(@(D(W_0,John)),Read(Ppr_1)),$ $Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1))))$ — 52,47,R1
54. $T(w_1,And(Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1)),$ $Res(Do(@(D(W_0,John)),Read(Ppr_1)),$ $Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))))$ — 50,53,L2

By making Ppr_1 a rigid designator, we imply that John knows what object has the combination of the safe written on it, so reading the the piece of paper in w_1 is the same action as reading the piece of paper in W_0 (lines 48 - 50). We already know that in W_2 John can open the safe by dialing the combination and that W_2 is the result of reading the piece of paper in w_1, so in the result of reading the piece of paper in w_1, John can open the safe by dialing the combination (lines 51 - 53). Line 54 conjoins this fact with the previous conclusion that reading the piece of paper in w_1 is the same action as reading the piece of paper in W_0.

55. $K(:John,W_0,w_1) \supset T(w_1,And(Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1)),$ $Res(Do(@(D(W_0,John)),Read(Ppr_1)),$ $Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))))$ — Dis(10,54)
56. $T(W_0,Know(John,And(Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1)),$ $Res(Do(@(D(W_0,John)),Read(Ppr_1)),$ $Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1))))))$ — 55,L11b,K1
57. $T(W_0,Can(John,Read(Ppr_1),$ $Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1))))$ — 56,C1
58. $T(W_0,Can(John,(Read(Ppr_1); Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))$ — 57,C2
59. $True(Can(John,(Read(Ppr_1); Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))$ — 58,L1

Since w_1 is an arbitrarily chosen world which is possible according to what John knows in W_0, we conclude that in W_0 John knows what action reading the piece of paper is, and

he knows that reading the piece of paper will result in a situation where he can open the safe by dialing the combination (lines 55 - 56). So in W_0, by reading the piece of paper John can bring about a situation where he can open the safe by dialing the combination (line 57). Finally, John can open the safe by first reading the piece of paper and then dialing the combination of the safe (lines 58 - 59).

This section concludes the discussion of purely representational and logical issues. We have presented a formalism that allows us to represent and reason with information about what someone knows, information about the effects of actions, and information about the interactions between the two. While we have used axioms describing the properties of particular actions and predicates, much of the power of the system comes from its ability to make use of general principles about knowledge and action. In the rest of the thesis we will examine the problems involved in designing procedures to do this reasoning automatically.

6. Automating Deductions about Knowledge

6.1 Procedural Deduction and First-order Logic

In this chapter, we will discuss the problem of how to algorithmically generate a deduction of a desired conclusion from a group of facts involving knowledge and action. We will begin by looking at some general considerations in the area of automatic deduction. This subject was reviewed in detail in Moore (1975), and on many points the reader may refer to this source for further discussion.

Before going further, we should consider the possibility that we have made an insurmountable mistake by choosing first-order logic as the basis of our representation. It is often argued in the AI literature (Minsky, 1974) (Hewitt, 1975) (Smith, 1977) that there is something fundamentally wrong with formal logic (and its semantics) as a representation of knowledge, and that many of the problems of creating reasoning programs are more easily handled by using Frames, Actors, or some other representation scheme instead.

It is certainly true that traditional logic is limited in many ways. Notions of plausible inference and retracting conclusions in the face of better evidence do not fit comfortably into the framework of model-theoretic semantics, as we have already seen. However, I believe that these deficiencies must be remedied by extending logic, not replacing it. Any system adequate for representing the knowledge of an intelligent being must surely be able to:

(1) Say that something has a certain property without saying which thing has that property.

(2) Say that everything in a certain class has a certain property without saying what everything in that class is.

(3) Say that at least one of two statements is true without saying which statement is true.

(4) Explicitly say that a statement is false.

(5) Either settle or leave open to doubt whether two non-identical expressions name the same object.

Any representation scheme that has all these abilities will have at least a subset which is isomorphic to first-order logic, and for which model-theoretic semantics will be an acceptable, if not total, explanation. Furthermore, it is precisely the difficult problems of reasoning with quantifiers, disjunction, and equality that have not been dealt with adequately by any of the proposed alternatives to logic.

It is important to note that in reasoning about knowledge and action we have to face most of the problems of reasoning in first-order logic. Often AI systems avoid this by embodying simplifying assumptions about the logical structure of the system's knowledge. The most frequent such assumption is that the system has a complete description of the problem domain and the problem situation. The blocks world reasoning component of Winograd's (1971) SHRDLU is the paradigm example of this type of system. In SHRDLU, questions of the form $\exists x_1(P(x_1))$ or $\forall x_1(P(x_1))$ are answered by checking whether the system knows of an object that satisfies P, or whether every object the system knows about satisfies P; any two terms are assumed to represent different objects unless they can be evaluated to the same expression; and any statement which is cannot be shown to be true is assumed to be false. In chapter 1 of Moore (1975) it is shown how these assumptions are virtually built into PLANNER and related AI problem-solving languages.

We cannot make any of these assumptions, however. The examples we looked at in the preceding chapters make use of the full logical power of our formalism. In particular, we depend crucially on being able to quantify over an infinite set of possible worlds, we reason explicitly about the equality of terms, and we seek positive evidence for inferring statements to be false.

These observations narrow the range of possibilities open to us. One extreme approach would be to devise an ad hoc set of inference procedures for exactly the inferences we think we will need to make. This approach, although it sometimes produces impressive performance, has serious problems. There is no reason to assume that the techniques used will generalize to other problem domains, or even to other problems in the same domain. Moreover, as the system is expanded to handle more and more situations, it can bog down in searching for rules or procedures that apply to the particular situation at hand.

The other extreme would be to use one of the uniform proof procedures for first-order logic and the axioms described in chapters 4 and 5. Experience has shown, however, that for even a moderate number of axioms this approach can be very inefficient, especially when many of the axioms are not needed for the solution of the problem. A number of possible reasons for this are discussed in Moore (1975).

We will try to steer a middle course between these two extremes. We will describe an approach which uses a general proof procedure, but which augments that procedure with domain-specific knowledge of how certain facts are to be used. We will call systems that take this approach *procedural deduction systems*.

In taking this type of approach, we have to decide what constitutes "cheating". In an ad hoc approach, nothing is considered to be cheating, but this leaves open the possibility that the answers to specific problems are directly built into supposedly general problem-solving techniques. In a strict approach based on a uniform proof procedure, including anything other than first-order axioms is considered cheating.

Neither of these attitudes seems to be quite right. We want to design programs which are expert at reasoning about knowledge and action. Certainly part of that expertise is knowing how and when to use a given fact, but this kind of knowledge is not accessible to a uniform proof procedure. At the same time, we don't want to require that the system have

special knowledge in order to solve specific problems; we seek generality at least across the problem domain. Therefore, problem-specific assertions and goals will be represented as expressions in pure first-order logic, and specific control information will be provided only for facts which have domain-wide applicability.

6.2 Outline of a Procedural Deduction System

The procedural deduction system we will use can be characterized as being a natural-deduction system which uses both backward and forward chaining, handles quantifiers by means of Skolemization and unification, and has some limited ability to handle equality. We will begin by describing how the system works given a complex goal and data-base of simple assertions. We will then go on to describe how complex assertions are used as either backward-chaining or forward-chaining rules of inference.

The most basic operation in any deduction system is matching. An assertion satisfies a goal just in case they match. For our matching routine we will use the unification procedure from resolution theorem proving. (See Chang and Lee (1973) for a review of this field.) Two expressions match if and only if there is a substitution for the variables in each expression which makes the expressions identical. For instance, the goal $P(x_1,A)$ matches the assertion $P(B,x_2)$ because they can be made identical by substituting B for x_1 in the goal and A for x_2 in the assertion. It should be obvious that the substitution must be uniform within each expression. That is, every instance of a particular variable in one of the expressions must receive the same value. If the two expressions being unified contain the same variables, it may be necessary to change the variables in one of them to avoid confusion in specifying the substitution.

Frequently, we will need to know what substitution was used to unify the expressions.

When that is the case, and when there is more that one possible unifying substitution, we will pick the most general (i.e. least restricting). This is called the most general unifier. For example, $P(x_1,x_2)$ and $P(x_3,A)$ can be unified by substituting A for every variable, but it would be less restrictive to substitute x_3 for x_1 and A for x_2. The most general unifier is guaranteed to be unique down to the choice of variable names.

Quantifiers are handled by Skolemization. Whenever a formula of the form $\forall x(P(x))$ is asserted, it will be replaced by the formula P(x). Whenever a formula of the form $\exists x(P(x,y_1,...,y_n)$ is asserted, where $y_1,...,y_n$ are the only free variables in the formula, we replace the formula by $P(F(y_1,...,y_n),y_1,...,y_n)$ where F is a newly created function symbol. If there are no free variables in the formula, $\exists x(P(x))$ becomes P(F) where F is a newly created constant symbol. These two cases can be combined if we think of constants as being functions of no arguments.

The function symbol F is called a Skolem function. The intuitive idea behind the introduction of Skolem functions is that if we know that some object satisfies the formula P(x), we can give that object a name such as F. If that name is not used anywhere else in the system, then we are in no danger of proving anything from P(F) that we couldn't have proved from $\exists x(P(x))$. If there are free variables in the formula, F must be a function of those variables to allow for the possibility that for each assignment of values to those variables, there is a different object which makes P(x) true.

For quantifiers in goals, the process is reversed. Any goal of the form $\exists x(P(x))$ will be replaced by the goal P(x). A goal of the form $\forall x(P(x,y_1,...,y_n)$, where $y_1,...,y_n$ are the only free variables in the goal, will be replaced by the goal $P(F(y_1,...,y_n),y_1,...,y_n)$, where F is a newly created function symbol. The intuitive basis for replacing universal quantifiers by Skolem functions in goals is that if we can prove that the arbitrarily selected object named by the Skolem function satisfies P(x), then everything must satisfy P(x).

To see how Skolemization interacts with unification, consider how we can prove $\forall x(\exists y(P(x,y)))$ from $\exists u(\forall v(P(v,u)))$. By Skolemization, the goal gets converted into $P(F,y)$, and the assertion gets converted into $P(v,G)$, where F and G are newly created Skolem constants. The goal and the assertion can be made to match by substituting F for v and G for y. Notice that we cannot make the converse inference, which would be invalid. That is, we cannot *infer* $\exists u(\forall v(P(v,u)))$ directly from $\forall x(\exists y(Px,y)))$. In this case, the goal is converted into $P(G(u),u)$, and the assertion is converted into $P(x,F(x))$. If we try to unify these two formulas, we have to substitute G(u) for x which makes F(x) into F(G(u)). But this expression then has to be unified with u, and there is obviously no substitution for u which will make it identical with F(G(u)), so the match fails.

We will elaborate this system slightly to handle the typed variables, functions, and constants introduced into our logical formalism in chapters 4 and 5. We will restrict unification so that two expressions match only if they are of the same type. Furthermore, Skolem functions and constants will always be of the same type as the variables they replace.

This method of handling quantifiers is essentially the same as is used in resolution theorem provers, except that we treat goals directly, rather than doing proof by contradiction. Unification of Skolemized formulas is known to be a logically complete treatment of quantifiers (see Chang and Lee (1973)). In our treatment of propositional connectives, however, we will give up completeness in order to simplify our approach.

In proving complex goals built up using propositional connectives, we will use an elaboration of the standard And/Or-tree approach. A goal of the form $(P \vee Q)$ will be replaced by two independent goals P and Q. If either of these goals is satisfied, then the original goal is satisfied. Conjunctive goals are more complicated. If we have $(P(x) \wedge Q(x))$ as a goal, we not only have to prove P(x) and prove Q(x), but we have to make sure that x

receives the same value in each proof. That is, if we prove P(x) by matching the assertion P(A), we have to then prove Q(A) in order to satisfy the original goal. So our procedure for handling conjunctive goals is as follows: To prove ($P_1 \wedge ... \wedge P_n$), first prove P_1, then prove each P_i, $i > 1$, in order using the bindings for free variables obtained from the proof of P_{i-1}. The original goal is satisfied if and only if all of the subgoals are satisfied. This method of solving conjunctive goals is called *splitting*.

We will also allow implications and biconditionals to occur as goals, and these will be proved using natural deduction. We will have several different ways of writing implications and biconditionals, corresponding to their different procedural interpretations as assertions, but all variants will be treated the same when they occur as goals. A biconditional goal, e.g. (P <-> Q) will be replaced by an equivalent conjunction of implications, ((P -> Q) ∧ (Q -> P)). This new goal will then be attacked using splitting. An implication, such as (P -> Q) will be proved by asserting the antecedent P and proving the consequent Q using any assertions derived from P and previously known facts. The proof of (P -> Q) succeeds if the proof of Q succeeds or if asserting P generates a contradiction. If (P -> Q) is being proved as a subgoal of a branch of a split, then the assertion of P is local to that branch of the split. The data base of assertions must be returned to its former state before the next branch of the split is attacked. If we are trying to prove ((P -> Q) ∧ R) we first assert P and try to prove Q. Then we must remove the effects of asserting P before trying to prove R, or else we will be proving only the weaker condition (P -> (Q ∧ R)).

Finally, we will treat negations in both goals and assertions by pushing them down to the atomic level. We will replace ¬(P ∨ Q) by (¬P ∧ ¬Q), ¬(P ∧ Q) by (¬P ∨ ¬Q), ¬(P -> Q) by (P ∧ ¬Q), ¬(P <-> Q) by ((P ∧ ¬Q) ∨ (Q ∧ ¬P)), and ¬(¬P) by P. At the atomic level, negation will be handled by the matcher. So just as P(x) matches P(A), ¬P(x) matches ¬P(A).

We will augment these methods with some simplification and deletion procedures. The

simplification rules are based on recognizing certain contradictory or tautologous subexpressions in goals and assertions. The easiest way to state these rules is to introduce the special proposition symbols T for true and F for false. The simplification rules are as follows: Replace any conjunction containing both a formula and its negation by F. Replace any disjunction containing both a formula and its negation by T. Then replace (T ∨ P) by T, (T ∧ P) by P, (T -> P) by P, (P -> T) by T, (T <-> P) or (? <-> T) by P, and ¬T by F. Replace (F ∧ P) by F, (F ∨ P) by P, (F -> P) by T, (P -> F) by ¬P, (F <-> P) or (P <-> F) by ¬P, and ¬F by T.

If an entire assertion simplifies to T, it is a tautology which can be discarded. If an assertion simplifies to F and it occurs as the result of asserting the antecedent of an implication we are trying to prove, then the implication is proved, otherwise it indicates that the premises of the problem are inconsistent. If a goal simplifies to T then the goal has been solved. If a goal simplifies to F then it is self-contradictory and should be abandoned.

The other deletion rules that we will use are to delete repeated instances of assertions and goals, and goals that are contradicted by a single assertion. We could use more elaborate techniques of this type based on the subsumption procedure used in resolution systems, but these simple methods are sufficient for our examples.

6.3 Procedural Interpretation of Complex Assertions

Complex assertions will be broken down less than complex goals. We have already explained the handling of quantifiers and negation in complex assertions. Conjunction will be handled quite simply by replacing any assertion of the form (P ∧ Q) by the two independent assertions P and Q. This leaves us with disjunctions, implications, and biconditionals still to treat. We will use these logical expressions as domain-specific inference rules. By specifying control information in these rules, we justify calling our deductive system procedural.

There are two types of control information that we will use. The first of these derives from the original work on PLANNER, where Hewitt (1972) pointed out that an assertion of the form $(P \supset Q)$ has two very natural interpretations as an inference procedure; either assert Q whenever P is asserted, or in order to prove Q, try to prove P. The first of these two methods is usually called forward chaining, and the second, backward chaining; so we will refer to implication assertions used in these ways as forward-chaining or backward-chaining rules, respectively.

The same observation holds for the contrapositive form, $(\neg Q \supset \neg P)$, so there are two more interpretations in the list of possibilities; either assert $\neg P$ whenever $\neg Q$ is asserted, or in order to prove $\neg P$, try to prove $\neg Q$. In many situations choosing a set of procedural interpretations for axioms of the form $(P \supset Q)$ is the most important way of controlling the size of the space that must be searched in making a deduction.

Most deductive systems do all their reasoning by backward chaining from the goal. This is done because unrestricted forward inference will frequently produce large numbers of formulas that have nothing to do with the current goal. This problem would be especially severe in a large data-base containing many types of knowledge. Unrestricted backward inference at least produces subgoals which are relevant to the main goal. There are many cases, however, when very large backward-chaining searches can be eliminated by allowing limited forward deduction.

One type of situation where this is true is reasoning about membership in a hierarchically structured set of classes. For instance, we might chose to represent the fact that cats are mammals by the formula $(\text{Cat}(x_1) \supset \text{Mammal}(x_1))$. We would have similar formulas to represent the facts that all dogs are mammals and that all mammals are animals. In the set of formulas defining this hierarchy, a predicate can occur many times on the right side of an implication, but only once on the left.

Suppose we know that Felix is a cat, and we want to deduce something about Felix that requires showing that Felix is an animal. If we interpret the axioms that define the hierarchy as backward-chaining rules, then to show that Felix is an animal, we may have to search through most of the kinds of animals we know about before hitting upon the assertion that Felix is a cat. If, on the other hand, we interpret the axioms as forward-chaining rules, then for each individual we know about we would make a few assertions about what classes in the hierarchy the individual belongs to, but there would be no searching at all on goals. So at the cost of a few assertions per individual, we can entirely avoid searching a potentially very large space.

Another case where forward deduction is desirable is where inferences can form a chain which is finite in the forward direction, but infinite in the backward direction. For instance one of the axioms of number theory is that if the successor of x_1 is less than x_2, then x_1 is less than x_2: $((S(x_1) < x_2) \supset (x_1 < x_2))$. If we interpret this axiom as a backward-chaining rule, it will generate infinitely many subgoals whenever it is invoked. The goal (A < B) will generate the subgoal (S(A) < B)), which will in turn generate the subgoal (S(S(A)) < B), etc. If we interpret the axiom as a forward-chaining rule, however, the number of assertions generated will be limited by the depth of nesting of S's in the original assertion. The assertion (S(S(A)) < B) will generate the assertion (S(A) < B), which will generate the assertion (A < B) and then stop.

Other cases where forward deduction is useful include expanding defined terms by their definitions, putting a problem description into canonical form, or making a change of representation for a problem. This last case would include our translating from the modal representation of facts about knowledge and action to the possible-world representation.

It is frequently the case that if there is a strong argument for interpreting an implication $(P \supset Q)$ as a forward-chaining or backward-chaining rule there is an equally strong dual

argument for interpreting the contrapositive form (¬Q ⊃ ¬P) in the opposite way. For instance, if we know that some individual is not an animal, we would not want to have to assert all the different kinds of animals that it is not. Moreover, to prove by backward chaining that this individual is not a cat would require checking only the few classes in the hierarchy above cat. So the formula (¬Mammal(x_1) ⊃ ¬Cat(x_1)) should definitely be interpreted as a backward-chaining rule. Similarly, a moment's thought will show that (¬(x_1 < x_2) ⊃ ¬(S(x_1) < x_2)) should also be interpreted as a backward-chaining rule.

We will use different notations to specify different combinations of possible procedural interpretations of complex assertions:

1. (P -> Q): If P is ever asserted, also assert Q.
2. (Q <- P): In order to prove Q, try to prove P.
3. (P => Q): If P is ever asserted, also assert Q, and in order to prove ¬P, try to prove ¬Q.
4. (Q <= P): In order to prove Q, try to prove P, and if ¬Q is ever asserted, also assert ¬P.
5. (P ∨ Q): In order to prove P, try to prove ¬Q, and in order to prove Q, try to prove ¬P.

1 - 4 are all different procedural interpretations for (P ⊃ Q). 1 and 2 are simple interpretations as a single forward-chaining or backward-chaining rule. 3 and 4 reflect our observation that frequently if an implication is most efficiently used as a forward-chaining rule, its contrapositive form is most efficiently used as a backward chaining rule, and vice-versa. 5 can also be thought of as a procedural interpretation of implication because (P ∨ Q) is equivalent to (¬P ⊃ Q) and (¬Q ⊃ P). 5 would be useful when there is no particular reason to use forward chaining, so backwards chaining is used in both cases. 5 can be generalized to handle more than two disjuncts as follows:

5'. ($P_1 \vee ... \vee P_n$): *In order to prove* P_i, *try to prove* ($\neg P_1 \wedge ... \wedge \neg P_{i-1} \wedge \neg P_{i+1} \wedge ... \wedge \neg P_n$).

That is, if the goal we wish to prove is one of a number of possibilities, we should try to prove that all the other possibilities are false.

We also have two procedural interpretations of biconditionals:

6. (P <=> Q): In order to prove P, ¬P, Q, or ¬Q, try to prove Q, ¬Q, P, or ¬P, respectively, but do not immediately reapply this rule.

7. (P <=> Q): In all goals and assertions replace any active occurrence of P by Q

6 interprets a (P <=> Q) as a set of backward chaining rules for transforming a goal containing P to a goal containing Q, or vice-versa. Since the rules go both directions, it is useful to restrict (P <=> Q) from being applied twice in a row to prevent regenerating the original goal. 7 is used when we always want to reason in terms of Q rather than P. It not only generates a new formula containing Q, but also eliminates the formula containing P. By an active occurrence of P, we mean an occurrence that is currently a candidate for being matched. This would include the current goal, all atomic assertions, the left hand side of 1 - 4 and 7, and all of 5 and 6. We restrict our attention to active occurrences, so that if we have an assertion like (P => Q) and Q is a very complicated expression, we don't have to go rummaging around in Q looking for possible substitutions until we actually try to use Q.

All of the interpretations of 1 - 7 have been stated in purely propositional terms, but they should be taken to cover cases with variables as well. For example, if we had the goal Q(A) and the assertion (Q(x) <= P(x)), we would generate the goal P(A). The result of applying a rule must of course take into account the substitution that was used to make the match succeed.

Notice that because our matcher works only on atomic expressions, the formulas in active positions in assertions in the forms given in 1 - 7, must be atomic expressions in

order to be used. For instance, we have not specified any way to use an assertion like ((P ∧ Q) => R). This is not as much of a restriction as it seems, however, because formulas can always be re-written or expanded, so that they fit into the patterns we handle. ((P ∧ Q) => R) can be re-written as (P => (Q => R)). We could work out a set of rules for doing this automatically, or we could extend our matching rules to handle more complex assertions, but since all of our examples can be handled by the current rules, we won't bother to do so.

The other type of control information we will want to put into assertions is syntactic restrictions on the use of those assertions. For example, one very concise way to say that John knows whether P is true is $(K(:John,W_0,w_1) \supset (T(w_1,P) \equiv T(W_0,P)))$. That is, any world which is compatible with what John knows in the actual world must agree with the actual world as to whether P is true. A straight-forward way of using a fact of this type would be as a forward-chaining rule: whenever we have an assertion that a world, say W_1, is compatible with what John knows in the actual world W_0, i.e. $K(:John,W_0,W_1)$, we would assert that W_1 agrees with W_0 as to whether P is true.

Recall, however, that since anything that is known by someone must be true, we have axiom K2, $\forall a_1,w_1(K(a_1,w_1,w_1))$, which says that every world is compatible with what anyone knows in that world. Combining this assertion with the rule representing the fact that John knows whether P is true would produce the tautologous conclusion $(T(W_0,P) \equiv T(W_0,P))$.

We could add to our rules for recognizing tautologies a check for this pattern, but a simpler solution to this problem would be to put a syntactic test into the assertion to prevent application of the rule if the expression being bound to w_1 is W_0. The representation of the assertion might then look like:

$(K(:John,W_0,w_1)/[W_0 \neq w_1] \rightarrow (T(w_1,P) \Leftrightarrow T(W_0,P)))$.

The square brackets indicate that $[W_0 \not= w_1]$ is a syntactic test and not a goal to be proved. The test indicated by $\not=$ is satisfied if after the pattern match the arguments of $\not=$ are not unifiable. Another way to achieve the same effect would be to add a piece of advice (as in PLANNER) not to apply this rule to axiom K2. Neither of these restrictions can be expressed as a pure logical formula. The closest we could come would be to write something like:

$$((K(:John,W_0,w_1) \wedge (W_0 \not= w_1)) \supset (T(w_1,P) \equiv T(W_0,P))).$$

This is too strong, however, since it would require us to *prove* that W_0 and w_1 are not the same possible world before applying the the rule. To avoid the problems we discussed above, we only need to do a simple test to see whether they are the same expression. In section 7.1 we will see a more complex example involving the axiom D3, where the use of a syntactic restriction is used to prevent a forward-chaining rule from generating infinitely many assertions.

The use of syntactic restrictions is also helpful in solving a problem relating to our treatment of $(P \Leftrightarrow Q)$. We have interpreted this as a rule to replace all occurrences of P by Q. But what happens if the occurrence of P being replaced is more general than the instance in the replacement rule? For example, suppose we have the assertions $(P(x) \vee R(x))$ and $(P(A) \Leftrightarrow Q(A))$. We can generate the new assertion $(Q(A) \vee R(A))$ by substituting A for x in the first assertion, and then substituting Q(A) for P(A), but if we delete the old assertion, we will lose the information that $(P(x) \vee R(x))$ is true for values of x other than A. We could just leave the old assertion as it is, but this would create an undesirable redundancy. If we came along later with the goal P(y), we would match $(P(x) \vee Q(x))$ and generate the goal $\neg R(y)$. We would also match $(P(A) \Leftrightarrow Q(A))$ and generate the goal P(A) which would in turn match $(Q(A) \vee R(A))$ and generate the goal $\neg Q(A)$. But this is a special case of a goal we

have already generated, and is therefore redundant. In complicated situations this can lead to massive generation of redundant goals.

A solution to this problem is to have the replacement rule change the first assertion to be (P(x) ∨ R(x))/[x ≠ A]. That is, we do effectively delete the particular instance of the assertion that our replacement rule applies to, by putting a syntactic restriction on the assertion not to match that instance. With this procedure, if we have as a goal P(y) we will ultimately generate one goal which is ¬R(A) and another goal which is ¬R(x)/[x ≠ A] these two goals are mutually exclusive as to the patterns they will match, so the redundancy is eliminated.

Except for the equality rules to be discussed in the next section, these are all the inference rules that we will use in our deduction system. As they stand, they are far from forming a logically complete system. The most glaring deficiency is an inability to do reasoning by cases. That is, even if we have asserted (P <= Q), (P <= R), and (Q ∨ R), we cannot deduce P. Our system can be modified in a relatively straightforward way to handle reasoning by cases by changing the treatment of disjunctive assertions to a splitting procedure which is the dual of the one we are using for goals. A system of this type requires a much more complicated control structure than we wish to use, and since none of our sample problems involve reasoning by cases, it did not seem worth the effort to describe. To see what is required for such a system, see Nevins (1974). Nevins's system is quite similar to ours, but he does not impose as much control as we do over the use of complex expressions, and he does not use syntactic restrictions at all.

Alternatively, we could have described a much simpler system closer in spirit to resolution, but conjunctive goals which include implications would not be handled as naturally as in a system based on splitting. This is particularly important in our domain, since every attempt to prove that someone knows something generates a goal of the form

$(K(:A,W_1,W_2) \rightarrow T(W_2,P))$. For a comparison of splitting-based systems to resolution-style systems, see chapter 3 of Moore (1975).

The global control strategy we will use is simply depth-first search. We could put in some heuristics to try to be more intelligent, but they would be unncecessary. The point is that just by the choice of procedural interpretations of our domain-specific axioms and the use of syntactic restrictions, we can constrain the search space for our examples so tightly that the order in which the space is searched does not matter very much.

6.4 Inference Rules for Equality

In this section we give the inference rules for reasoning about equality. The first rules embody the fact that everything is equal to itself:

1. Replace any expression of the form (A = A) by T.
2. Replace any expression of the form (A ≠ A) by F.
3. If (A = B) is a goal where A and B are unifiable, solve the goal by unifying A and B.

The first two rules should require no explanation. The point of actually carrying out the unification in the third rule is that the goal (A = B) may have been generated by splitting a conjunctive goal, and the variable bindings created by the unification may be required by the other conjuncts of the goal which was split.

One practical problem in reasoning about equality in an AI system is that typically there are large numbers of specific individuals that the system knows about and has names for. In the blocks world every block usually has an "internal" name, and in circuit analysis systems every component is usually given a unique identifier. The problem this creates is that to reason using these identifiers, the system needs to know that they refer to distinct individuals. For instance, suppose there are three blocks, A, B, and C, and B is put on C.

To be able to infer that A is still in its original location, the system must not only understand the effects of putting one block on another, it must also know that A and B are not the same block.

In standard logic the only way of indicating that A and B are not the same is to have a specific axiom ($A \neq B$). To avoid cluttering up our system with large numbers of axioms of this form, we can make use of the notion of a standard name for an individual which we introduced in section 2.5. If we assume that each individual has only one standard name, then it follows that two syntactically distinct standard names must name different individuals. We will designate certain constants in our formalism as being standard names, and we will build into our system the assumption that two distinct standard names cannot be equal. This fact gives us the following two rules:

4. If A and B are different standard names, replace $(A = B)$ by F.
5. If A and B are different standard names, replace $(A \neq B)$ by T.

We also have functions that act as constructors of standard names. A standard name constructor is a function symbol such that a term consisting of the function symbol applied to standard names is itself a standard name. For instance, $:Dial(:C_1,:Sf_1)$ is the standard name of the action of dialing the combination named by $:C_1$ on the safe named by $:Sf_1$, just in case $:C_1$ and $:Sf_1$ are themselves standard names.

In an actual implementation we would probably pick some notational convention to distinguish standard names from other terms. In our examples, however, we will not use any special notation, but will simply point out when we are assuming that an expression is a standard name. We will note, however, that the meta-language terms that we use to denote object-language expressions will be regarded as standard names of those expressions. Also, the ":" terms that denote intensional objects will be the standard names of those objects.

Since there are several levels to deal with, we must be careful which level we are in. For example, $Comb(Sf_1)$ is the standard name of the object-language expression which means "the combination of Sf_1". That is, we know implicitly that $Comb(Sf_1) \neq Comb(Sf_2)$, because the two terms denote different object-language expressions. $:Comb(:Sf_1)$ is the standard name of the intentional object corresponding to the combination of the safe named by $:Sf_1$, just in case Sf_1 is the standard name of the safe. $V(W_0,:Comb(:Sf_1))$ refers to the actual combination of the safe, so it can't be a standard name since it might be the case that $V(W_0,:Comb(:Sf_1)) = V(W_0,:Comb(:Sf_2))$. That is, two different safes can have the same combination.

There are four special rules for standard name constructors:

6. If F and G are different standard name constructors, replace all expressions of the form $F(A_1,...,A_n) = G(B_1,...,B_n)$ by F.

7. If F and G are different standard name constructors, replace all expressions of the form $F(A_1,...,A_n) \neq G(B_1,...,B_n)$ by T.

8. If F is a standard name constructor, replace all expressions of the form $F(A_1,...,A_n) = F(B_1,...,B_n)$ by $((A_1 = B_1) \wedge ... \wedge (A_n = B_n))$.

9. If F is a standard name constructor, replace all expressions of the form $F(A_1,...,A_n) \neq F(B_1,...,B_n)$ by $((A_1 \neq B_1) \vee ... \vee (A_n \neq B_n))$.

After all of the previous rules have been applied, the following more general rules are applied:

10. If $(A = B)$ is an assertion, replace all active occurrences of A by B, unless A is a standard name, in which case, replace all active occurrences of B by A.

11. If $(A \neq B)$ is a goal, for each assertion of the form P(A), generate the goal $\neg P(B)$. If that cannot be proved, for each assertion of the form P(B), generate the goal $\neg P(A)$.

12. If $F(A_1,...,A_n) = F(B_1,...,B_n)$ is a goal, generate $(A_1 = B_1) \wedge ... \wedge (A_n = B_n)$ as a goal.

13. If $F(A_1,...,A_n) \neq F(B_1,...,B_n)$ is an assertion, also assert $(A_1 \neq B_1) \vee ... \vee (A_n \neq B_n)$.

Rule 10 is the standard equality substitution rule. It rewrites all expressions involving one of the terms as expressions involving the other. When given a chance, it prefers to state things in terms of standard names, because this cuts down the possibilities for further equality substitutions, and also because there is usually more known about an object under its standard name than under other descriptions. This rule needs the same modification as the replacement rule for ⟨=⟩. If the expression that would be replaced is more general than the replacement rule, the old expression is not deleted, but is instead modified by a syntactic restriction. For instance, if $F(A) = B$ is applied to the formula $P(F(x))$, the result is the two formulas $P(B)$ and $P(F(x))/[x \neq A]$.

Rule 11 is the dual of equality substituition. The idea is that two individuals cannot be the same if they differ in some property. Since this rule is so general it should be tried only as a last resort. It probably could be tightened up, but since no applications of it will be made in our examples, there is little motivation to do so.

As with the rules in the previous section, these rules for equality are incomplete, although they are adequate for our examples. In particular, equalities which are part of a larger assertion, e.g. $((A = B) \vee P)$, and equalities which permute expressions, e.g. $(x_1 \cdot x_2 = x_2 \cdot x_1)$, are not adequately handled. These problems are discussed more fully in chapter 4 of Moore (1975).

6.5 Procedural Interpretation of the Axioms for Knowledge

In this section we will give procedural interpretations to the basic axioms for knowledge. (The procedural versions of all our axioms are listed for reference in appendix B.) For axioms L1 - L13 the procedural interpretations are quite straightforward. Each of these

axioms or schemas specifies an equivalence between an object-language expression and its meta-language interpretation. The procedural interpretation of these axioms will be simply to replace the object-language expression by its meta-language equivalent.

L1. $True(p_1) \Leftrightarrow T(W_0,p_1)$

L2. $T(w_1,(And(p_1,p_2)) \Leftrightarrow (T(w_1,p_1) \wedge T(w_1,p_2))$

L3. $T(w_1,(Or(p_1,p_2))) \Leftrightarrow (T(w_1,p_1) \vee T(w_1,p_2))$

L4a. $T(w_1,(p_1 \rightarrow p_2)) \Leftrightarrow (T(w_1,p_1) \rightarrow T(w_1,p_2))$

L4b. $T(w_1,(p_1 \leftarrow p_2)) \Leftrightarrow (T(w_1,p_1) \leftarrow T(w_1,p_2))$

L4c. $T(w_1,(p_1 \Rightarrow p_2)) \Leftrightarrow (T(w_1,p_1) \Rightarrow T(w_1,p_2))$

L4d. $T(w_1,(p_1 \Leftarrow p_2)) \Leftrightarrow (T(w_1,p_1) \Leftarrow T(w_1,p_2))$

L5a. $T(w_1,(p_1 \Leftrightarrow p_2)) \Leftrightarrow (T(w_1,p_1) \Leftrightarrow T(w_1,p_2))$

L5b. $T(w_1,(p_1 \Leftrightarrow p_2)) \Leftrightarrow (T(w_1,p_1) \Leftrightarrow T(w_1,p_2))$

L6. $T(w_1,Not(p_1)) \Leftrightarrow \neg T(w_1,p_1)$

L7. $T(w_1,Exist(?S_i,P)) \Leftrightarrow \exists s_i(T(w_1,P[@(s_i)/?S_i]))$

L8. $T(w_1,All(?S_i,P)) \Leftrightarrow \forall s_i(T(w_1,P[@(s_i)/?S_i]))$

L9a. $T(w_1,P(trm_1,...,trm_n)) \Leftrightarrow H(w_1,:P(D(w_1,trm_1),...,D(w_1,trm_n)))$
if P is not an essential property of the things it is true of.

L9b. $T(w_1,P(trm_1,...,trm_n))] \Leftrightarrow :P(D(w_1,trm_1),...,D(w_1,trm_n))$
if P is an essential property of the things it is true of.

L10a. $D(w_1,@(x_1)) = x_1$

L10b. $(@(x_1) = @(x_2)) \Leftrightarrow (x_1 = x_2)$

L11a. $D(w_1,Cnst) = V(w_1,:Cnst)$ if Cnst is not a rigid designator.

L11b. $D(w_1,Cnst)] = :Cnst$ if Cnst is a rigid designator.

L12a. $D(w_1,F(trm_1,...,trm_n)) = V(w_1,:F(D(w_1,trm_1),...,D(w_1,trm_n))$
if F is not a rigid function.

L12b. $D(w_1,F(trm_1,...,trm_n)) = :F(D(w_1,trm_1),...,D(w_1,trm_n))$
if F is a rigid function.

L13. $T(w_1,Eq(trm_1,trm_2)) <=> (D(w_1,trm_1) = D(w_1,trm_2))$

These axioms are basically straightforward translation rules, but there are a couple of interesting points. First, since the meta-language now has several forms for implications and biconditionals, we have augmented the object-language to contain these same forms. Although we have used the same symbols in both the object language and the meta-language, the context will always disambiguate their use.

Second, we have introduced L10b as a new simplification rule. It says that if two standard names are the same, the objects which they refer to must also be the same. This is actually a logical consequence of L10a and is not strictly necessary, but it will simplify certain proofs to have it explicitly asserted.

K1. $T(w_1,Know(trm.a_1,p_1)) <=> \forall w_2(K(D(w_1,trm.a_1),w_1,w_2) -> T(w_2,p_1))$

K2. $K(a_1,w_1,w_1)$

K3. $K(a_1,w_1,w_2)/[w_1 \neq w_2] -> (K(a_1,w_2,w_3)/[w_2 \neq w_3] -> K(a_1,w_1,w_3))$

Axiom K2 is also very simple, being an atomic assertion, but K1 and K3 are more complicated. Like L1 - L13, K1 translates from the object language into the meta-language, but the meta-language side contains an implication for which we have to choose a procedural interpretation. The interpretation we have chosen is to assert that everything that John knows in W_1 is true in W_2, for any world W_2 such that $K(A,W_1,W_2)$ is asserted. Furthermore, the implications in K3 have been interpreted in a way that promotes the principle that whenever a formula of the form $K(A,W_1,W_2)$ is true, it should be explicitly asserted.

The reason for these decisions is the fact that otherwise, forward chaining in the context

of someone's knowledge will not work. In section 6.3 we cited several cases where efficient reasoning about ordinary, non-modal concepts requires forward chaining. Suppose $(P \supset Q)$ is such a case. That means that given $(P \supset Q)$ and P as assertions and Q as a goal, we should proceed by reasoning forward from P to Q rather than backwards from Q to P. We might choose to represent $(P \supset Q)$ as $(P \Rightarrow Q)$.

Suppose that this reasoning was embedded in a knowledge context; e.g. $T(W_0,Know(John,(P \Rightarrow Q))))$. Presumably, we still want $(P \Rightarrow Q)$ to function as a forward-chaining rule, so the meta-language expression of John's knowledge would be a forward-chaining rule that asserts $H(W_1,:Q)$ for any world W_1 for which $H(W_1,:P)$ is asserted, provided $K(:John,W_0,W_1)$ is true. Now the meta-language translation of $T(W_0,Know(John,(P \Rightarrow Q))))$ would be $K(:John,W_0,w_1) \rightarrow T(w_1,(P \Rightarrow Q))$. Suppose that $K(:John,W_0,W_1)$ is asserted. This will result in $T(W_1,(P \Rightarrow Q))$ being asserted, which will be transformed by the L rules into $H(W_1,:P) \Rightarrow H(W_2,:Q)$, which is exactly what we want. If we had chosen to represent the meta-language translation of $T(W_0,Know(John,(P \Rightarrow Q)))$ as a backward chaining rule, the formula $H(W_1,:P) \Rightarrow H(W_2,:Q)$ would not have been explicitly asserted, and so, would not function as a forward-chaining rule. Therefore, the right side of K1 needs to be a forward chaining rule. Furthermore, $K(:John,W_0,W_1)$ also had to be explicitly asserted; for it to be merely derivable would not have been enough. Therefore, we will want facts like $K(:John,W_0,W_1)$ to be asserted whenever possible, so rules like K3 will always be expressed in forward-chaining form.

Another point about K1 and K3 is that the procedural interpretation we have chosen for the implications in those axioms ignores the contrapositive form of the axioms. The most natural contrapositive of a forward-chaining rule which triggers on (Ks_1,w_1,w_2) would be a backward-chaining rule for showing $\neg K(s_1,w_1,w_2)$. For instance, K3 could give rise to

a rule which says to prove something of the form $\neg K(a_1,w_1,w_2)$ try proving $K(a_1,w_2,w_3)$ and $\neg K(a_1,w_1,w_3)$. The reason that we ignore the contrapositive form of assertions like K3 or the right hand side of K1 is that we can structure our system so that assertions and goals of the form $\neg K(a_1,w_1,w_2)$ do not occur.

To see this, note that the only axiom which translates from the object language to the meta-language and produces formulas containing the predicate K is K1. If we assert $T(W_1,Know(A,P))$ we obviously do not generate any formulas containing anything of the form $\neg K(a_1,w_1,w_2)$. If we have a goal of the form $T(W_1,Know(A,P))$, we would generate a subgoal of the form $\forall w_2(K(:A,W_1,w_2) \rightarrow T(w_2,P))$. This would be Skolemized to something like $(K(:A,W_1,W_2) \rightarrow T(W_2,P))$, which would be proved by natural deduction, asserting $K(:A,W_1,W_2)$ and deriving $(T(W_2,P)$.

Conversely, asserting $T(W_1,Not(Know(A,P)))$ would result in asserting something like $K(:A,W_1,W_2)$ and $\neg T(W_2,P)$, and trying to show $T(W_1,Not(Know(A,P)))$ would generate the subgoal $(K(:A,W_1,w_2) \wedge \neg T(w_2,P))$.

In neither case is anything of the form $\neg K(a_1,w_1,w_2)$ generated. So if all problem descriptions are stated in the object language, they will not create anything of the form $\neg K(a_1,w_1,w_2)$. Furthermore, it is quite easy to structure the other axioms so that they do not introduce any formulas of that form. So whenever we have a forward-chaining rule that triggers on $K(a_1,w_1,w_2)$, it will not be necessary to have the contrapositive rule. One way of viewing this fact is to note that the meta-language of our formalism is in some ways richer than the object language, but we do not need to make use of all of that richness.

A final point about K3 is that it includes syntactic restrictions on its application. These syntactic restrictions are included because having K2 around makes it important to check whether a rule produces useful information if its input is $K(a_1,w_1,w_1)$. If we did not place

these restrictions on K3, it would combine with K2 to produce the tautologous assertion $(K(a_1,w_1,w_3) \rightarrow K(a_1,w_2,w_3))$. Moreover, even if w_1 and w_2 are distinct, the inner rule of K3 requires w_2 and w_3 to be distinct to avoid asserting $K(a_1,w_1,w_2)$, which would simply repeat the initial pattern which triggered the inference.

6.6 Some Examples

We are now in a position to work out some examples of automatically generated deductions about knowledge. It should be noted that no program has been written to produce these deductions, so our examples are subject to all the possible errors and omisions of hand simulations.

Our "automatically" generated proofs will be produced by applying all applicable inference rules in a depth-first fashion. The order of application of the rules will be to first apply any rules which replace the current expression by another expression, then if no further rules of that type apply, to try any other rule. Within these two groups of rules we will follow the order that they are presented in the text. Rules which involve a second formula will take those formulas in the order they appear in the proof. The point is to show that the search space is so tightly controlled that a fixed search strategy produces satisfactory results.

In respect to the form of proofs, indentations will indicate the tree structure of the proof, with each indented line being directly derived from the most recent line at the next higher level. These proofs will intermix assertions and goals. Goals will be distinguished by being prefixed with a *. Formulas which are deleted as they are generated, whether by application of a deletion rule or by replacement by another formula, will be prefixed with a •. A solved goal will be indicated by *T.

Since the structure of the proof indicates which preceding line each line is immediately

derived from, the justification column gives only the additional lines or axioms used for that step. We will suppress much of the detail of translating from the object language to the meta-language. Whenever two or more consecutive steps involve rules L1 - L13, we will combine them into a single step and simply give L as the justification.

The special justification notations Ante and Cons indicate the assertion antecedent and the goal consequent of an implication being proved by natural deduction. Subgoals generated by splitting a conjunctive goal will be indicated by the notation Split. The notation Eq indicates the application of one of the simplification rules for equality.

One simple example is to show that if A knows that P implies Q then, if A knows that P, then A knows that Q.

Prove: True(Know(A,(P => Q)) => (Know(A,P) => Know(A,Q)))

1.	*True(Know(A,(P => Q)) => (Know(A,P) => Know(A,Q)))	Goal
2.	*T(W_0,Know(A,(P => Q))) ⊃ T(W_0,(Know(A,P) => Know(A,Q)))	L
3.	*T(W_0,Know(A,(P => Q)))	Ante
4.	K(:A,W_0,w_1) -> T(w_1,(P => Q))	K1
5.	*T(W_0,(P => Q))	K2
6.	H(W_0,:P) => T(W_0,Q)	L
7.	*T(W_0,(Know(A,P) => Know(A,Q)))	Cons
8.	*T(W_0,Know(A,P)) => T(W_0,Know(A,Q))	L
9.	*T(W_0,Know(A,P))	Ante
10.	K(:A,W_0,w_1) => T(w_1,P)	K1
11.	*T(W_0,P)	K2
12.	H(W_0,:P)	L
13.	*T(W_0,Q)	6
14.	H(W_0,:Q)	L
15.	*T(W_0,Know(A,Q))	Cons
16.	*K(:A,W_0,W_1) => T(W_1,Q)	K1
17.	K(:A,W_0,W_1)	Ante
18.	K(:A,W_1,w_3)/[W_1 ≠ w_3] -> K(:A,W_0,w_3))	K3
19.	*T(W_1,(P => Q))	4
20.	H(W_1,:P) => T(W_1,Q)	L
21.	*T(W_1,P)	10
22.	H(W_1,:P)	L
23.	*T(W_1,Q)	20

24.	$H(W_1,:Q)$	L
25.	$**T(W_1,Q)$	Cons
26.	$*H(W_1,:Q)$	Cons
27.	$*T$	24

Line 1 is the statement of the problem in the object-language. Line 2 translates this into a meta-language implication to be proved by natural deduction. Line 3 asserts the antecedent, that A knows that P implies Q. Line 4 translates this into meta-language terms, saying that P implies Q in every world which is compatible with what A knows in W_0. We are treating A as a rigid designator to make the formulas simpler, although this does not affect the length of the proof. Since every world is compatible with what anyone knows in that world, P implies Q in W_0 (lines 5 - 6). We now try to prove the consequent of line 2, that if A knows that P, then A knows that Q (line 7). This translates into a meta-language implication (line 8), so the antecedent is asserted (line 9), which translates into the assertion that P is true in every world which is compatible with what A knows in W_0 (line 10). This implies that P is true in W_0 (lines 11 - 12), and hence, that Q is true in W_0 (line 13 - 14).

We now try to prove the consequent of line 8, that A knows that Q (line 15), which translates into the goal of proving that Q is true in every world compatible with what A knows in W_0. The quantifier in this goal is removed by Skolemization, so we try to prove that if W_1 is a typical world which is possible according to what A knows in W_0, then Q is true in W_1 (line 16). To prove this implication we assert the antecedent, that W_1 is possible according to what A knows in W_0 (line 17), which triggers an application of K3 (line 18), and also implies that P implies Q in W_1 (lines 19 - 20), and that P is true in W_1 (lines 21 - 22), hence Q is true in W_1 (lines 23 - 24). We now try to prove the consequent of line 14, that Q is true in W_1 (lines 25 - 26). This immediately succeeds (line 27), completing the proof.

It is worth making a few comments about this proof. First of all, not every formula generated was needed for the proof. In fact, none of the inferences that depended on axioms K2 or K3 were used. These inferences, however, accounted for only seven of the twenty-seven lines generated. Perhaps more significant is the low branching factor of the proof tree. For the non-terminal nodes of the tree (i.e. those formulas which generated at least one other formula) the average number of branches was less than 1.5. This is reflected in the fact that of the twenty-seven formulas generated, fifteen were immediately replaced by other formulas and deleted. These are cases where the knowledge that there is only one reasonable inference to make from a formula is embedded in the rules of inference and axioms of the system.

Finally it should be emphasized that these are the *only* inferences that can be made from the initial problem statement, given the way the axioms are structured. If we modified the problem slightly, so that the final goal were to prove that Q is not true in W_1, there would still only be about thirty formulas generated, even though the proof would fail. This would definitely not be the case if we turned a standard theorem prover loose on the purely logical version of the formalism given in chapter 4. There are many possibilities for infinite search paths through these axioms, such as trying to prove $T(W_1,Q)$ by proving $T(W_1,And(Q,p_1))$ or by the meta-language equivalent of trying to prove that A knows that Q by proving that A knows that he knows that Q. (Both of these approaches obviously recurse infinitely.) If we tried to prove anything that did not follow from the premises of the problem, a typical theorem-proving algorithm would never terminate. It is the care taken in structuring the domain-defining axioms that makes our system tightly controlled.

As a second example, we will show an algorithmically generated proof that if A knows who B is and A knows who C is, then A knows whether B equals C.

Given: True(Exist(?X_1,Know(A,Eq(B,?X_1))))
True(Exist(?X_1,Know(A,Eq(C,?X_1))))

Prove: True(And(Eq(B,C) => Know(A,Eq(B,C))),(Not(Eq(B,C)) => Know(A,Not(Eq(B,C)))))

1.	*True(Exist(?X_1,Know(A,Eq(B,?X_1))))	Given
2.	K(:A,W_0,w_1) -> T(w_1,Eq(B,@(:B'))	L,K1
3.	*T(W_0,Eq(B,@(:B')))	K2
4.	V(W_0,:B) = :B'	L
5.	*True(Exist(?X_1,Know(A,Eq(C,?X_1))))	Given
6.	K(:A,W_0,w_1) -> T(w_1,Eq(C,@(:C'))	L,K1
7.	*T(W_0,Eq(C,@(:C')))	K2
8.	V(W_0,:C) = :C'	L

Lines 1 - 8 give the premises of the problem and the forward deductions made from them. Line 1 is the first premise, and line 2 is its translation into the meta-language. We have combined several applications of L rules and an application of K1 into a single step. The object-language premise says that there is some individual which A knows to be named by B, which translates into the meta-language assertion that there is some individual (represented by the Skolem constant :B') which is the denotation of B in every world which is compatible with what A knows in W_0. Since W_0 itself is one of those worlds, :B' is the denotation of B in W_0 (lines 3 - 4). Lines 5 - 8 make the analogous inferences for the premise that A knows who C is. As in the previous example, we are assuming that A is a rigid designator to simplify the meta-language formulas, without affecting the length of the proof. B and C, of course, must not be rigid designators to avoid trivializing the proof.

9.	**True(And(Eq(B,C) => Know(A,Eq(B,C))), (Not(Eq(B,C)) => Know(A,Not(Eq(B,C)))))	Goal
10.	**T(W_0,(Eq(B,C) => Know(A,Eq(B,C)))) ∧ T(W_0,(Not(Eq(B,C)) => Know(A,Not(Eq(B,C)))))	L
11.	**T(W_0,(Eq(B,C) => Know(A,Eq(B,C))))	Split
12.	**T(W_0,Eq(B,C)) => T(W_0,Know(A,Eq(B,C)))	L
13.	*T(W_0,Eq(B,C))	Ante
14.	*V(W_0,:B) = V(W_0,:C)	L
15.	*:B' = V(W_0,:C)	4

16.	:B' = :C'	8
17.	V(W_0,:B) = :C'	4

Now we try to prove the goal, that if B and C are the same individual, A knows that they are the same individual, and if they are not the same individual, he knows that they are not. Line 9 states the goal in the object language, and line 10 transforms it into a meta-language conjunction to be solved by splitting. Line 11 states the first conjunct of the split, and line 12 converts it into a meta-language implication, to be proved by natural deduction. Line 13 asserts the antecedent, which translates into the meta-language statement that the denotation of B in W_0 is the same as the denotation of C in W_0 (line 14). Since the denotation of B is :B' and the denotation of C is :C', it follows that :B' is the same as :C' (lines 15 - 16). Making this inference causes the the instance of :B' in line 4 to be replaced by :C'. From this point on in this branch of the split, line 4 is deleted.

18.	*T(W_0,Know(A,Eq(B,C)))	Cons
19.	*K(:A,W_0,W_1) => T(W_1,Eq(B,C))	L,K1
20.	K(:A,W_0,W_1)	Ante
21.	K(:A,W_1,w_3)/[W_1 / w_3] -> K(:A,W_0,w_3))	K3
22.	*T(W_1,Eq(B,@(:B')))	2
23.	*V(W_1,:B) = :B'	L
24.	V(W_1,:B) = :C'	16
25.	*T(W_1,Eq(C,@(:C')))	6
26.	V(W_1,:C) = :C'	L
27.	**T(W_1,Eq(B,C))	Conse
28.	**V(W_1,:B) = V(W_1,:C)	L
29.	**:C' = V(W_1,:C)	24
30.	**:C' = :C'	26
31.	*T	Eq

Line 18 makes the consequent of line 12 into the goal of showing that A knows that B and C are the same individual. This translates into the meta-language goal of showing that in every world (represented by the Skolem constant W_1) which is compatible with what A knows in W_0, B and C refer to the same individual (line 19). This is itself an implication to

be attacked using natural deduction, so we assert the antecedent (line 20), which triggers K3 (line 21). The fact that W_1 is one of the worlds which are compatible with what A knows in W_0 triggers lines 2 and 6 to assert that the denotations of B and C in W_1 are :B' and :C', respectively (lines 23 and 26). The occurrence of :B' in line 23 is replaced by :C' (line 24). We now try to prove the antecedent of line 19, by showing that the denotation of B in W_1 is the same as the denotation of C in W_1 (lines 27 - 28). Since the denotations of B and C are both :C', this goal is transformed into the goal of showing that :C' is the same as :C' (line 30), which is immediately satisfied, completing the proof of the first branch of the split (line 31).

32.	**T(W_0,(Not(Eq(B,C)) => Know(A,Not(Eq(B,C)))))	Split
33.	**T(W_0,Not(Eq(B,C))) => T(W_0,Know(A,Not(Eq(B,C))))	L
34.	*T(W_0,Not(Eq(B,C)))	Ante
35.	*V(W_0,:B) ≠ V(W_0,:C)	L
36.	*:B' ≠ V(W_0,:C)	4
37.	:B' ≠ :C'	8
38.	**T(W_0,Know(A,Not(Eq(B,C))))	Cons
39.	**K(:A,W_0,W_1) => T(W_1,Not(Eq(B,C)))	L,K1
40.	K(:A,W_0,W_1)	Ante
41.	K(:A,W_1,w_3)/[W_1 ≠ w_3] -> K(:A,W_0,w_3))	K3
42.	*T(W_1,Eq(B,@(:B')))	2
43.	V(W_1,:B) = :B'	L
44.	*T(W_1,Eq(C,@(:C')))	6
45.	V(W_1,:C) = :C'	L
46.	**T(W_1,Not(Eq(B,C)))	Conse
47.	**V(W_1,:B) ≠ V(W_1,:C)	L
48.	**:B' ≠ V(W_1,:C)	43
49.	*:B' ≠ :C'	45
50.	*T	37

The proof of the second branch of the split is quite similar to that of the first. We begin with the goal of showing that if B and C are not the same, A knows that they are not the same (lines 32 - 33). This is an implication, so we assert the antecedent (line 34), which is translated into the meta-language assertion that the denotation of B in W_0 is not the

same as the denotation of C in W_0 (line 35). Since we know from lines 4 and 8 that these denotations are :B' and :C', respectively, we infer that :B' and :C' are not the same (lines 36 - 37).

Now we try to prove the consequent of line 33, showing that A knows that B and C are not the same individual (line 38). This translates into the meta-language goal of showing that in every world (represented by the Skolem constant W_1) which is compatible with what A knows in W_0, B is not the same individual as C (line 39). We assert that W_1 is compatible with what A knows in W_0 (line 40), which triggers K3 (line 41), and implies that the denotations of B and C in W_1 are :B' and :C', respectively (lines 42 - 45). We now try to prove that the denotations of B and C in W_1 are not the same (lines 46 - 47). Lines 43 and 45 transform this into the goal of showing that :B' and :C' are not the same (lines 48 - 49). This matches the assertion on line 37, so we are done (line 50). This completes both branches resulting from splitting line 10, so the entire proof is complete.

7. Automating Deductions about Knowledge and Action

7.1 Interpreting Axioms for Knowlege and Action

In this chapter, we deal with more complex problems of algorithmically generating deductions involving both knowledge and action. In the previous chapter, the examples of reasoning about knowledge alone involved only one or two possible worlds. The problem was simply to set up a typical world which is compatible with what someone knows, and to do a simple deduction relative to that world. In reasoning about both knowledge and action, however, we will be dealing with fairly complicated structures of several possible worlds. Managing the flow of information among these possible worlds is a major problem. The axioms relating to action and its interaction with knowledge must be structured to manage this information flow in an efficient manner. These issues can best be explored by examining the axioms involved. It may be helpful to refer to appendix A to compare the procedural versions of these axioms with the purely logical versions.

R1. $T(w_1,Res(trm.ev_1,p_1))/$
$[(trm.ev_1 \neq Do(trm.a_1,(trm.act_2; trm.act_3))) \wedge$
$(trm.ev_1 \neq Do(trm.a_1,If(p_2,trm.act_2,trm.act_3))) \wedge$
$(trm.ev_1 \neq Do(trm.a_1,While(p_2,trm.act_2)))] \langle\alpha\rangle$
$\exists w_2(R(D(w_1,Do(trm.a_1,trm.act_1)),w_1,w_2) \wedge T(w_2,p_1))$

R2. $T(w_1,Res(Do(trm.a_1,(trm.act_1; trm.act_2)),p_1)) \langle\alpha\rangle$
$T(w_1,Res(Do(trm.a_1,trm.act_1),Res(Do(@(D(w_1,trm.a_1)),trm.act_2),p_1)))$

R3. $T(w_1,Res(Do(trm.a_1,If(p_1,trm.act_1,trm.act_2)),p_2)) \langle\alpha\rangle$
$((T(w_1,p_1) \wedge T(w_1,Res(Do(trm.a_1,trm.act_1),p_2))) \vee$
$(\neg T(w_1,p_1) \wedge T(w_1,Res(Do(trm.a_1,trm.act_2),p_2))))$

R4. $T(w_1,Res(Do(trm.a_1,While(p_1,trm.act_1)),p_2)) \langle\alpha\rangle$
$T(w_1,Res(Do(trm.a_1,If(p_1,(trm.act_1; While(p_1,trm.act_1)),Nil)),p_2))$

R5. $R(Do(trm.a_1,Nil),w_1,w_2) \langle\alpha\rangle (w_1 = w_2)$

Res is the basic object-language predicate for talking about the results of actions. Recall from chapter 3 that $T(W_1,Res(Ev,P))$ means that in the world W_1 it is possible for the event Ev to occur and that in the resulting situation/world P is true. R1 is a translation rule from the object language to the meta-language that embodies this definition. The syntactic restrictions in R1 prevent its application to events that are described as complex sequences of actions. We use syntactic restrictions here because, although R1 is true for complex sequences, heuristically we want to use R2 - R4 instead if they are applicable. The syntactic restrictions on R1 simply rule it out in cases where R2 - R4 apply. R2 is an expansion rule for object-language expressions which transforms a formula which talks about the results of a sequence of actions into a formula which talks about doing the first action in the sequence, and then doing the rest. If the first action in the sequence is a simple action, R1 can the be applied. Otherwise, the decomposition of the complex action continues. R3 - R5 describe similar decompositions for conditionals and loops.

In addtion to Res, we also have the weaker operator Resl. Recall that the difference between Res and Resl is that Resl assumes that the event is possible, rather than asserting that it is. That is, Resl(Ev,P) means that if Ev were to happen, P would be true in the resulting situation, while Res(Ev,P) makes the additional assertion that it is possible for Ev to happen. The procedural version of the axiom which defines Resl for one-step actions is as follows:

R6. $T(w_1,Resl(trm.ev_1,p_1))/$
$[(trm.ev_1 \neq Do(trm.a_1,(trm.act_2; trm.act_3))) \wedge$
$(trm.ev_1 \neq Do(trm.a_1,If(p_2,trm.act_2,trm.act_3))) \wedge$
$(trm.ev_1 \neq Do(trm.a_1,While(p_2,trm.act_2)))] <=>$
$\forall w_2(R(D(w_1,Do(trm.a_1,trm.act_1)),w_1,w_2) \rightarrow T(w_2,p_1))$

Proving a goal involving Resl will be the same as proving a goal involving Res, except that we will assert that there is a situation which is the result of the event happening, instead of proving that there is such a situation.

C1. $T(w_1,Can(trm.a_1,trm.act_1,p_1))/$
$[(trm.act_1 \neq (trm.act_2; trm.act_3)) \wedge$
$(trm.act_1 \neq If(p_2,trm.act_2,trm.act_3)) \wedge$
$(trm.act_1 \neq While(p_2,trm.act_2))] \Leftarrow$
$T(w_1,Know(trm.a_1,And(Eq(@(D(w_1,trm.act_1)),trm.act_1),$
$Res(Do(@(D(w_1,trm.a_1)),trm.act_1),p_1)))))$

C2. $T(w_1,Can(trm.a_1,(trm.act_1; trm.act_2),p_1)) \Leftrightarrow$
$T(w_1,Can(trm.a_1,trm.act_1,Can(@(D(w_1,trm.a_1)),trm.act_2,p_1)))$

C3. $T(w_1,Can(trm.a_1,If(p_1,trm.act_1,trm.act_2),p_2)) \Leftrightarrow$
$((T(w_1,p_1) \wedge T(w_1,Can(trm.a_1,trm.act_1,p_2))) \vee$
$(\neg T(w_1,p_1) \wedge T(w_1,Can(trm.a_1,trm.act_2,p_2))))$

C4. $T(w_1,Can(trm.a_1,While(p_1,tr.act_1),p_2)) \Leftrightarrow$
$T(w_1,Can(trm.a_1,If(p_1,(trm.act_1; While(p_1,tr.act_1)),Nil),p_2))$

The object language operator Can describes the ability of an agent to obtain a result by performing a given action. In essence, $T(W_1,Can(A,Act,P)$ means that A knows how to achieve P by doing Act. C1 says that $T(W_1,Can(A,Act,P)$ is true if A knows what action Act describes, and knows that his doing Act will bring about P. Like R1, C1 is restricted to apply only to simple actions, but for a somewhat different reason. The trouble with applying C1 to a complex action is that it imposes too strong a requirement on whether an agent can carry out the action. Recall that in chapter 3, we said that knowing what action Act describes amounts to knowing exactly how to carry out Act, But if Act describes a sequence of actions, it need not exactly specify every step of the sequence in order for an agent to be able to carry out the sequence. The first step must be specified exactly, but for the remaining steps described by Act, it is only necessary that the agent know that at each step he will know what to do next. This allows for steps during which the agent acquires information about what to do. This idea is expressed by the expansion rule C2, with C3 and C4 integrating loops and conditionals into this structure.

This this finishes most of the very general axioms; the remainder are about specific

actions or predicates. Since all our examples involving both knowledge and action deal with opening safes, we will look at the axioms for Dial next.

D1a. $R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \Rightarrow$
$(\exists w_3(V(w_3,:Comb(x_2)) = x_1) \wedge :Safe(x_2) \wedge H(w_1,:At(a_1,x_2)))$

D1b. $R(:Do(a_1,:Dial(x_1,x_2)),w_1,F_1(a_1,x_1,x_2,w_1)) \Leftarrow$
$((V(w_3,:Comb(x_2)) = x_1) \wedge :Safe(x_2) \wedge H(w_1,:At(a_1,x_2)))$

D1a and D1b describe the circumstances under which it is possible to perform a dialing action. The thing being dialed must be a combination, the thing it is dialed on must be a safe, and the agent must be at the same place as the safe. We have split axiom D1 into two parts so that the existential quantifier could be removed from the left side. Removal of the quantifier is necessary for the unification-based matching routine to work properly. The biconditional in D1 had to be broken apart because the quantifier in a formula of the form $(\exists x(P) \Rightarrow Q)$ is Skolemized differently than the quantifier in $(\exists x(P) \Leftarrow Q)$.

D2. $R(:Do(\varepsilon_1,:Dial(x_1,x_2)),w_1,w_2) \rightarrow$
$((H(w_2,int.p_1) \Leftrightarrow$
$(((int.p_1 = :Open(x_2)) \wedge$
$((V(w_1,:Comb(x_2)) = x_1) \vee H(w_1,:Open(x_2)))) \vee$
$((int.p_1 \neq :Open(x_2)) \wedge H(w_1,int.p_1)))) \wedge$
$(V(w_2,int.trm_1) = V(w_1,int.trm_1)))$

D2 is significantly more complex than the preceding axioms and deserves special attention. D2 describes the total physical effects of dialing and incorporates the previous frame axiom D4. It says that in the situation/world resulting from dialing the combination x_1 on the safe x_2, any proposition is true just in case the proposition is that the safe is open, and either the combination dialed was the combination of the safe or the safe was already open, or the proposition is something other than that the safe is open and the propostion was true before the dialing took place. Also, any term refers to the same object in the new situation as it did in the preceding situation.

D2 is structured so that all assertions and goals "flow" backwards in time from the new situation to the old situation. This incorporates a solution to the frame problem which has been advocated by many authors, including Kowalski (1974), Hewitt (1975), and Waldinger (1975). That method is: To decide whether a proposition is true in the situation resulting from performing an action, first see whether the action made the proposition true or false and report success or failure accordingly, and if the action did not affect the proposition, see whether the action was true in the situation prior to the action. If there is a sequence of situations leading to the situation we are interested in, the procedure is recursive. D2 implements this procedure for physical propositions, because whether the safe is open is the only physical condition affected by dialing.

D2 also takes in to account another possibility ignored by most systems. That possibility is that we may be told something about a situation that implies something about a preceding situation. For instance, if we are told that the safe is not open after the dialing action, this implies that the safe was not open before the dialing action either. D2 handles this inference, since its output functions as a forward-chaining rule which, for any fact which is asserted about the situation resulting from the action, asserts the information it provides about the preceding situation.

Another significant fact about D2 is that it is a forward-chaining rather than a backward-chaining rule. If we are trying to verify the effects of a given action, we clearly want to confine our attention to assertions about that action. If we turned D2 around and made it a backward-chaining rule, the test whether the action involved is the one we are interested in would be the last thing checked. If we had axioms describing many other actions, the system could do a lot of useless search before finding the action it needed. On the other hand, if we were doing plan generation, we would be looking primarily for a specific result, and be willing to take any action that provided it to us, so a backward-

chaining rule would be appropriate. We would probably want to have two logically equivalent rules, with restrictions on the input variables to distinguish them. If the variable for the resulting situation were unbound, it would indicate that we were looking for a way to achieve a goal, and we would use the backward-chaining rule. If the variable for the resulting situation were bound, it would indicate that we were trying to verify the results of a particular action, and we would use the forward-chaining rule.

Also note that for D2, we do not use the contrapositive form. The natural contrapositive would be a backward-chaining rule for proving a goal of the form $\neg R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2)$, but the only goals of that form will be attempts to prove that a dialing action is not possible because its prerequisites are not satisfied. In these cases, D1a is the only appropriate rule to use. This is another case where the meta-language provides for more possibilities than the object language requires.

There is one problem with using D2 as a forward-chaining rule, however. Suppose we want to prove something of the form $T(W_1,Res(Do(A,Dial(C_1,Sf_1)),P)$. That is, we want to show that P could be achieved by A doing $Dial(C_1,Sf_1)$ in W_1. This would get translated by R1 into the meta-language goal of showing that there is some world w_2 such that $R(:Do(:A,:Dial(:C_1,:Sf_1)),W_1,w_2)$ is true and $T(w_2,P)$; i.e. w_2 is a possible outcome of $Do(A,Dial(C_1,Sf_1))$ happening in W_1, and P is true in w_2. The only rule we have for attacking a goal of the form $R(:Do(:A,:Dial(:C_1,:Sf_1)),W_1,w_2)$ is D1b. If all the prerequisites are satisfied, this rule will succeed, leaving w_2 bound to the Skolem term $F_1(:A,:C_1,:Sf_1,W_1)$ (which we will abreviate as W_2), and we will try to prove the remaining goal $T(W_2,P)$. Typically, showing this will depend on the information contained in D2. The trouble is that D2 needs the explicit assertion $R(:Do(:A,:Dial(:C_1,:Sf_1)),W_1,W_2)$ in order to trigger. Although we have just proved this formula, the proof procedure we currently have does

not cause it to be asserted. There is a danger here that asserting everything that is proved may produce an explosion of forward-chaining inferences. Therefore we will make a special case of formulas of the form $R(ev_1,w_1,w_2)$. Whenever anything matching this pattern is proved, it will be be explicitly asserted in order to give rules like D2 (and, as we shall see, D3) a chance to fire.

D3. $R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \rightarrow$
$(K(a_1,w_2,w_3)/[w_2 \neq w_3] \leftrightarrow$
$(\exists w_4(K(a_1,w_1,w_4)/[w_1 \neq w_4] \wedge$
$R(:Do(a_1,:Dial(x_1,x_2)),w_4,w_3)) \wedge$
$(H(w_2,:Open(x_2)) \leftrightarrow H(w_3,:Open(x_2)))))$

D3 explains how dialing affects the knowledge of the agent. Basically, it says that after dialing, the agent knows what action he has performed and he knows whether the safe is open. How this is expressed in terms of possible worlds was thoroughly explained in chapter 3. D3 is similar in many respects to D2. It is a forward-chaining rule with no contrapositve interpretation for exactly the same reasons as D2. The consequent of D3 introduces some new concerns. Just as in its purely logical form, the consequent of D3 is a biconditional. The interpretation of this biconditional, however, is different from either of the principal procedural interpretations introduced in chapter 6. (P <-> Q) may simply be regarded a syntactic abreviation for simultaneously expressing (P -> Q) and (P <- Q). That is, only goals and assertions corresponding to P are dealt with; ¬P is ignored. We make this restriction in D3, since the formula on the left side of <-> is $K(a_1,w_2,w_3)$, a formula whose negation should never occur.

D3 also contains syntactic restrictions on some of its subformulas. To see why, we need to look in detail at how D3 works. Suppose W_2 is the result of $Do(A,Dial(C_1,Sf_1)$ occurring in W_1. D3 will trigger on this assertion, producing the new assertion:

(1) $K(:A,W_2,w_3)/[W_2 \neq w_3] \leftrightarrow$
$(\exists w_4(K(:A,W_1,w_4)/[W_1 \neq w_4] \wedge$
$R(:Do(:A,:Dial(:C_1,:Sf_1)),w_4,w_3)) \wedge$
$(H(W_2,:Open(:Sf_1)) \Leftrightarrow H(w_3,:Open(:Sf_1))))$

This says that any world w_3 which is compatible with what A knows in the new situation W_2 must be the result of $Do(A,Dial(C_1,Sf_1))$ happening in some world which was possible according to what A knew in the old situation W_1, and agree with W_2 as to whether Sf_1 is open. The syntactic restriction $[W_2 \neq w_3]$ prevents consideration of W_2 itself as a binding for w_3. One reason for this restriction is that to allow W_2 as a binding for w_3 would generate no real information. We already know that W_2 is the result of $(Do(A,Dial(C_1,Sf_1)))$ happening in a world which was possible according to what A knew in W_1, namely W_1 itself; and W_2 obviously agrees with itself as to whether Sf_1 is open.

An even more important reason for the restriction $[W_2 \neq w_3]$ is to avoid generating an infinite number of assertions. Suppose we did allow w_3 to be bound to W_2. K2 would then apply, generating:

(2) $\exists w_4(K(:A,W_1,w_4)/[W_1 \neq w_4] \wedge$
$R(:Do(:A,:Dial(:C_1,:Sf_1)),w_4,W_2)) \wedge$
$(H(W_2,:Open(:Sf_1)) \Leftrightarrow H(W_2,:Open(:Sf_1)))$

The last part of this assertion could be deleted as a tautology, but the first part would be Skolemized and turned into the two assertions:

(3) $K(:A,W_1,W_4)$
(4) $R(:Do(:A,:Dial(:C_1,:Sf_1)),W_4,W_2)$

Assertion (4), however would trigger D3 all over again, recursing infinitely. The trouble is that the Skolem constant W_4 really refers to the same world as W_1, but our techniques are

not clever enough to catch this. If we had a more clever from form of subsumption, we could notice that the existentially quantified part of assertion (2) is already known to be true before Skolemization takes place. Alternatively, we could have an axiom saying that every world is the successor of exactly one other world with respect to a particular action. This fact plus assertion (4) would cause us to conclude that W_4 equals W_1, and W_4 would be replaced by W_1 in assertions (3) and (4), which would then be deleted by subsumption. Either of these methods, however, is much more complicated than using a simple syntactic restriction.

The other syntactic restriction in D3 ($[w_1 \neq w_4]$ in the third line) is used when the consequent of D3 is used as a backward-chaining rule. Suppose we know that in W_1, A knows that the safe is open, but does not know that P is true, where P has nothing to do with whether the safe is open; and we want to show that after performing $Dial(C_1,Sf_1)$, A still does not know that P is true. Asserting that A already knows that the safe is open is the easiest way to insure that finding out whether the safe is open after dialing will not indirectly tell A whether P is true. The forward inferences from the premises of the problem will include meta-language assertions to the effect that there is some possible world, say W_4, which is compatible with what A knows in W_1, i.e. $K(:A,W_1,W_4)$, and in which P is false, i.e. $\neg H(W_4,:P)$, and the safe is open.

In proving that A does not know that P is true after dialing the safe, we would assert that some world, say W_2, is the result of $Do(A,Dial(C_1,Sf_1))$ happening in W_1, which would trigger D2 and D3. As in the previous example, D3 would produce assertion (1), above. Then we would try to prove that there is some world which is compatible with what A knows in W_2 in which P is false, i.e. $K(:A,W_2,w_3) \wedge \neg H(w_3,:P)$. K2 provides one solution to the first goal in this conjunct. That is, one way to prove that A does not know that P is

true in W_2 is to prove that P is, in fact, not true in W_2. If this fails the only other applicable rule we have is assertion (1). Assertion (1) is now used as a backward-chaining rule and produces the following conjunctive goal:

(5) $K(:A,W_1,w_4)/[W_1 \neq w_4] \wedge$
$R(:Do(:A,:Dial(:C_1,:Sf_1)),w_4,w_3) \wedge$
$(H(W_2,:Open(:Sf_1)) \Leftrightarrow H(w_3,:Open(:Sf_1)))$

Now the syntactic restriction in goal (5) comes into play. If we let w_4 be bound to W_1, then the second conjunct would be solved by binding w_3 to W_2, which would give us the same case we considered when we applied K2 to our top level goal. So this restriction, like the others we have seen eliminates a redundancy in our search space.

If we continue with the deduction, we would eventually try binding w_4 to W_4. Solving the second conjunct would produce a binding for w_3, say W_3, which is the result of $Do(A,Dial(C_1,C_2))$ happening in W_4. Since the safe was open in both W_1 and W_4, the safe would also be open in both W_2 and W_3, so the third conjunct of goal (5) is also satisfied. This leaves us with only the second conjunct of the top level goal, $\neg H(W_3,:P)$, left to satisfy. Since W_3 is the successor of W_4, and $\neg H(W_3,:P)$ is true, the part of D2 that says what does not change will let us prove that $\neg H(W_3,:P)$ is true, completing the proof.

There is one more comment to make about D3. Since the output of D3 has one interpretation as a backward-chaining rule for proving goals of the form $K(a_1,w_1,w_2)$, and since most of the rules that have $K(a_1,w_1,w_2)$ as an antecedent are forward-chaining rules, we may have a problem proving goals like $K(A,W_1,w_2) \wedge T(w_2,P)$. A goal like this could come from trying to show that A doesn't know that $\neg P$ is true in W_1. The problem is that that we might use the output of D3, i.e. assertion (1), to solve the first part of the goal, but

$K(A,W_1,W_2)$ might have to be explicitly asserted to trigger forward-chaining rules to solve the second part of the goal. This is essentially the same problem as with $R(ov_1,w_1,w_2)$, which we pointed out in the discussion of D2. As in that case, we will make a practice of explicitly asserting any formula that matches $K(s_1,w_1,w_2)$ which has just been proved using a backward-chaining rule.

This concludes the analysis of the most important rules which we will use in doing deductions that involve both knowledge and action. The point of going into so much detail about them is to convey a feeling for the kinds of considerations that go into making a procedural deduction system work efficiently. It should also be obvious by now that no uniform inference procedure could hope to do the right thing in all these special cases. If there were a good theory of this sort of thing, it would probably be possible to paint a more coherent picture of what is going on. Unfortunately, such a theory does not currently exist.

7.2 An Example of an Action which Requires Knowledge

In the rest of this chapter, we will examine in detail algorithmically generated proofs of our three benchmark examples of reasoning about knowledge and action from chapter 1. In the first example, knowledge is required to achieve a goal; in the second, an action is used to obtain knowledge; and in the third, there is a sequence of two actions, where the first action produces information required by the second action. The possible-world structures for these proofs are the same as for the hand generated proofs in chapter 5. It may be of some help, therefore, to refer to figures 5.2 - 5.5 in studying the examples.

These examples are long and complicated, so the casual reader may prefer to skim them or skip over them entirely. Some general analysis of the examples is presented in section 7.5. The major point made there (and the thing to note in the proofs themselves) is how

tightly the procedural information we have built into the axioms constrains the search for proofs. In fact, there is almost no blind searching at all and the search space itself is finite. The general pattern of these proofs is that the goal is transformed into an implication which is proved by asserting the antecedent and deriving the consequent. The assertion of the antecedent triggers off many forward deductions which describe the possible world structure relevant to the problem, and the consequent of the goal is derived by doing simple backward-chaining inferences in that structure. As a result, although these proofs are long, no combinatorial explosion of formulas occurs.

The first example is to show that if John is at the same place as a safe, and he knows the combination to the safe, then he can open the safe by dialing the combination. As we saw in chapter 5, this proof requires one auxiliary fact in addition to those already discussed:

A1. $K(a_1,w_1,w_2)/[w_1 \neq w_2] \rightarrow$
$(H(w_2,:At(a_1,x_1)) \vee \neg H(w_1,:At(a_1,x_1)))$

This axiom says that if a person is at the same place as some object, he knows that he is at the same place as the object. A1 is stated somewhat differently than in chapter 5. Here it is treated as a forward-chaining rule triggered by $K(a_1,w_1,w_2)$. We restrict w_1 and w_2 from having the same binding in order not to allow K2 to trigger the tautologous conclusion $H(w_1,:At(a_1,x_1)) \vee \neg H(w_1,:At(a_1,x_1))$. The consequent is expressed as a disjunction, which is procedurally interpreted as two backward-chaining rules, because there does not seem to be any need to use a fact of this form as a forward-chaining rule.

We can now algorithmically generate the following proof (see figure 5.2):

Given: True(Safe(Sf_1))
True(At(John,Sf_1))
True(Exist(?X1,Know(John,Eq(?X1,Comb(St_1)))))

Prove: True(Can(John,Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))

1. ▪True(Safe(Sf_1))	Given
2. :Safe(Sf_1)	L
3. ▪True(At(John,Sf_1))	Given
4. H(W_0,:At(:John,:Sf_1))	L
5. ▪True(Exist(?X1,Know(John,Eq(Comb(Sf_1),?X1))))	Given
6. K(:John,W_0,w_1) -> T(w_1,Eq(Comb(Sf_1),@(:C)))	L,K1
7. ▪T(W_0,Eq(Comb(Sf_1),@(:C)))	K2
8. V(W_0,:Comb(:Sf_1)) = :C	L

Lines 1 - 8 state the premises of the problem and the forward inferences made from them. Line 1 says that Sf_1 is a safe, and line 3 says that John is at the same place as the safe. Lines 2, and 4 translate these facts into the meta-language. Line 5 expresses the fact that John knows the combination to the safe, by saying that there is some entity which John knows to be identical to the combination of the safe. In translating this statement into the meta-language, we let :C denote this entity. The meta-language translation of line 5 says that in every world which is compatible with what John knows in W_0, the combination of the safe is :C (line 6). Since W_0 is compatible with what John knows in W_0, we conclude that the comination of the safe in W_0 is :C (lines 7 - 8). This is the meta-language expression of the fact that the entity which John knows to be the combination to the safe, is the combination to the safe.

9. ▪▪True(Can(John,Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))	Goal
10. ▪T(W_0,Can(John,Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))	L
11. ▪▪T(W_0,Know(John, And(Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))))	C1
12. ▪▪K(:John,W_0,W_1) -> T(W_1,And(Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))	K1,L

Lines 9 - 12 state the goal and its meta-language translation. The goal is to show that

John can open the safe by dialing the combination (lines 9 - 10). According to C1, he can do this if he knows precisely what action dialing the combination of the safe is, and he knows that dialing the combination of the safe will result in the safe being open (line 11). This is re-expressed as the goal of showing that in every world which is compatible with what John knows in W_0, dialing the combination of the safe is the same action as it is in W_0, and dialing the combination of the safe will result in the safe being open (line 12).

13.	K(:John,W_0,W_1)	Ante
14.	K(:John,W_1,w_3)/[W_1 / w_3] -> K(:John,W_0,w_3)	K3
15.	H(W_1,:At(:John,x_1)) v ¬H(W_0,:At(:John,x_1))	A1
16.	*T(W_1,Eq(Comb(Sf_1),@(:C)))	6
17.	V(W_1,:Comb(:Sf_1)) = :C	L

Since line 12 is an implication goal, we assert the antecedent and try to prove the consequent. We assert K(:John,W_0,W_1) which triggers K3 (line 14) and A1 (line 15). We also conclude from line 6 that the combination of the safe in W_1 is :C (lines 16 - 17).

18.	T(W_1,And(Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))	
19.	**T(W_1,Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1))) ∧ T(W_1,Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))	Conse
20.	**T(W_1,Eq(@(D(W_0,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)))	Split
21.	**:Dial(V(W_0,:Comb(:Sf_1)),:Sf_1) = :Dial(V(W_1,:Comb(:Sf_1)),:Sf_1)	L
22.	**:Dial(:C,:Sf_1) = :Dial(V(W_1,:Comb(:Sf_1)),:Sf_1)	8
23.	**:Dial(:C,:Sf_1) = :Dial(:C,:Sf_1)	17
24.	*T	Eq

We now try to prove the consequent of line 12, that in W_1, dialing the combination of the safe is the same action as it is in W_0, and dialing the combination of the safe will result in the safe being open (lines 18 - 19). This conjunctive goal is split into two subgoals. First we try to prove that dialing the combination of the safe in W_0 is the same action as dialing

the combination of the safe in W_1 (lines 20 - 21). Since the combination of the safe is :C in both W_0 and W_1, this goal is transformed into the goal of showing that the action of dialing :C on $:Sf_1$ is identical to itself (line 22 - 23). This simplifies to proving T (line 24), so this branch of the split succeeds.

25.	**T(W_1,Res(Do(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))	Split
26.	**R(:Do(:John,:Dial(V(W_1,:Comb(:Sf_1)),:Sf_1)),W_1,w_2) ∧ T(w_2,Open(Sf_1))	R1,L
27.	**R(:Do(:John,:Dial(V(W_1,:Comb(:Sf_1)),:Sf_1)),W_1,w_2)	Split
28.	*R(:Do(:John,:Dial(:C,:Sf_1)),W_1,w_2)	17
29.	**(V(w_3,:Comb(:Sf_1)) = :C) ∧ :Safe(:Sf_1) ∧ H(W_1,:At(:John,:Sf_1))	D1b
30.	*V(w_3,:Comb(:Sf_1)) = :C	Split
31.	**:C = :C	8
32.	*T	Eq
33.	*:Safe(:Sf_1)	Split
34.	*T	2
35.	*H(W_1,:At(:John,:Sf_1))	Split
36.	*H(W_0,:At(:John,:Sf_1))	15
37.	*T	4

The other branch of the split is to show that dialing the combination of the safe in W_1 will result in the safe being open (line 25). This reduces to showing that it is possible to for John to dial the combination of the safe in W_1, and that in the resulting situation the safe is open (line 26). We split this goal, and try first to show that it is possible for John to dial the combination of the safe in W_1 (line 27). Since the combination of the safe in W_1 is :C, this is transformed into showing that it is possible for John to dial :C on Sf_1 in W_1 (line 28). According to D1b, this can be done if :C is a possible combination of Sf_1, Sf_1 is a safe, and John is at the same place as Sf_1 in W_1 (line 29). This goal splits three ways. The first subgoal is satisfied by the fact that :C is the combination of Sf_1 in W_0 (lines 30 - 32), and the second is satisfied by the fact that Sf_1 is asserted to be a safe (lines 33 - 34). The third

subgoal is that John is at the same place as the safe in W_1 (line 35). According to line 15, this is true if John is at the same place as the safe in W_0 (line 36). But this is one of the premises of the problem, so we have solved this subgoal (line 37), and also the conjunctive goal on line 29, and the goal on line 28.

38.	R(:Do(:John,:Dial(:C,:Sf_1)),W_1,W_2)	Solved
39.	V(W_3,:Comb(:Sf_1)) = :C	D1a
40.	=:Safe(:Sf_1)	D1a
41.	=H(W_1,:At(:John,:Sf_1))	D1a
42.	H(W_2,int.p_1) <=> (((int.p_1 = :Open(:Sf_1)) ∧ ((V(W_1,:Comb(:Sf_1)) = :C) ∨ H(W_1,:Open(:Sf_1)))) ∨ ((int.p_1 ≠ :Open(:Sf_1)) ∧ H(W_1,int.p_1)))	D2
43.	V(W_2,int.trm$_1$) = V(W_1,int.trm$_1$)	D2
44.	K(:John,W_2,w_3)/[W_2 ≠ w_3] <=> (∃w_4(K(:John,W_1,w_4)/[W_1 ≠ w_4] ∧ R(:Do(:John,:Dial(:C,:Sf_1)),w_4,w_3)) ∧ (H(W_2,:Open(:Sf_1)) <=> H(w_3,:Open(:Sf_1))))	D3

Since line 28 is a solved goal which matches R(ev_1,w_1,w_2), we now turn the solution to line 28 into an assertion. Line 28 was solved by binding w_2 to F_1(:John,:C,:Sf_1,W_1), but for simplicity we abreviate this as W_2 (line 38). This assertion triggers several forward-chaining rules. D1a produces assertions that :C is a possible combination of Sf_1, Sf_1 is a safe, and John is at the same place as Sf_1 in W_1 (lines 39 - 41). Basically, all this information is redundant. If we made the algorithm a bit more clever, it might notice that D1a and D1b together form a biconditional, and since we just proved line 38 using D1, we already know all the information that it contains, and it is unnecessary to trigger D1 again as a forward-chaining rule. Line 38 also triggers D2, which produces assertions describing the physical effects of dialing :C on Sf_1 (lines 42 - 43), and D3, which produces an assertion describing the effects of the action on John's knowledge.

45.	⊛⊛T(W_2,Open(Sf_1))	Split
46.	⊛⊛H(W_2,:Open(:Sf_1))	L
47.	⊛⊛(:Open(:Sf_1) = :Open(:Sf_1)) ∧ ((V(W_1,:Comb(:Sf_1)) = :C) ∨ H(W_1,:Open(:Sf_1)))	42
48.	⊛⊛(:Open(:Sf_1) = :Open(:Sf_1))	Split
49.	⊛T	Eq
50.	⊛⊛V(W_1,:Comb(:Sf_1)) = :C	Split
51.	⊛⊛:C = :C	17
52.	⊛T	Eq

Now we try to solve the second branch of the split of line 26. Taking the binding from the solution to the first branch, we try to show that the safe is open in W_2 (line 45 - 46). This is a question about the physical effects of dialing :C on Sf_1, so line 42 is used (line 47). Since we are asking about whether the safe is open (lines 48 - 49), we can solve our goal by showing that :C is the combination to the safe in W_1 (line 50). But we know this is true, so the current subgoal is satisfied (lines 51 - 52). Since this is the last branch of the last split, the entire proof is complete.

7.3 An Example of an Action which Produces Knowledge

In this example, we assume that C_1 is the combination of Sf_1 and that John knows that Sf_1 is not open. We show how to algoithmically generate a proof that if John tries to open Sf_1 by dialing C_1, he will know that C_1 is the combination of S_1. This proof involves the facts that after dialing C_1 John knows that he tried to open the safe, he knows whether the safe is open, and he understands how the safe being open depends on whether the combination of he dialed is the combination of the safe. (See figure 5.3.)

Given: True(Know(John,Not(Open(Sf_1))))
True(Eq(Comb(Sf_1),C_1))

Prove: True(Res1(Do(John,Dial(C_1,Sf_1)),Know(John,Eq(Comb(Sf_1),C_1))))

1.	*True(Know(John,Not(Open(Sf_1))))	Given
2.	K(:John,W_0,w_1) -> T(w_1,Not(Open(Sf_1)))	L,K1
3.	*T(W_0,Not(Open(Sf_1)))	K2
4.	¬H(W_0,:Open(:Sf_1))	L
5.	*True(Eq(Comb(Sf_1),C_1))	Given
6.	V(W_0,:Comb(:Sf_1)) = :C_1	L

Line 1 is the premise that John knows that the safe is not open, and line 2 is its meta-language translation. Since John knows that the safe is not open, we can conclude that the safe is not open (lines 3 - 4). Line 5 is the premise that C_1 is the combination of the safe, and line 6 is its meta-language translation.

7.	**True(Res1(Do(John,Dial(C_1,Sf_1)),Know(John,Eq(Comb(Sf_1),C_1))))	Goal
8.	**R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,W_1) -> T(W_1,Know(John,Eq(Comb(Sf_1),C_1)))	L,R6
9.	R(:Do(:John,:Dial(:C_1,:Sf_1)),W_0,W_1)	Ante
10.	V(W_3,:Comb(:Sf_1)) = :C_1	D1a
11.	:Safe(:Sf_1)	D1a
12.	H(W_0,:At(:John,:Sf_1))	D1a
13.	H(W_1,int.p_1) <=> (((int.p_1 = :Open(:Sf_1)) ∧ ((V(W_0,:Comb(:Sf_1)) = :C_1) ∨ H(W_0,:Open(:Sf_1)))) ∨ ((int.p_1 ≠ :Open(:Sf_1)) ∧ H(W_0,int.p_1)))	D2
14.	V(W_1,int.trm_1) = V(W_0,int.trm_1)	D2
15.	K(:John,W_1,w_3)/[W_1 ≠ w_3] <-> (∃w_4(K(:John,W_0,w_4)/[W_0 ≠ w_4] ∧ R(:Do(:John,:Dial(:C_1,:Sf_1)),w_4,w_3)) ∧ (H(W_1,:Open(:Sf_1)) <=> H(w_3,:Open(:Sf_1))))	D3

Line 7 is the goal of showing that if John dials C_1 on Sf_1 he will find out that C_1 is the combination of Sf_1. R6 transforms this into the goal of showing that if W_1 is the result of John dialing C_1 on Sf_1 in W_0, then in W_1 John knows that C_1 is the combination of Sf_1 (line 8). Since this is an implication, we assert the antecedent, and try to prove the consequent. Asserting that W_1 is the result of John dialing C_1 on Sf_1 in W_0 (line 9) triggers

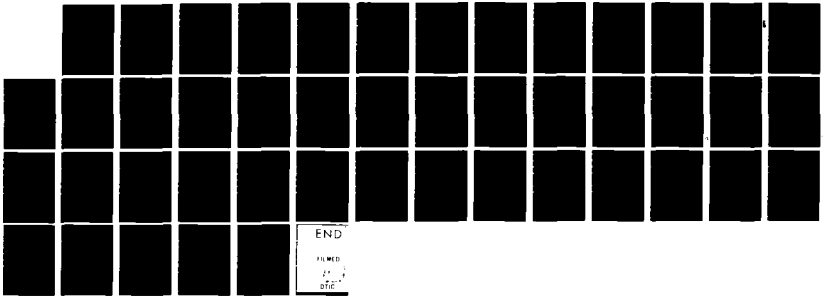
several forward-chaining rules. D1a produces assertions that C_1 is a possible combination of Sf_1, that Sf_1 is a safe, and that John is at the same place as Sf_1 (lines 10 - 12). D2 produces assertions specifying the physical effects of John dialing C_1 on Sf_1 (lines 13 - 14), and D3 produces a specification of the effects of the action on John's knowledge (line 15).

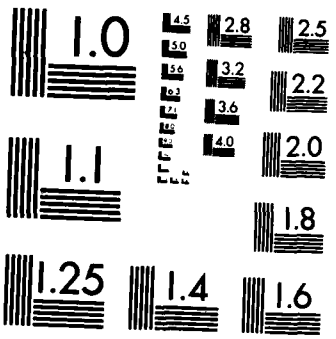
16.	##T(W_1,Know(John,Eq(Comb(Sf_1),C_1)))	Conse
17.	##K(:John,W_1,W_2) -> T(W_2,Eq(Comb(Sf_1),C_1))	L,K1
18.	K(:John,W_1,W_2)	Ante
19.	K(:John,W_2,w_3)/[$W_2 \neq w_3$] -: K(:John,W_1,w_3)	K3
20.	#H(W_2,:At(:John,x_1)) v ¬H(W_1,:At(:John,x_1))	A1
21.	#H(W_2,:At(:John,x_1)) v (((:At(:John,x_2) ≠ :Open(:Sf_1)) v ((V(W_0,:Comb(:Sf_1)) ≠ :C_1) v ¬H(W_0,:Open(:Sf_1)))) ∧ ((:At(:John(x_1) = :Open(:Sf_1)) v ¬H(W_0,int.p_1)))	13
22.	H(W_2,:At(:John,x_1)) v ¬H(W_0,:At(:John,x_1))	Eq

Now we try to prove the consequent of line 8, that in W_1 John knows that C_1 is the combination of Sf_1 (line 16). This is transformed into the goal of showing that in every world which is compatible with what John knows in W_1, C_1 is the combination of Sf_1 (line 17). This is an implication so we assert the antecedent, letting W_2 be a typical world which is possible according to what John knows in W_1 (line 18). This triggers K3 (line 19) and A1 (line 20). The result of A1 is the assertion that either John is at the same place as the safe in W_2, or he is not at the same place as the safe in W_1. Since whether John is at the same place as the safe in W_1 depends on the physical effects of John dialing C_1 on Sf_1 in W_0, line 13 applies to this assertion. The occurrence of ¬H(W_1,:At(:John,:Sf_1)) in line 20 is replaced by the corresponding instance of the right side of line 13 (line 21), but since John being at the same place as the safe is unaffected by this action, this formula simplifies to the assertion that either John is at the same place as the safe in W_2 or he is not in the same

AD-A126 244 REASONING ABOUT KNOWLEDGE AND ACTION(U) SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER R C MOORE OCT 80 SRI-TN-191 3/3

UNCLASSIFIED F/G 6/4 NL





place as the safe in W_0.

23.	K(:John,W_0,W_4)	15
24.	K(:John,W_4,w_3)/[$W_4 \neq w_3$] -> K(:John,W_0,w_3)	K3
25.	H(W_4,:At(:John,x_1)) ∨ ¬H(W_0,:At(:John,x_1))	A1
26.	*T(W_4,Not(Open(Sf_1)))	2
27.	¬H(W_4,:Open(:Sf_1))	L
28.	R(:Do(:John,:Dial(:C_1,:Sf_1)),W_4,W_2))	15
29.	V(W_5,:Comb(:Sf_1)) = :C_1	D1a
30.	*:Safe(:Sf_1)	D1a
31.	H(W_4,:At(:John,:Sf_1))	D1a
32.	H(W_2,int.p_1) <=> (((int.p_1 = :Open(:Sf_1)) ∧ ((V(W_4,:Comb(:Sf_1)) = :C_1) ∨ H(W_4,:Open(:Sf_1)))) ∨ ((int.p_1 ≠ :Open(:Sf_1)) ∧ H(W_4,int.p_1)))	D2
33.	*(((:At(:John,:Sf_1) = :Open(:Sf_1)) ∧ ((V(W_4,:Comb(:Sf_1)) = :C_1) ∨ H(W_4,:Open(:Sf_1)))) ∨ ((:At(:John,:Sf_1) ≠ :Open(:Sf_1)) ∧ H(W_4,int.p_1))) ∨ ¬H(W_0,:At(:John,x_1))	22
34.	*H(W_4,:At(:John,x_1)) ∨ ¬H(W_0,:At(:John,x_1))	Eq
35.	V(W_2,int.trm_1) = V(W_4,int.trm_1)	D2
36.	K(:John,W_2,w_3)/[$W_2 \neq w_3$] <-> (∃w_4(K(:John,W_4,w_4)/[$W_4 \neq w_4$] ∧ R(:Do(:John,:Dial(:C_1,:Sf_1)),w_4,w_3)) ∧ (H(W_2,:Open(:Sf_1)) <=> H(w_3,:Open(:Sf_1))))	D3

The assertion on line 18 that W_2 is compatible with everything John knows in W_1 also triggers line 15 as a forward-chaining rule. This results in assertions to the effect that there is some world, say W_4, such that W_4 is compatible with everything that John knows in W_0 (line 23), and W_2 is the result of John dialing C_1 on Sf_1 in W_4 (line 28), and that W_2 agrees with W_1 as to whether Sf_1 is open (line 35). The assertion that W_4 is compatible with what John knows in W_0 triggers K3 (line 24) and A1 (line 25). We also conclude that the safe is not open in W_4 (lines 26 - 27), since this is something John knows in W_0.

The assertion that W_2 is the result of Do(John,Dial(C_1,Sf_1)) happening in W_4 triggers D1a

to assert that C_1 is a possible combination of Sf_1 (line 29), that Sf_1 is a safe (line 30), and that John is at the same place as the safe in W_4 (line 31). D2 triggers, producing an assertion which describes how the physical conditions in W_2 depend on its being the result of Do(John,Dial(C_1,Sf_1)) happening in W_4 (line 32). Since line 22 involves a physical condition in W_2, this assertion is immediately applied as a substitution rule. The instance of ¬H(W_2,:At(:John,:Sf_1)) in line 22 is replaced by the appropriate instance of the right side of line 32. Since the proposition in question has nothing to do with the safe being open, this expression simplifies to ¬H(W_4,:At(:John,:Sf_1)), leaving the whole expression as it is on line 34. Since this repeats line 25, it is deleted.

Some comment should be made on this last set of steps. We effectively had one assertion that John knows where he before the dialing action, and another assertion that he knows where he is after the dialing action. But since the dialing action does not affect where John is, and he knows this, one of these two assertions is redundant; we could deduce either of them given the other. By using the frame axiom for dialing, we transformed one of these assertions into the other, enabling us to recognize the redundancy and eliminate it.

Line 35 is the rest of the frame axiom for dialing, noting that dialing does not change the reference of any term expressions. The last rule triggered by the assertion that W_2 is the result the dialing happening in W_4 is D3, which produces a description of the effects of the action on John's knowledge in W_2 (line 36).

37.	▪H(W_1,:Open(:Sf_1)) <=> H(W_2,:Open(:Sf_1)))	15
38.	▪(((:Open(:Sf_1) = :Open(:Sf_1)) ∧ ((V(W_0,:Comb(:Sf_1)) = :C_1) ∨ H(W_0,:Open(:Sf_1)))) ∨ ((:Open(:Sf_1) ≠ :Open(:Sf_1)) ∧ H(W_0,int.p_1))) <=> H(W_2,:Open(:Sf_1))	13
39.	▪((V(W_0,:Comb(:Sf_1)) = :C_1) ∨ H(W_0,:Open(:Sf_1))) <=> H(W_2,:Open(:Sf_1))	Eq

40.	*((:C_1 = :C_1) ∨ H(W_0,:Open(:Sf_1))) <=> H(W_2,:Open(:Sf_1))	5
41.	*H(W_2,:Open(:Sf_1))	Eq
42.	*((:Open(:Sf_1) = :Open(:Sf_1)) ∧ ((V(W_4,:Comb(:Sf_1)) = :C_1) ∨ H(W_4,:Open(:Sf_1)))) ∨ ((:Open(:Sf_1) ≠ :Open(:Sf_1)) ∧ H(W_4,int.p_1))	32
43.	(V(W_4,:Comb(:Sf_1)) = :C_1) ∨ H(W_4,:Open(:Sf_1))	Eq

Now we pop back up and resume considering the consequences of the assertion on line 18 that W_2 is compatible with what John knows in W_1. The last inference which is drawn from this assumption and line 15 is that the safe is open in W_2 if and only if it is open in W_1 (line 38). Both sides of this assertion refer to physical conditions in situations to which a frame assertion applies, line 13 for W_1, and line 32 for W_2. We first replace H(W_1,:Open(Sf_1)) in line 37 with the matching instance of the right side of line 13 (line 38). This expression simplifies to (V(W_0,:Comb(:Sf_1)) = :C_1) ∨ H(W_0,:Open(:Sf_1)) (line 39). Since we know that C_1 is the combination of Sf_1 in W_0, the left side of line 39 simplifies to T. Since this is a biconditional, we conclude that the right side is also true (lines 40 - 41), leaving us with the assertion that the safe is open in W_2. We now apply the frame assertion for W_2 to line 41, and the resulting expression simplifies to the assertion that either C_1 is the combination of the safe in W_4 or the safe is open in W_4 (lines 42 - 43). Since W_2 is the result Do(John,Dial(C_1,Sf_1)) happening in W_4, these are the only two alternatives that could lead to the safe being open in W_2.

44.	**T(W_2,Eq(Comb(:Sf_1),:C_1))	Conse
45.	**V(W_2,:Comb(:Sf_1)) = :C_1	L
46.	*V(W_4,:Comb(:Sf_1)) = :C_1	35
47.	*¬H(W_4,:Open(:Sf_1)))	43
48.	*T	27

Finally we come to trying to prove the consequent of line 17, given all the conclusions which we have drawn from the antecedent. The goal is to show that C_1 is the combination of Sf_1 in W_2 (lines 44 - 45). Since dialing does not change the combination of the safe, this is equivalent to showing that C_1 is the combination of Sf_1 in W_4 (line 46). According to line 43, we can show this if we can show that the safe is not open in W_4 (line 47). But we already know that this is true from line 27, so the proof is complete (line 48).

7.4 An Example of Acquiring Knowlege Required for an Action

Our final example is to produce a proof that if John has a piece of paper with the combination of the safe written on it, if he can read, and if he is at the safe, then he can open the safe by reading the piece of paper and dialing the combination. This requires the introduction of a new action *Read*, and the associated predicates *Reads* and *Info*.

INF1. $T(w_1,Info(trm.x_1,exp_1)) \Leftrightarrow (V(w_1,:Info(D(w_1,trm.x_1))) = exp_1)$

RDS1. $K(a_1,w_1,w_2)/[w_1 \neq w_2] \rightarrow$
$(H(w_2,:Reads(a_1)) \vee \neg H(w_1,:Reads(a_1)))$

An object-language formula of the form Info(X,Exp) means that the object X has the information Exp written on it, where Exp is some well-formed object-language expression. INF1 translates this into the meta-language. An object-language language formula of the form Reads(A) means that A can read. We will treat Reads as though it were a simple physical predicate, so its translation into the meta-language will be handled by L9a. RDS1 says that anyone who can read knows that he can read. This rule is expressed in exactly the same form as A1.

RD1a. $R(:Do(a_1,:Read(x_1)),w_1,w_2) \Rightarrow$
$(H(w_1,:Reads(a_1)) \wedge H(w_1,:At(a_1,x_1)))$

RD1b. $R(:Do(a_1,:Read(x_1)),w_1,F_2(a_1,x_1,w_1)) \Leftarrow$
$(H(w_1,:Reads(a_1)) \wedge H(w_1,:At(a_1,x_1)))$

RD2. $R(:Do(a_1,:Read(x_1)),w_1,w_2) \rightarrow$
$(K(a_1,w_2,w_3)/[w_2 \neq w_3] \Leftrightarrow$
$\exists w_4(K(a_1,w_1,w_4)/[w_1 \neq w_4] \wedge$
$(V(w_4,:Info(x_1)) = V(w_1,:Info(x_1))) \wedge$
$R(:Do(a_1,:Read(x_1)),w_4,w_3)))$

RD3. $R(:Do(a_1,:Read(x_1)),w_1,w_2) \rightarrow$
$((V(w_2,int.trm_1) = V(w_1,int.trm_1)) \wedge$
$(H(w_2,int.p_1) \Leftrightarrow H(w_1,int.p_1))$

Read(X) is the object-language representation of the action of reading the information written on the object X. RD1a and RD1b specify the prerequisites for reading something; the agent has to be able to read, and he has to be at the same place as the thing he is going to read. These to rules are expressed in the same form as D1a and D1b. RD2 gives the effects of reading on the knowledge of the agent. It says in the usual form that he knows what was written on the object, and he knows that he has just read what was written on the object. RD3 describes the physical effects of reading. Since there really aren't any, it just says that all physical conditions are the same as they were before the action took place.

The proof is as follows (see figure 5.5):

Given: $True(Safe(Sf_1))$
$True(At(John,Sf_1))$
$True(At(John,Ppr_1))$
$True(Reads(John))$
$True(Know(John,Exist(?X1,And(Eq(Comb(Sf_1),?X1),Info(Ppr_1,?X1)))))$

Prove: $True(Can(John,(Read(Ppr_1); Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))$

1. $*True(Safe(Sf_1))$	Given
2. $:Safe(Sf_1)$	L
3. $*True(At(John,Sf_1))$	Given
4. $H(W_0,:At(:John,:Sf_1))$	L
5. $*True(At(John,Ppr_1))$	Given
6. $H(W_0,:At(:John,:Ppr_1))$	L

7.	*True(Reads(John))	Given
8.	H(W_0,:Reads(:John))	L
9.	*True(Know(John,Exist(?X1,And(Eq(Comb(Sf_1),?X1),Info(Ppr_1,?X1)))))	Given
10.	K(:John,W_0,w_1) -> T(w_1,Exist(?X1,And(Eq(Comb(Sf_1),?X1),Info(Ppr_1,?X1))))	L,K1
11.	*T(W_0,Exist(?X1,And(Eq(Comb(Sf_1),?X1),Info(Ppr_1,?X1))))	K2
12.	V(W_0,:Comb(:Sf_1)) = :C_0	L
13.	V(W_0,:Info(:Ppr_1)) = @(:C_0)	L

Lines 1 - 8 give the first four premises and their meta-language translations. These are the assertions that Sf_1 is a safe, that John is at the same place as the safe, that John is at the same place as the piece of paper Ppr_1, and that John can read. Line 9 says that John knows that there is some entity which is the combination of the safe, and which is the information written on the piece of paper; i.e. John knows that the combination of the safe is the only thing written on the piece of paper. K1 transforms this into the assertion that in every world which is compatible with what John knows in the actual world, the combination of the safe is the only thing written on the piece of paper (line 10). K2 triggers this rule to assert that the combination of the safe, represented by :C_0, is actually the only thing written on the piece of paper (lines 11 - 13).

14.	**True(Can(John,(Read(Ppr_1); Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))	Goal
15.	*T(W_0,Can(John,Read(Ppr_1), Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1))))	L,C2
16.	**T(W_0,Know(John,And(Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1)), Res(Do(@(D(W_0,John)),Read(Ppr_1)), Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1))))))	C1
17.	**K(:John,W_0,W_1) -> T(W_1,And(Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1)), Res(Do(@(D(W_0,John)),Read(Ppr_1)), Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))))	K1

Line 14 states the goal that John can open the safe by reading the piece of paper and dialing the combination of the safe. C2 expands this into the goal that by reading the piece of paper John can bring it about that he can open the safe by dialing the combination (line

15). C1 expands this into the goal that John knows what action reading the piece of paper is, and he knows that reading the piece of paper would bring it about that he can open the safe by dialing the combination (line 16). Finally, K1 transforms this into the goal that if W_1 is a typical world which is possible according to what John knows in W_0, then it is true in W_1 that reading the piece of paper is the same action that it is in the actual world, and that John reading the piece of paper would bring it about that he can open the safe by dialing the combination (line 17).

18.	K(:John,W_0,W_1)	Ante
19.	K(:John,W_1,w_3)/[W_1 ≠ w_3] -> K(:John,W_0,w_3)	K3
20.	H(W_1,:At(:John,x_1)) v ¬H(W_0,:At(:John,x_1))	A1
21.	H(W_1,:Reads(:John)) v ¬H(W_0,:Reads(:John))	RDS1
22.	*T(W_1,Exist(?X1,And(Eq(Comb(Sf_1),?X1),Info(Ppr_1,?X1))))	10
23.	V(W_1,:Comb(:Sf_1)) = :C_1	L
24.	V(W_1,:Info(:Ppr_1)) = @(:C_1)	L

To prove the implication on line 17, we first assert the antecedent, that W_1 is compatible with what John knows in W_0 (line 18). This triggers K3, A1, and RDS1 (lines 19 - 21). It also triggers line 10 to assert that the combination of the safe in W_1, represented by :C_1, is the only thing written on the piece of paper in W_1 (lines 22 - 23).

25.	**T(W_1,And(Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1)), Res(Do(@(D(W_0,John)),Read(Ppr_1)), Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))))	Conse
26.	**T(W_1,Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1))) ∧ T(W_1,Res(Do(@(D(W_0,John)),Read(Ppr_1)), Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1))))	L
27.	**T(W_1,Eq(@(D(W_0,Read(Ppr_1))),Read(Ppr_1)))	Split
28.	**:Read(:Ppr_1) = :Read(:Ppr_1)	L
29.	*T	Eq

Now we try to prove the consequent of line 17, the goal that reading the piece of paper

in W_1 is the same action that it is in the actual world and that John reading the piece of paper in W_1 would bring it about that John can open the safe by dialing the combination (lines 25 - 26). Since this is a conjunctive goal, we split it into two subgoals. The first subgoal, showing that reading the piece of paper in W_1 is the same action as it is in the actual world, is easily solved if we treat Ppr_1 as a rigid designator for the piece of paper. This amounts to assuming that John knows what object Ppr_1 is. If he knows what object he is supposed to read, then he certainly knows what action reading that object is (lines 27 - 29).

30.	**T(W_1,Res(Do(@(D(W_0,John)),Read(Ppr_1)), Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1))))	Split
31.	**R(Do(:John,Read(Ppr_1)),W_1,w_2) ∧ T(w_2,Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1))))	R1,L
32.	*R(Do(:John,Read(Ppr_1)),W_1,w_2)	Split
33.	**H(W_1,:Reads(:John)) ∧ H(W_1,:At(:John,:Ppr_1))	RD1b
34.	*H(W_1,:Reads(:John))	Split
35.	*H(W_0,:Reads(:John))	21
36.	*T	8
37.	*H(W_1,:At(:John,:Ppr_1))	Split
38.	*H(W_0,:At(:John,:Ppr_1))	20
39.	*T	6

Now we try to prove the second subgoal which results from splitting line 26, the goal that John reading the piece of paper in W_1 would bring it about that John can open the safe by dialing the combination (line 30). R1 transforms this into the goal that there is some world which is the result of John reading the piece of paper in W_1 and in which John can open the safe by dialing the combination (line 31). This goal is split, with the first subgoal being to find a world which is the result of John reading the piece of paper in W_1 (line 32). RD1b says there is such a world if John can read, and if John is at the same place as the piece of paper (line 33). This goal is also split, and we first try to show that

John can read in W_1 (line 34). Since if John can read, he knows he can read, this goal is satisfied if John can read in the actual world (line 35). This was one of the premises of the problem, so this branch of the split succeeds (line 36). The other branch requires us to show that John is at the same as the piece of paper in W_1 (line 37). We are assuming that if John is at the same place as the piece of paper, he knows he is at the same place as the piece of paper, so this goal is satisfied if John is at the same place as the piece of paper in the actual world (line 38). This is also one of the premises, so this branch of the split is also satisfied (line 39). This proves line 33, and hence produces a solution to line 32, with w_2 bound to $F_2(:John,:Ppr_1,W_1)$. To keep formulas short in the rest of the proof, we will abreviate this as W_2.

40.	$R(Do(:John,Read(Ppr_1)),W_1,W_2)$	Solved
41.	$H(W_1,:Reads(:John))$	RD1a
42.	$H(W_1,:At(:John,:Ppr_1))$	RD1a
43.	$K(:John,W_2,w_3)/[W_2 \neq w_3] \leftrightarrow$ $\exists w_4(K(:John,W_1,w_4)/[W_1 \neq w_4] \wedge$ $(V(w_4,:Info(:Ppr_1)) = V(W_1,:Info(:Ppr_1))) \wedge$ $R(:Do(:John,:Read(:Ppr_1)),w_4,w_3))$	RD2
44.	$\neg V(W_2,int.trm_1) = V(W_1,int.trm_1)$	RD3
45.	$V(W_2,:Comb(:Sf_1)) = :C_1$	23
46.	$\neg V(W_2,int.trm_1) = V(W_1,int.trm_1)/[int.trm_1 \neq :Comb(:Sf_1)]$	23
47.	$V(W_2,:Info(:Ppr_1)) = \neg(:C_1)$	24
48.	$V(W_2,int.trm_1) = V(W_1,int.trm_1)/$ $[(int.trm_1 \neq :Comb(:Sf_1) \wedge (int.trm_1 \neq :Info(:Ppr_1)]$	24
49.	$H(W_2,int.p_1) \leftrightarrow H(W_1,int.p_1)$	RD3

Since we have just solved a goal which matches $R(ev_1,w_1,w_2)$, we assert the solution (line 40). RD1a causes us to assert that John can read in W_1 and that John is at the same place as the piece of paper in W_1 (lines 41 - 42). RD2 triggers, producing a description of the effect of reading the piece of paper on John's knowledge (line 43). RD3, the frame axiom for reading also triggers, producing two assertions. The first assertion is that all

terms in W_2 refer to the same objects as they do in W_1 (line 44). However, on lines 23 and 24, we have assertions specifying the referents of two particular terms in W_1, the combination of the safe, and the information written on Ppr_1. In section 6.4 we described a rule for equality substitutions where the term being substituted for is more general than the assertion specifying the substitution. Applying this rule in the present case produces an assertion that all terms in W_2 refer to the same objects as they do in W_1, but with syntactic restrictions preventing the assertion from being applied to the terms for the combination of the safe and the information written on the piece of paper. We also get specific assertion saying that the information written on the piece of paper in W_2 is the combination $:C_1$ (actually, its standard name), and that the combination of the safe in W_1 is also $:C_1$ (lines 45 - 48). In addition, to these three assertions, RD3 produces the assertion that all simple physical conditions are the same in W_2 as they are in W_1 (line 49).

50.	*T(W_2,Can(@(D(W_0,John)),Dial(Comb(Sf_1),Sf_1),Open(Sf_1)))	Split
51.	**T(W_2,Know(@(D(W_0,John)), And(Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))))	C1
52.	**K(:John,W_2,W_3) -> T(W_3,And(Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))	K1,L

Now we try to prove the second half of the goal on line 31, that in the world which we have just shown to be the result of John reading the piece of paper in W_1, namely W_2. John can open the safe by dialing the combination (line 50). C1 reduces this to the goal that in W_2 John knows what action dialing the combination of the safe is, and he knows that dialing the combination of the safe would result in the safe being open (line 51). K1 reduces this to showing that if W_3 is a typical world which is compatible with what John knows in W_2, then in W_3, dialing the combination of the safe is the same action as it is in

W_2 and John dialing the combination of the safe would result in the safe being open (line 52). To prove this, we first assert the antecedent, and then try to prove the consequent.

53.	K(:John,W_2,W_3)	Ante
54.	K(:John,W_3,w_3)/[W_3 ≠ w_3] -> K(:John,W_2,w_3)	K3
55.	∗H(W_3,:At(:John,x_1)) ∨ ¬H(W_2,:At(:John,x_1))	A1
56.	H(W_3,:At(:John,x_1)) ∨ ¬H(W_1,:At(:John,x_1))	49
57.	∗H(W_3,:Reads(:John)) ∨ ¬H(W_2,:Reads(:John,)	RDS1
58.	H(W_3,:Reads(:John)) ∨ ¬H(W_1,:Reads(:John))	49

Asserting that W_3 is compatible with what John knows in W_2 triggers a number of forward-chaining rules, including K3, A1, and RDS1 (lines 53, 55, and 57). Since the formulas asserted by A1 and RDS1 both mention a physical condition in W_2, and every physical condition in W_2 is the same as in W_1, The occurrences of W_2 are replaced by W_1 (lines 56 and 58).

59.	K(:John,W_1,W_4)	43
60.	K(:John,W_4,w_3)/[W_4 ≠ w_3] -> K(:John,W_1,w_3)	K3
61.	H(W_4,:At(:John,x_1)) ∨ ¬H(W_1,:At(:John,x_1))	A1
62.	H(W_4,:Reads(:John)) ∨ ¬H(W_1,:Reads(:John))	RDS1
63.	K(:John,W_0,W_4)	19
64.	K(:John,W_4,w_3)/[W_4 ≠ w_3] -> K(:John,W_0,w_3)	K3
65.	H(W_4,:At(:John,x_1)) ∨ ¬H(W_0,:At(:John,x_1))	A1
66.	H(W_4,:Reads(:John)) ∨ ¬H(W_0,:Reads(:John))	RDS1
67.	∗T(W_4,Exist(?X1,And(Eq(Comb(Sf_1),?X1),Info(Ppr_1,?X1))))	10
68.	V(W_4,:Comb(:Sf_1)) = :C_4	L
69.	V(W_4,:Info(:Ppr_1)) = @(:C_4)	L

The assertion K(:John,W_2,W_3) on line 53 also triggers the assertion on line 43 which decribes what John knows in W_2. This results in assertions that there is a world, say W_4, which is compatible with what John knows in W_1 (line 59) in which the information written on the piece of paper is the same as in W_1 (line 70) , and in which the result of John reading the piece of paper is W_3 (line 76). The first of these assertions triggers K3, A1, and

RDS1 as usual (lines 60 - 62). It also triggers line 19 to assert that W_4 is compatible with what John knows in W_0 (line 63). This is the first time we have actually used the fact that for a particular knower, K is transitive (axiom K3). This in turn triggers K3, A1, and RDS1 (lines 64 - 66). It also triggers line 10 to assert that in W_4 the combination of the safe, represented by $:C_4$, is the only thing written on the piece of paper (lines 67 - 69).

70.	$\Box V(W_4,:Info(:Ppr_1)) = V(W_1,:Info(:Ppr_1))$	43
71.	$\Box @(:C_4) = V(W_1,:Info(:Ppr_1))$	69
72.	$\Box @(:C_4) = @(:C_1)$	24
73.	$:C_4 = :C_1$	L
74.	$V(W_4,:Comb(:Sf_1)) = :C_1$	68
75.	$V(W_4,:Info(:Ppr_1)) = @(:C_1)$	69

The second assertion produced by line 43 is that the information written on the piece of paper in W_4 is the same as the information written on the piece of paper in W_1. Since the information written on the piece of paper in W_4 is the standard name of the combination $:C_4$, and in W_1 the information written on the piece of paper is the standard name of the combination $:C_1$, we conclude that $:C_4$ is the same as $:C_1$ (lines 71 - 73). This causes us to substitute $:C_1$ for $:C_4$ in lines 68 and 69, producing assertions that the combination of the safe in W_4 is $:C_1$, and the information written on the piece of paper in W_4 is the standard name of $:C_1$ (lines 74 - 75). This gives us all the information we need to prove that in the actual world, W_0, John knows that reading the piece of paper would result in his knowing the combination of the safe.

76.	$R(:Do(:John,:Read(:Ppr_1)),W_4,W_3)$	43
77.	$H(W_4,:Reads(:John))$	RD1a
78.	$H(W_4,:At(:John,:Ppr_1))$	RD1a
79.	$K(:John,W_3,w_3)/[W_3 \neq w_3] \leftrightarrow$ $\exists w_4(K(:John,W_4,w_4)/[W_4 \neq w_4] \wedge$	RD2

	$(V(w_4,:Info(:Ppr_1)) = V(W_4,:Info(:Ppr_1))) \wedge$ $R(:Do(:John,:Read(:Ppr_1)),w_4,w_3))$	
80.	$\Box V(W_3,int.trm_1) = V(W_4,int.trm_1)$	RD3
81.	$V(W_3,:Comb(:Sf_1)) = :C_1$	74
82.	$\Box V(W_3,int.trm_1) = V(W_4,int.trm_1)/[int.trm_1 \neq :Comb(:Sf_1)]$	74
83.	$V(W_3,:Info(:Ppr_1)) = @(:C_1)$	75
84.	$V(W_3,int.trm_1) = V(W_4,int.trm_1)/$ $[(int.trm_1 \neq :Comb(:Sf_1) \wedge (int.trm_1 \neq :Info(:Ppr_1)]$	75
85.	$H(W_3,int.p_1) \Leftrightarrow H(W_4,int.p_1)$	RD3
86.	$\Box H(W_4,:At(:John,x_1)) \vee \neg H(W_1,:At(:John,x_1))$	56
87.	$\Box H(W_4,:Reads(:John)) \vee \neg H(W_1,:Reads(:John))$	58

The last assertion added by line 43 is that W_3 is the result of John reading the piece of paper in W_4. This triggers D1a to assert that in W_4 John can read and he is at the same place as the piece of paper (lines 77 - 78). It also triggers RD2 to produce a description of the effects of reading the piece of paper on what John knows in W_3 (line 79). Finally, it triggers the frame axiom RD3 to produce a description of the physical effects of John reading the piece of paper in W_4. The first part of this description is the assertion that all terms refer to the same objects in W_3 as they do in W_4 (line 80). Since we have specific assertions about the referents of the terms for the combination of the safe and the information written on the piece of paper in W_4, we syntactically exclude these cases from the general axiom, and explicitly assert that the information written on the piece of paper in W_3 is the standard name of $:C_1$ and that the combination of the safe in W_3 is $:C_1$ (lines 81 - 84). The other assertion produced by the frame axiom is that all simple physical conditions are the same in W_3 as they are in W_4 (line 85). This causes the references in line 56 to what John is near in W_3 and in line 58 to John being able to read in W_3 to be replaced by these same conditions in W_4 (lines 86 - 87). This transforms these assertions into copies of lines 61 and 62, so they are deleted.

88.	**T(W_3,And(Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)), Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1))))	Conse
89.	**T(W_3,Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1))) ∧ T(W_3,Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))	L
90.	**T(W_3,Eq(@(D(W_2,Dial(Comb(Sf_1),Sf_1))),Dial(Comb(Sf_1),Sf_1)))	Split
91.	**:Dial(:V(W_2,:Comb(:Sf_1)),:Sf_1) = :Dial(:V(W_3,:Comb(:Sf_1)),:Sf_1)	L
92.	**(V(W_2,:Comb(:Sf_1)) = V(W_3,:Comb(:Sf_1))) ∧ (:Sf_1 = :Sf_1)	Eq
93.	**V(W_2,:Comb(:Sf_1)) = V(W_3,:Comb(:Sf_1))	Split
94.	**:C_1 = V(W_3,:Comb(:Sf_1))	45
95.	**:C_1 = :C_1	81
96.	*T	Eq
97.	**:Sf_1 = :Sf_1	Split
98.	*T	Eq

Now we go back and try to prove the consequent of line 52 from the information generated by asserting the antecedent. The goal is to show that in W_3, dialing the combination of the safe is the same action as in W_2, and that John dialing the combination of the safe would result in the safe being open (lines 88 - 89). We split this goal into its two subgoals. The first subgoal reduces to showing that the combination of the safe is the same in W_2 as it is in W_3, and that the safe is identical to itself (lines 90 - 92). This goal is also split, and the first subgoal is solved by noting that the combination to the safe in both W_2 and W_3 is :C_1 (lines 93 - 96). This is basically a proof that in the actual world, W_0, John knows that reading the piece of paper would result in his knowing the combination of the safe. The goal that the safe is identical to itself is trivially solved by the simplification rules for identity (lines 97 - 98).

99.	**T(W_3,Res(Do(@(D(W_2,John)),Dial(Comb(Sf_1),Sf_1)),Open(Sf_1)))	Split
100.	**R(Do(:John,:Dial(V(W_3,Comb(:Sf_1)),:Sf_1)),W_3,w_5) ∧ T(w_5,Open(Sf_1))	R1,L
101.	**R(Do(:John,:Dial(V(W_3,Comb(:Sf_1)),:Sf_1)),W_3,w_5)	Split
102.	*R(Do(:John,:Dial(:C_1,:Sf_1)),W_3,w_5)	81
103.	**(V(w_3,:Comb(:Sf_1)) = :C_1) ∧	D1b

	$:Safe(:Sf_1) \wedge H(W_3,:At(:John,:Sf_1))$	
104.	$**V(w_3,:Comb(:Sf_1) = :C_1$	Split
105.	$*:C_0 = :C_1$	12
106.	$**V(w_3,:Comb(:Sf_1) = :C_1 /$ $[w_3 \neq W_0]$	12
107.	$*:C_1 = :C_1$	23
108.	$*T$	Eq
109.	$*:Safe(:Sf_1)$	Split
110.	$*T$	2
111.	$**H(W_3,:At(:John,:Sf_1))$	Split
112.	$*H(W_4,:At(:John,:Sf_1))$	85
113.	$*H(W_1,:At(:John,:Sf_1))$	61
114.	$*H(W_0,:At(:John,:Sf_1))$	20
115.	$*T$	4

The second subgoal of line 89 is to show that dialing the combination of the safe in W_3 will result in the safe being open (line 99). This reduces to showing that there is some world which is the result of John dialing the combination of the safe in W_3 in which the safe is open (line 100). We split this goal into two subgoals, first trying to find a world which is the result of John dialing the combination of the safe in W_3 (line 101). Since the combination of the Sf_1 in W_3 is known to be $:C_1$, we transform the goal into showing that there is a world which is the result of John dialing $:C_1$ on Sf_1 in W_3 (line 102).

D1b says that there is such a world if $:C_1$ is a possible combination of Sf_1, if Sf_1 is a safe, and if John is at the same place as Sf_1 in W_3 (line 103). We split this goal, and first try to find a world in which $:C_1$ is the combination of Sf_1 (line 104). We know that the combination of Sf_1 in W_0 is $:C_0$, so by equality substitution we first try showing that $:C_0$ is the same as $:C_1$ (line 105). We have no rules or assertions that apply to this goal, so it fails. Next we try to show that $:C_1$ is the combination of the safe in W_1, and this succeeds (lines 106 - 108).

The second subgoal of line 103, showing that Sf_1 is a safe, is immediately satisfied by

one of the premises of the problem (lines 109 -110). The last subgoal of line 103 is to show that John is at the same place as the safe in W_3 (line 111). Since all physical conditions in W_3 are the same as in W_4, this is transformed into showing that John is at the same place as the safe in W_4 (line 112). Since W_4 is possible according to what John knows in W_1 and we are assuming that if John is at the same place as the safe he knows it, we can solve this goal by showing that John is at the same place as the safe in W_1 (line 113). A similar argument reduces the goal to showing that John is at the same place as the safe in W_0 (line 114). But this is another of the problem premises, so the goal is solved (line 115). This solves all of the subgoals of line 103, so line 102 is also solved with w_5 bound to $F_1(:John,:C_1,:Sf_1,W_3)$, which for simplicity we will abreviate W_5.

116.	$R(Do(:John,:Dial(:C_1,:Sf_1)),W_3,W_5)$	Solved
117.	$V(W_6,:Comb(:Sf_1) = :C_1$	D1a
118.	$*:Safe(:Sf_1)$	D1a
119.	$*H(W_3,:At(:John,:Sf_1))$	D1a
120.	$H(W_4,:At(:John,:Sf_1))$	85
121.	$H(W_5,int.p_1) \langle=\rangle$	D2
	$(((int.p_1 = :Open(:Sf_1)) \wedge$	
	$((V(W_3,:Comb(:Sf_1)) = :C_1) \vee H(W_3,:Open(:Sf_1)))) \vee$	
	$((int.p_1 \neq :Open(:Sf_1)) \wedge H(W_3,int.p_1)))$	
122.	$V(W_5,int.trm_1) = V(W_3,int.trm_1))$	D2
123.	$K(:John,W_5,w_3)/[W_5 \neq w_3] \langle\rightarrow\rangle$	D3
	$(\exists w_4(K(:John,W_3,w_4)/[W_3 \neq w_4] \wedge$	
	$R(:Do(:John,:Dial(:C_1,:Sf_1)),w_4,w_3)) \wedge$	
	$(H(w_3,:Open(:Sf_1)) \langle=\rangle H(W_5,:Open(:Sf_1))))$	

Since line 102 is a goal of the form $R(ev_1,w_1,w_2)$, we assert its solution (line 116). This triggers D1a to assert that $:C_1$ is a possible combination of Sf_1, that Sf_1 is a safe, and that John is at the same place as Sf_1 in W_3 (lines 117 - 119). This last assertion is transformed into the assertion that John is at the same place as the safe in W_4 (line 120). Also, D2

triggers, producing a description of the physical effects of John dialing $:C_1$ on Sf_1 in W_3 (lines 121 - 122), and D3 triggers, producing a description of the effects of the action on what John knows in W_5 (line 123).

124.	**T(W_2,Open(Sf_1))	Split
125.	**H(W_5,:Open(:Sf_1))	L
126.	**(:Open(:Sf_1) = :Open(:Sf_1)) ∧ ((V(W_3,:Comb(:Sf_1)) = :C_1) ∨ H(W_3,:Open(:Sf_1)))	121
127.	**:Open(:Sf_1) = :Open(:Sf_1)	Split
128.	*T	Eq
129.	**V(W_3,:Comb(:Sf_1)) = :C_1	Split
130.	**:C_1 = :C_1	81
131.	*T	Eq

Finally we try to satisfy the other subgoal of line 100, using the solution to the first subgoal. This gives us the goal of showing that the safe is open in W_5 (lines 124 - 125). This is a question about a simple physical condition in W_5, so line 121 is applied. Line 121 tells us that if we want to show that the safe is open in W_5, we have to show either that $:C_1$ is the combination of the safe in W_3, or that the safe was already open in W_3 (line 126). Whether the safe is open is the question we are interested in (lines 127 - 128), so we try to show that $:C_1$ is the combination of the safe in W_3 (line 129). But we already know that this is true, so the goal is satisfied (lines 130 - 131). This was the last case we had to consider, so the proof is complete.

7.5 Remarks on the Examples

Following three rather complex examples of algorithmically generated deductions involving knowledge and action, some general remarks are in order. First of all, it is rather surprising that such intuitively simple problems require such complicated deductions. It is possible that our formalism is more complicated than it should be, although since it seems

possible to come up with a fairly straightforward example which turns on any given detail, this seems unlikely. It is more probable that common-sense reasoning is more complicated than it first appears.

Although the proofs were long and complex, the deduction process was extremely well controlled with virtually no blind searching. In the last and most complicated example, of the 131 lines generated, 93 were actually necessary for the proof. In terms of percentages, 71% of the lines generated were used; only 29% were not. Another measure of the efficiency of the search is the fact that 54 formulas were transformed into other formulas or deleted, as soon as they were generated. These represent cases where the knowledge is built into the system that there is only one appropriate inference to do. Furthermore, the search space was finite. If at the last moment the proof had failed, it would have soon terminated anyway. At that point, there was only one alternative left to try, showing that the safe was already open before the dialing action took place.

This efficiency in searching for a proof was achieved by the careful structuring that went into the procedural interpretations for the axioms. The proofs were largely driven by forward chaining. In the last example, there were 75 assertions generated, but only 58 goals. Most of the assertions were not generated by blind forward chaining from the premises, however. Only 13 assertions were created in that way. The remaining assertions were triggered by the goals, either in trying to prove an implication, or by asserting the solution to a goal of the form $R(ev_1,w_1,w_2)$.

The fundamental structure of these proofs is that the goal itself triggers a large number of forward inferences which describe a structure of possible worlds, and the goal is reduced to some fairly simple backward inferences involving that structure. This enables us to tightly constrain the backward searching. Again, in the last example, while 38 of the 75 assertions generated were actually used in the proof, 57 of the 58 goals were used. This is

somewhat misleading, since the premises of the problem provided only just enough information to solve the problem, but the really complex parts of the problem involving the relations among possible worlds offered plenty of possibilities for thrashing if not treated correctly. Even the forward deductions which were not used in the proof mostly represented inferences that were reasonable to make. Many of these set up forward-chaining rules that would have been triggered if the goals had been still more complicated.

There seems to be only one way in which these methods create a possibility of generating large numbers of unnecessary inferences. That way involves assertions about what someone knows that are not required for the problem at hand. These assertions would be represented as forward-chaining rules of the form $(K(A,W_1,w_2) \rightarrow T(w_2,P_i)$. If we have a lot of information about what A knows, and hence a large number of P_i's, then whenever we want to deduce that A knows something, we will assert $K(A,W_1,W_2)$ and be inundated by assertions of the form $T(W_2,P_i)$. This problem is particularly severe in the case of axioms like A1 and RDS1, which assert something about what everyone knows. The only alternative to this within the present framework would be to represent the statements about what people know as backward-chaining rules, but this seems to be ruled out for the reasons discussed in section 6.5.

One possible way out of this problem would be to introduce a new kind of syntactic restriction into the pattern matching routine, so that the pattern $H(w_2,:P)/[K(A,W_1,w_2)]$ matches the pattern $H(W_2,:P)$ if and only if $K(A,W_1,W_2)$ is asserted. That way, if $K(A,W_1,W_2)$ is asserted, then the fact that A knows P will match the pattern $H(W_2,:P)$, without any explicit new formulas being generated. For this to work really efficiently the indexer for the data base should take these restrictions into account, so that $K(A,W_1,W_2)$ will be checked before looking at any of the assertions about what A knows. It appears that this could be

done without too much trouble. The implications of this approach need to be looked at in more detail, to take into account all the various possibilities of pattern matching, but the idea looks promising.

8. Summary and Conclusions

8.1 What has been Achieved?

In chapter 1, the goal of this thesis was stated to be the development a formalism which (i) takes into account the important role of the agent's knowledge in planning and acting and (ii) permits reasonably efficient automatic deduction. I believe that the formalism presented here achieves that goal. The most important ideas which were used in bringing this about appear to be the following:

(1) Rather than reason directly about what facts someone knows, we can gain efficiency by reasoning instead about what possible worlds are compatible with what he knows.

The first problem that we faced in reasoning about knowledge was that, while the basic facts about knowing are most easily expressed in a modal logic, there are no known techniques for efficiently searching for proofs in such logics. The solution to this problem which has been pursued in the thesis is to translate statements expressed in the modal logic of knowledge into a language which talks about possible worlds, where the reasoning can be carried out without the use of modal operators. While this idea is not original in itself, the realization that the possible-world approach could lead to more efficient proof methods than the known alternatives seems not to have been made before. The reason for this efficiency, as we pointed out in section 2.3, is that the possible-world approach permits the standard logical operators to be lifted directly into the first-order meta-language where they can be operated on using standard deduction methods. The inefficiencies that result from the lack of this feature in other approaches were analyzed in sections 2.2 and 2.6.

(2) We have worked out the details of formalizing the semantics of a fully quantified logic of knowledge and action in a first-order meta-language.

There have been other formalizations of the possible-world semantics for the propositional logic of knowledge carried out in first order-logic, but ours appears to be the first to successfully work out the important problems of handling quantifiers and equality. This extension of the previous work was essential to our overall theory of knowledge and action, which depended heavily on handling quantifying into knowledge contexts correctly.

(3) We axiomatize the syntax of the modal logic of knowledge and action and its possible-world semantics within a single first-order theory.

Rather than axiomatizing only the possible-world language for knowledge and action, we also axiomatize the interpretation of the modal logic of knowledge in that language. This allows us to state problems in the more compact and more direct modal object language, while reasoning in the possible-world meta-language, and to formulate concepts using the object language which are difficult or impossible to represent using the meta-language alone (e.g., Can). Again, this general technique was borrowed from elsewhere (McCarthy, 1975), but the idea of using the object-language to gain conceptual power seems to be new.

(4) We integrate the logic of knowledge with the logic of actions by identifying possible worlds in the logic of knowledge with situations in the logic of actions.

To integrate the logic of knowledge with a logic of actions, the logic of actions should also be expressed in terms of possible worlds. In AI the standard way of looking at an action is as a binary relation on states of the world, or situations. But this already is formally a possible-world theory of actions. We make the integration complete by identifying situations in the logic of actions with possible worlds in the logic of knowledge. This is a nonstandard interpretation of possible worlds, but it turns out to be more flexible than the usual approach and it enables us to state very easily the way actions affect what the agent knows.

(5) We analyze the knowledge preconditions for actions in terms of knowing what action to perform, and we describe the effects of actions on knowledge in terms of patterns of relationships among possible-worlds.

These are the key theoretical contributions of this thesis. Both these ideas seem to be entirely original. As shown in chapter 3, together they enable us to minimize the amount of problem-specific knowledge that must be used to make the inferences we want to make about the interaction of knowledge and action. An example of this is how the notion of a test falls out as a special case of our general theory of knowledge and action.

(6) We use domain-specific control information to help produce efficient solutions to problems.

The logic of knowledge and action used in this thesis is a complex axiomatization on an infinite domain of possible worlds. There are numerous possibilities for generating fruitless infinite searches in attempting to do automatic deductions in this formalism. By carefully controlling the way the axioms of the theory are used, we have been able to restrict the search in typical problems to a well-behaved finite space. While most of the techniques we use are not new, our sample problems are some of the most complex to which they have ever been applied, so our positive results represent encouraging evidence for the usefulness of these techniques.

These ideas solve many of the problems of reasoning about knowledge and action, but there are other questions in this area that we have left untouched. In the next section we will examine some of the limitations of the current approach, and we will conclude by trying to place this piece of work in the context of the overall goals of AI.

8.2 Limitations and Extensions of the Current Approach

The approach to reasoning about knowledge and action presented in this thesis has a number of limitations. Some of these are limitations of the logic of knowledge and action; others involve the procedural ideas for generating deductions.

One way of improving the logic of knowledge would be to make it more in agreement with "common-sense psychology", that is, make it closer to the way people usually describe the reasoning processes of others. A serious treatment of the issues of plausible reasoning raised in chapter 2 would be a major improvement. For instance, it would be nice to be able to reason that although the laws of arithmetic imply that every positive integer is the sum of four squares, if John does not know much mathematics, we shouldn't assume that he knows this fact even if he knows the laws of arithmetic. Formalising this reasoning would require, among other things, specifying what inferences are "about" mathematics.

A general problem here is that the possible-world approach makes it difficult to specify exactly what inference a person fails to make. Suppose that John knows that P, that $(P \supset Q)$, and that $(Q \supset R)$. Suppose that we also know that John is likely not to notice that R is true even if he knows that Q is true. In the possible-worlds formalism, however, the only place that we can block the inference that John knows that R is true, is going from the fact that R is true in every world which is compatible with what John knows, to the conclusion that John knows that R is true. The step that really corresponds to the inference which John fails to make is that if $(Q \supset R)$ and Q are true in every world which is compatible with what John knows, then R is true in every world which is compatible with what John knows. But this inference can not be blocked by the logic, because it is perfectly valid. One might say that the inferences that John does not do get "stacked up" while reasoning in the possible-world domain, and they all have to be cashed in, in the single step of going from what is possible according to what he knows to what he actually concludes. Formalizing this seems likely to be difficult.

We can handle some of these problems if the reasoning processes we wish to describe are expressible in terms of the procedural interpretations which we give to formulas. For example, in chapter 6 it was shown how Know(John,(P => Q)) is processed so that (P => Q) gets its usual procedural interpretation in the context of what John knows. Suppose then that the reason that John does not infer R, even though he knows P, (P ⊃ Q), and (Q ⊃ R), is that he uses (P ⊃ Q) as a backward-chaining rule, e.g. (Q <= P), and he uses (Q ⊃ R) as a forward-chaining rule, e.g. (Q => R). If this were the case, John would not be able to infer R from P, because both rules are triggered by the intermediate assertion Q, which never gets generated. We can simulate this by making the assertions Know(John,(Q <= P)) and Know(John,(Q => R)). These assertions would not generate a deduction of the goal R from the premise P, not because the logical interpretation blocks the inference, but because the procedural interpretation does. This is about as close as we can come to reasoning about what someone knows by simulating his reasoning.

This idea seems fairly promising for many applications. Ironically, where it most obviously fails is in reasoning about what someone knows about knowledge, that is, where we would have two or more nested applications of the modal operator Know. The problem is that the possible-world theory of knowledge is a much more powerful method of reasoning about what people know than the methods people themselves seem to use. That is all right when we are trying to reason about what John knows about blocks, but leads to problems when we try to reason about what John knows about what Bill knows about blocks. The difficulty is that we pretend that John also uses the possible-world theory of knowledge in reasoning about what Bill knows. By doing this we run the risk that we may credit John with much better abilities to reason about what Bill knows than John actually uses.

To be more specific, most people seem to have little trouble with the inference that if

John knows that Bill knows that (P ⊃ Q) and John knows that Bill knows that P, then John knows that Bill knows that Q. (This is, of course, a plausible inference based on the assumption that people generally know the consequences of their knowledge.) But the same assumptions that lead to this inference lead to the conclusion that if John knows that Bill knows that (P ⊃ Q) and John doesn't know that Bill doesn't know that P is false, then John doesn't know that Bill doesn't know that Q is false. This inference seems to be not obvious at all to most people, yet in the possible-world theory, it is of no greater complexity than the first inference.

The trouble is that, given that John knows (P ⊃ Q), people seem much better at reasoning that if John knows that P, then he probably knows that Q, than at reasoning that if John doesn't know that Q, then he probably doesn't know that P. The second rule is simply the contrapositive of the first, and is therefore logically equivalent to it. The second inference in the previous paragraph requires two applications of this principle, one application being to the axiom which expresses the principle. This self-application of an already difficult principle of reasoning is what makes that inference so obscure. My intuition is that if we allowed this principle to be applied where P and Q involve only nonintensional concepts (i.e. operators that are not explained in terms of possible worlds), then we would have a more reasonable model of the reasoning ability of people. I believe that this could be imposed on top of the possible-world theory of knowledge by syntactic restrictions of the type we have been using, but the details of this remain to be worked out. Probably, however, there ought to be a search for a different approach in which this restriction would be more natural.

Another problem is that simply saying that John knows that P(A) does not adequately characterize John's knowledge. It does not distinguish the case where John is able to answer the question "Is P(A) true?" from the case where he can supply A as an answer to a

request to name something that has property P. (This distinction was pointed out to me by John McCarthy.) If the property P is being the solution to some high order polynomial equation, the difference between the two is vast. The first interpretation requires only that John have some very simple knowledge of elementary algebra, so that he can plug in the proposed solution to see whether it works. The second interpretation might require John to have very sophisticated skills in algebraic manipulation. Neither the possible-world approach nor the modal logic of knowledge takes account of this distinction.

A related distinction which we might wish to draw is the difference between what someone explicitly knows and what he can deduce from his knowledge. For example, most people would explicitly know that 2 is an even number, but not that 38194604 is an even number. Most people do "know" that this number is even, however, in the sense that they can readily deduce that it is even from the fact that the last digit is 4. This distinction would certainly be part of a more detailed theory of knowledge, but ignoring it does not seem to produce any striking anomalies, as does ignoring the distinction made in the previous paragraph.

Our system also has certain limitations in its fundamental logical power. Some of these derive from basing our system on modal logic. The key fact here is that from the point of view of the object language, Know is an operator which is applied to a term denoting a possible knower and a formula which expresses a fact that he knows. An alternative approach would be to make Know a predicate which applies to a term denoting a possible knower and a term denoting a formula. With the modal logic we have to be specific about what someone knows; making Know a predicate (usually called the "syntactic" approach (Montague, 1963)) would allow much greater flexibility. For instance, currently we cannot express something like "John knows what Bill said," in the object language, because the English phrase "what Bill said" cannot be represented by a formula. If we know that Bill

said that all crows are black, we could express the fact that John knows that Bill said that all crows are black, but the more direct statement that John knows what Bill said cannot be made. With the syntactic approach, there would be no reason in principle why "what Bill said" could not be represented as a term denoting a formula.

The main reason that modal logics are generally favored over syntactic methods, however, is that there are severe difficulties in formalizing the syntactic approach. Montague (1963) has proved that syntactic treatments of modal concepts which have certain rather general (and superficially desirable) properties are in fact inconsistent. Specifically, any syntactic treatment of a modal logic is inconsistent if it has axioms corresponding to M1, M2, M4, and M5, and if it has a finite set of axioms which allow all recursive functions on names of formulas to be represented. Such theories are inconsistent because they allow the formation of self-referential, paradoxical sentences. The simplest example of this is attempting to syntactically axiomatize the notion of truth. If this is done in a theory that meets Montague's conditions, then there will be at least one term Exp_1 which dentotes the formula $\neg True(Exp_1)$. This formula, then, asserts its own falsehood. A similar, although more complex, construction can be carried out for syntactic treatments of modalities such as Know. (See Montague (1963).)

One might attempt to restrict the language so that self-referential statements cannot be formed. Kripke (1975) points out, though, that whether a statement is self-referential is often a matter of empirical fact rather than a matter of form. To take the simpler case of the predicate True, suppose that on a certain day, John makes the prediction "Everything Bill says today will be true," and says nothing else all day. If all of Bill's statements on that day have determinate truth values, then there will be no problem assigning a truth value to John's prediction. Suppose, however, that the only thing that Bill says on that day is "Everything John says today will be false." In this case, John's statement is true if and only

if Bill's statement is true, but Bill's statement is true if and only if John's statement is false. Thus we have a paradox. The point is that the paradoxical nature of these statements depends on their being taken together. Most of the time they can be used independently to make perfectly reasonable assertions. Any attempt to restrict the form of such sentences will have to rule out sentences that are all right most of the time. It seems likely that similar considerations will apply in the more complicated case of Know.

Kripke's solution to this problem is not to restrict the language, but rather to define the semantics of the language so that statements or sets of statements that are self-referential are not assigned a truth value. How this is done for True is explained in Kripke (1975). These techniques would seem to be applicable to Know as well.

Another extension to the logical power of the formalism would be to allow the system to reason about its own knowledge. This raises issues which are surprisingly quite different from reasoning about the knowledge of others. For example, if the system uses I to refer to itself and W_0 to refer to the current situation, then all true statements of the form $T(W_0,Know(I,P))$ are recursively enumerable for the system. Furthermore, if the underlying logic is decidable, all statements of the form $T(W_0,Know(I,P))$ should be decidable. If this is the case, it makes no sense to have any explicit assertions of the form $T(W_0,Know(I,P))$ or $\neg T(W_0,Know(I,P))$. Any assertion of this form will either be implied or contradicted by an assertion already implicit in the system. In fact, the most direct way to implement reasoning about the system's own knowledge is as a recursive call to the deductive routines to evaluate any expressions in this form.

Things get a little more complicated if we allow quantifying into such expressions. For instance, suppose we want to tell the system that it knows everything that has property P. This could be expressed formally as $T(W_0,All(?X1,(P(?X1) \Rightarrow Know(I,P(?X)))))$. One interpretation of this formula should be that in order to prove that P(A) is false, prove that

A is not one of the objects which the system is able to deduce has property P. (Recall that deducing that an object has property P means deducing P(B), where B is a rigid designator for the object.)

At present it is not entirely clear to me how to implement this extension to our formalism. One particular problem is that this interpretation of $T(W_0,Know(I,P))$ makes the system "non-monotonic" in Minsky's (1974) phrase. That is, we can have a theory with the property that adding axioms causes some statements that were previously theorems to be non-theorems. The preceding paragraph provides a good example of this. Suppose that A, B, and C are the only objects that we can explicitly prove have property P. Then if we know that D is not the same as A, B, or C, and we know that we know everything that has property P, we would want to be able to prove ¬P(D). But if we explicitly add P(D) as a theorem, ¬P(D) should no longer be provable. It is well known that this type of reasoning can create problems (Sandewall, 1972). For example, if P is asserted to be true whenever Q is not deducible, and Q is asserted to be true whenever P is not deducible, then to be consistent we must regard either P or Q as deducible, although we may have no basis to choose one over the other. This is reminiscent of the self-reference problems which we discussed above, and Kripke's techniques may be of use here as well.

There are also some interesting problems to be considered which relate to making deductions about lack of knowledge. For instance, as this is being written, I am quite certain that nothing that I know would tell me whether the President is sitting down at this moment. It is clear, however, that I did not come to this conclusion by exploring all the consequences of everything I know. Rather, it seems as though I have partitioned my knowledge into independent subsets, and I find there is no information in the subset that would contain statements about the President's postural position. (Once again, the credit for recognizing this problem belongs to John McCarthy.)

Some work on this type of problem is reported in an unpublished paper by Goad (1976). The basic idea is this: If we have set of formulas that "span" John's knowledge (i.e. everything he knows is derivable from those formulas), we can prove that he doesn't know P by proving that there is some possible world in which all the formulas of the knowledge set plus ¬P are true. This approach has two major difficulties. First it requires a complete description of what worlds are *a priori* possible. A potential solution to this problem is to use the techniques dicussed above to say that there is a possible world that fits a given description unless there is a proof that no such world exists. The second and more serious difficulty, though, is that Goad's proposal gives us no help with the partitioning problem. What we need is a way of saying that such-and-such is all that John knows about P, so that we restrict our attention to the relevant facts. It is not at all obvious how this notion of "about" can be captured.

In addition to the limitations of our logic of knowledge, it should be pointed out that we do not even pretend to attack the serious limitations of the situation calculus approach to describing actions. The most obvious such limitation is the inability to reason about concurrent actions. We also have avoided the problem of actions being continuous processes rather than discrete steps. A third problem would be representing action modifiers, e.g. relating dialing the combination of the safe to dialing the combination of the safe carefully or hurriedly or left-handed, etc. A really adequate logic of actions would have to solve all these problems and probably many more.

Finally, there are the limitations of the procedural techniques we have used in generating deductions. There is much less to be said about this than about the representational issues, not because there are fewer problems, but because the problems are much less well understood. One observation is that we have not presented anything like a coherent strategy for doing deductions in this domain. Instead, we looked microscopically at

individual axioms to see how they would behave if used in certain ways. This gives us no guarantee that we have not overlooked some major problem, or that if we attempt to extend the system, things will not get completely out of control. This is largely a consequence of the fact that AI has produced no real theory of how to control deductive processes, only a large number of examples of what will or will not work in particular cases. Very recently some serious work has begun on including explicit control axioms in deductive systems (McDermott, 1976) (de Kleer et al., 1977) (Doyle, 1978) (McAllester, 1978). This may provide superior ways of supplying the control information that is needed by our formalism and should be investigated further.

Two more points about these problems: First, one possible objection to the way we have embedded heuristic knowledge in the axioms of our system is that although our goal was to make any such knowledge applicable over the entire problem domain, we have actually put it into specific axioms about specific actions. Although this is true, it should be pointed out that all of the axioms describing actions were in very stereotyped forms. It does not appear to be difficult for the system to accept one of these axioms in a neutral form, recognize what type of axiom it is, and automatically add the required heuristic information.

Second, we pointed out in section 7.1 that the procedural interpretations which were given to the axioms describing actions are strongly biased towards checking the effects of a given action, as opposed to finding an action which will produce desired effects. This leaves completely open the problem of generating plans which involve the acquisition of knowledge. A major worry here is that in our formalism it might be neccessary to search an infinite set of possible worlds to do plan generation. The deductive techniques we have developed so far depend on the search space being finite. Perhaps one way out of this problem would be to propose possible plans using some weaker formalism that does not talk about possible worlds, and then test the proposed plans in the more rigorous possible-world formalism. Whatever the case, this looks like a very rich area for further research.

8.3 Conclusions

In their classic (1969) paper, McCarthy and Hayes define three standards of adequacy for representations of knowledge. The first standard is called metaphysical adequacy. A representation is metaphysically adequate if every aspect of reality has a description in terms of the representation. This seems to be what Laplace had in mind when he asserted that given the position and velocity of every particle in the universe and all the forces acting on them, he could predict exactly the future history of the universe. A modern analogue of this representation might be the quantum mechanical wave equation for the entire universe.

The trouble with representations such as these is that they cannot be used in any practical way to represent the knowledge that an intelligent being actually has about the world. Representations that can be used in this way are called epistemologically adequate. This is the standard of adequacy to which formal logic directs itself. Finally, a representation system is heuristically adequate to the extent that it can represent knowledge about how to solve problems involving those aspects of the world represented in the system.

It seems clear that epistemological and heuristic adequacy are the twin standards by which work in Artificial Intelligence must be judged. Moreover, I believe that these two goals cannot be pursued independently. Representation systems may display a remarkable degree of epistemological adequacy without there being any indication of how they can be used in a practical way to do reasoning. The modal logic of knowledge discussed in chapter 2 seems fit this description. On the other hand, heuristic methods that work for representation systems of limited descriptive power may be of little use in richer systems. As we pointed out in chapter 6, the methods used in PLANNER and related languages run into difficulties if they are applied to systems that permit incomplete descriptions of situations.

It is for these reasons that this thesis has emphasized both representational and procedural issues. We have tried to increase the range of facts that an AI system can describe, while at the same time giving the system some degree of competence in reasoning with those descriptions. Much more progress will be required before AI systems approach the common-sense reasoning abilities of humans. I hope that the research reported here is a step in that direction.

Bibliography

Brown, A. L. (1977) *Qualitative Knowledge, Causal Reasoning, and the Localization of Failures.* MIT Artificial Intelligence Laboratory, AI-TR-362.

Chang, C. and Lee, R. C. (1973) *Symbolic Logic and Mechanical Theorem Proving.* New York: Academic Press, Inc.

Cohen, P. R. (1978) *On Knowing What to Say: Planning Speech Acts.* University of Toronto, Department of Computer Science, Technical Report No. 118.

Curry, H. B. and Feys, R. (1958) *Combinatory Logic.* Amsterdam: North-Holland Publishing Company.

Doyle, J. (1978) *Truth Maintenance Systems for Problem Solving.* MIT Artificial Intelligence Laboratory, AI-TR-419.

de Kleer, J. et al. (1977) "Explicit Control of Reasoning". MIT Artificial Intelligence Laboratory, AIM-427.

Fahlman, S. E. (1973) *A Planning System for Robot Construction Tasks.* MIT Artificial Intelligence Laboratory, AI-TR-283.

Fikes, R. E. and Nilsson, N. J. (1971) "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving". *Artificial Intelligence*, 2: 189-208.

Frege, G. (1892) "On Sense and Nominatum", in H. Feigl and W. Sellars (eds.) *Readings in Philosophical Analysis*, 85-102. New York: Appleton-Century-Crofts, Inc., 1949.

Gettier, E. (1963) "Is Justified True Belief Knowledge?", in A. P. Griffiths (ed.) *Knowledge and Belief*, 144-146. London: Oxford University Press, 1967.

Goad, C. (1976) "A Formal Representation for Situations Involving Knowledge", unpublished manuscript.

Harel, D., Meyer, A. R., and Pratt, V. R. (1977) "Computability and Completeness in Logics of Programs: Preliminary Report". Proceedings of the 9th Annual ACM Symposium on Theory of Computing, 261-268.

Hayes, P. J. (1974) "Some Problems and Non-Problems in Representation Theory". *AISB Summer Conference*, University of Sussex, 63-79.

Hayes, P. J. (1978) "The Naive Physics Manifesto", unpublished manuscript.

Hoare, C. A. R. (1969) "An Axiomatic Basis for Computer Programming". *Communications of the ACM*, 12: 576-583.

Hewitt, C. (1972) *Description and Theoretical Analysis (Using Schemata) of PLANNER: a Language for Proving Theorems and Manipulating Models in a Robot.* MIT Artificial Intelligence Laboratory, AI-TR-258.

Hewitt, C. et al. (1973) "A Universal Modular ACTOR Formalism for Artificial Intelligence". *Advance Papers of the Third International Conference on Artificial Intelligence*, 235-245.

Hewitt, C. (1975) "How to Use What You Know". *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*, 189-198.

Hintikka, J. (1962) *Knowledge and Belief.* Ithica, New York: Cornell University Press.

Hintikka, J. (1969) "Semantics for Propositional Attitudes", in L. Linsky (ed.) *Reference and Modality*, 145-167. London: Oxford University Press, 1971.

Hughes, G.E. and Cresswell, M. J. (1968) *Introduction to Modal Logic.* London: Methuen and Co Ltd.

Kaplan, D. (1969) "Quantifying In", in L. Linsky (ed.) *Reference and Modality*, 112-144. London: Oxford University Press, 1971.

Kowalski, R. (1974) *Logic for Problem Solving.* Department of Computational Logic, School of Artificial Intelligence, University of Edinburgh, Memo 70.

Kripke, S. A. (1963a) "Semantical Analysis of Modal Logic", *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 9: 67-96.

Kripke, S. A. (1963b) "Semantical Considerations on Modal Logic", in L. Linsky (ed.) *Reference and Modality*, 63-72. London: Oxford University Press, 1971.

Kripke, S. A. (1972) "Naming and Necessity", in D. Davidson and G. Harmon (eds.) *Semantics of Natural Language*, 253-355. Dordrecht, Holland: D. Reidel Publishing Company.

Kripke, S. A. (1975) "Outline of a Theory of Truth". *The Journal of Philosophy*, 72: 690-716.

McAllester, D. A. (1978) "A Three Valued Truth Maintenance System". MIT Artificial Intelligence Laboratory, AIM-473.

McCarthy, J. (1962) "Towards a Mathematical Science of Computation", in C. Popplewell (ed.), *Information Processing, Proceedings of IFIP Congress 62*, 21-28. Amsterdam: North-Holland Publishing Company.

McCarthy, J. (1963) "Programs with Common Sense", in M. Minsky (ed.) *Semantic Information Processing*, 403-418. Cambridge, Mass.: The MIT Press, 1968.

McCarthy, J. and Hayes, P. J. (1969) "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in B. Meltzer and D. Michie (eds.) *Machine Intelligence 4*, 463-502. Edinburgh: Edinburgh University Press.

McCarthy, J. (1975) "An Axiomatization of Knowledge and the Example of the Wise Man Puzzle", unpublished manuscript.

McCarthy, J. (1977) "Epistemological Problems of Artificial Intelligence". *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1038-1044.

McCarthy, J. (1978) personal communication.

McCarthy, J. (1979) "First Order Theories of Individual Concepts and Propositions", in D. Michie (ed.) *Machine Intelligence 9*. Edinburgh: Edinburgh University Press.

McDermott, D. V. (1974) *Assimilation of New Information by a Natural Language Understanding System*. MIT Artificial Intelligence Laboratory, AI-TR-291.

McDermott, D. V. (1976) *Flexibility and Efficiency in a Computer Program for Designing Circuits*. MIT Artificial Intelligence Laboratory, AI-TR-402.

Minsky, (1974) "A Framework for Representing Knowledge", MIT Artificial Intelligence Laboratory, AIM-306.

Montague, R. (1963) "Syntactical Treatments of Modality, with Corollaries on Reflexion Principles and Finite Axiomatizability". *Acta Philosphica Fennica*, 16: 153-167.

Moore, R. C. (1973) "D-SCRIPT: A Computational Theory of Descriptions". *Advance Papers of the Third International Joint Conference on Artificial Intelligence*, 223-229.

Moore, R. C. (1975) *Reasoning from Incomplete Knowledge in a Procedural Deduction System*. MIT Artificial Intelligence Laboratory, AI-TR-347.

Morgan, C. G. (1976) "Methods for Automated Theorem Proving in Nonclassical Logics". *IEEE Transactions on Computers*, C-25: 852-862.

Nevins, A. J. (1974) "A Human Oriented Logic for Automatic Theorem-Proving". *Journal of the Association for Computing Machinery*, 21: 606-621.

Pratt, V. R. (1976) "Semantical Considerations on Floyd-Hoare Logic". MIT Laboratory for Computer Science, LCS-TR-168.

Pratt, V. R. (1979) "Process Logic Preliminary Report". *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages*, 93-100.

Quine, W. V. O. (1953) "Reference and Modality", in L. Linsky (ed.) *Reference and Modality*, 17-34. London: Oxford University Press, 1971.

Quine, W. V. O. (1966) "Quantifiers and Propositional Attitudes", in L. Linsky (ed.) *Reference and Modality*, 101-111. London: Oxford University Press, 1971.

Rescher, N. and A. Urquhart (1971) *Temporal Logic.* Vienna: Springer-Verlag.

Rogers, R. (1971) *Mathematical Logic and Formalized Theories.* Amsterdam: North-Holland Publishing Company.

Russell, B. (1905) "On Denoting", in H. Feigl and W. Sellars (eds.) *Readings in Philosophical Analysis*, 85-102. New York: Appleton-Century-Crofts, Inc., 1949.

Sacerdoti, E. D. (1977) *A Structure for Plans and Behavior.* New York: Elsevier North-Holland, Inc.

Sandewall, E. (1972) "An Approach to the Frame Problem, and its Implementation", in B. Meltzer and D. Michie (eds.) *Machine Intelligence 7*, 195-204. Edinburgh: Edinburgh University Press.

Sato, M. (1976) *A Study of Kripke-type Models for Some Modal Logics by Gentzen's Sequential Method.* Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan.

Smith, B. (1977) "Knowledge Representation Semantics". *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 989-990.

Stallman, R. M. and Sussman G. J. (1976) "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis". MIT Artificial Intelligence Laboratory, AIM-380.

Sussman, G. J. (1973) *A Computational Model of Skill Acquisition.* MIT Artificial Intelligence Laboratory, AI-TR-297.

Sussman, G. J. and Steele, G. L. (1975) "SCHEME: An Interpreter for Extended Lambda Calculus". MIT Artitficial Intelligence Laboratory, AIM-349.

Waldinger, R. (1975) "Achieving Several Goals Simultaneously". Stanford Research Institute, Artificial Intelligence Center, Technical Note 107.

Whitehead, A. N. and Russell, B. (1910) *Principia Mathematica.* Cambridge: Cambridge University Press.

Winograd, T. (1971) *Proceedures as a Representation for Data in a Computer Program for Understanding Natural Language.* MIT Artificial Intelligence Laboratory, AI-TR-235.

Appendix A: First-Order Axioms for Knowledge and Action

L1. $\forall p_1 (True(p_1) \equiv T(W_0,p_1))$ 80

L2. $\forall w_1,p_1,p_2 (T(w_1,And(p_1,p_2)) \equiv (T(w_1,p_1) \wedge T(w_1,p_2)))$ 80

L3. $\forall w_1,p_1,p_2 (T(w_1,Or(p_1,p_2)) \equiv (T(w_1,p_1) \vee T(w_1,p_2)))$ 80

L4. $\forall w_1,p_1,p_2 (T(w_1,(p_1 \Rightarrow p_2)) \equiv (T(w_1,p_1) \supset T(w_1,p_2)))$ 80

L5. $\forall w_1,p_1,p_2 (T(w_1,(p_1 \Leftrightarrow p_2)) \equiv (T(w_1,p_1) \equiv T(w_1,p_2)))$ 81

L6. $\forall w_1,p_1 (T(w_1,Not(p_1)) \equiv \neg T(w_1,p_1))$ 81

L7. $\forall w_1 (T(w_1,Exist(?S_i,P)) \equiv \exists s_i (T(w_1,P[@(s_i)/?S_i])))$ 90

L8. $\forall w_1 (T(w_1,All(?S_i,P)) \equiv \forall s_i (T(w_1,P[@(s_i)/?S_i])))$ 90

L9a. $\forall w_1,trm_1,...,trm_n (T(w_1,P(trm_1,...,trm_n)) \equiv H(w_1,:P(D(w_1,trm_1),...,D(w_1,trm_n))))$ 93
if P is not an essential property of the things it is true of.

L9b. $\forall w_1,trm_1,...,trm_n (T(w_1,P(trm_1,...,trm_n)) \equiv :P(D(w_1,trm_1),...,D(w_1,trm_n)))$ 93
if P is an essential property of the things it is true of.

L10. $\forall w_1,x_1 (D(w_1,@(x_1)) \equiv x_1)$ 94

L11a. $\forall w_1 (D(w_1,Cnst) \equiv V(w_1,:Cnst))$ if Cnst is not a rigid designator. 94

L11b. $\forall w_1 (D(w_1,Cnst) \equiv :Cnst)$ if Cnst is a rigid designator. 94

L12a. $\forall w_1,trm_1,...,trm_n (D(w_1,F(trm_1,...,trm_n)) \equiv V(w_1,:F(D(w_1,trm_1),...,D(w_1,trm_n))))$ 95
if F is not a rigid function.

L12b. $\forall w_1,trm_1,...,trm_n (D(w_1,F(trm_1,...,trm_n)) \equiv :F(D(w_1,trm_1),...,D(w_1,trm_n)))$ 95
if F is a rigid function.

L13. $\forall w_1,trm_1,trm_2 (T(w_1,Eq(trm_1,trm_2)) \equiv (D(w_1,trm_1) \equiv D(w_1,trm_2)))$ 95

K1. $\forall w_1,trm.a_1,p_1 (T(w_1,Know(trm.a_1,p_1)) \equiv \forall w_2 (K(D(w_1,trm.a_1),w_1,w_2) \supset T(w_2,p_1)))$ 81

K2. $\forall a_1,w_1 (K(a_1,w_1,w_1))$ 81

K3. $\forall a_1,w_1,w_2 (K(a_1,w_1,w_2) \supset \forall w_3 (K(a_1,w_2,w_3) \supset K(a_1,w_1,w_3)))$ 82

R1. $\forall w_1,trm.ev_1,p_1$ 101
$(T(w_1,Res(trm.ev_1,p_1)) \equiv \exists w_2 (R(D(w_1,trm.ev_1),w_1,w_2) \wedge T(w_2,p_1)))$

R2. $\forall w_1,trm.a_1,trm.act_1,trm.act_2,p_1$ 103
$(T(w_1,Res(Do(trm.a_1,(trm.act_1; trm.act_2)),p_1)) \equiv$
$T(w_1,Res(Do(trm.a_1,trm.act_1),Res(Do(@(D(w_1,trm.a_1)),trm.act_2),p_1))))$

R3. $\forall w_1,trm.a_1,trm.act_1,trm.act_2,p_1,p_2$ 103
$(T(w_1,Res(Do(trm.a_1,If(p_1,trm.act_1,trm.act_2)),p_2)) \equiv$
$((T(w_1,p_1) \wedge T(w_1,Res(Do(trm.a_1,trm.act_1),p_2))) \vee$
$(\neg T(w_1,p_1) \wedge T(w_1,Res(Do(trm.a_1,trm.act_2),p_2)))))$

R4. $\forall w_1,trm.a_1,trm.act_1,p_1,p_2$ 103
$(T(w_1,Res(Do(trm.a_1,While(p_1,trm.act_1)),p_2)) \equiv$
$T(w_1,Res(Do(trm.a_1,If(p_1,(trm.act_1; While(p_1,trm.act_1)),Nil)),p_2)))$

R5. $\forall trm.a_1,w_1,w_2(R(Do(trm.a_1,Nil),w_1,w_2) \equiv (w_1 = w_2))$ 103

R6. $\forall w_1,trm.ev_1,p_1$ 104
$(T(w_1,Resl(trm.ev_1,p_1)) \equiv \forall w_2(R(D(w_1,trm.ev_1),w_1,w_2) \supset T(w_2,p_1)))$

P1. $\forall a_1,x_1,x_2,w_1,w_2$ 105
$(\exists w_2(R(:Do(a_1,:Puton(x_1,x_2)),w_1,w_2)) \equiv$
$((:Block(x_1) \wedge \forall x_3(\neg H(w_1,:On(x_3,x_1))) \wedge$
$((x_1 \neq x_2) \wedge \forall x_3(\neg H(w_1,:On(x_3,x_2)))) \vee :Table(x_2))))$

P2. $\forall a_1,x_1,x_2,w_1,w_2$ 105
$(R(:Do(a_1,:Puton(x_1,x_2)),w_1,w_2) \supset$
$(H(w_2,:On(x_1,x_2)) \wedge \forall x_3((x_2 \neq x_3) \supset \neg H(w_2,:On(x_1,x_3)))))$

P3. $\forall a_1,x_1,x_2,w_1,w_2$ 105
$(R(:Do(a_1,:Puton(x_1,x_2)),w_1,w_2) \supset$
$(\forall int.trm_1(V(w_1,int.trm_1) \equiv V(w_2,int.trm_1)) \wedge$
$\forall int.p_1(\forall x_3(int.p_1 \neq :On(x_1,x_3)) \supset (H(w_1,int.p_1) \equiv H(w_2,int.p_1)))))$

P4. $\forall a_1,x_1,x_2,w_1,w_2$ 118
$(R(:Do(a_1,:Puton(x_1,x_2)),w_1,w_2) \supset$
$\forall w_3(K(a_1,w_2,w_3) \equiv \exists w_4(K(a_1,w_1,w_4) \wedge R(:Do(a_1,:Puton(x_1,x_2)),w_4,w_3))))$

C1. $\forall w_1,trm.a_1,trm.act_1,p_1$ 112
$(T(w_1,Know(trm.a_1,And(Eq(@(D(w_1,trm.act_1)),trm.act_1),$
$Res(Do(@(D(w_1,trm.a_1)),trm.act_1),p_1)))) \supset$
$T(w_1,Can(trm.a_1,trm.act_1,p_1)))$

C2. $\forall w_1,trm.a_1,trm.act_1,trm.act_2,p_1$ 112
$(T(w_1,Can(trm.a_1,(trm.act_1; trm.act_2),p_1)) \equiv$
$T(w_1,Can(trm.a_1,trm.act_1,Can(@(D(w_1,trm.a_1)),trm.act_2,p_1))))$

C3. $\forall w_1, trm.a_1, trm.act_1, trm.act_2, p_1, p_2$ 112
$(T(w_1, Can(trm.a_1, If(p_1, trm.act_1, trm.act_2), p_2)) \equiv$
$((T(w_1, Know(trm.a_1, p_1)) \wedge T(w_1, Can(trm.a_1, trm.act_1, p_2))) \vee$
$(T(w_1, Know(trm.a_1, Not(p_1))) \wedge T(w_1, Can(trm.a_1, trm.act_2, p_2)))))$

C4. $\forall w_1, trm.a_1, trm.act_1, p_1, p_2$ 112
$(T(w_1, Can(trm.a_1, While(p_1, trm.act_1), p_2)) \equiv$
$T(w_1, Can(trm.a_1, If(p_1, (trm.act_1; While(p_1, tr.act_1)), Nil), p_2)))$

D1. $\forall a_1, x_1, x_2, w_1$ 113
$(\exists w_2(R(:Do(a_1, :Dial(x_1, x_2)), w_1, w_2)) \equiv$
$(\exists w_3(x_1 = V(w_3, :Comb(x_2))) \wedge :Safe(x_2) \wedge H(w_1, :At(a_1, X_2))))$

D2. $\forall a_1, x_1, x_2, w_1, w_2$ 113
$R(:Do(a_1, :Dial(x_1, x_2)), w_1, w_2) \supset$
$(((x_1 = V(w_1, :Comb(x_2))) \supset H(w_2, :Open(x_2))) \wedge$
$(((x_1 \neq V(w_1, :Comb(x_2))) \wedge \neg H(w_1, :Open(x_2))) \supset \neg H(w_2, :Open(x_2))) \wedge$
$(H(w_1, :Open(x_2)) \supset H(w_2, :Open(x_2)))))$

D3. $\forall a_1, x_1, x_2, w_1, w_2$ 120
$(R(:Do(a_1, :Dial(x_1, x_2)), w_1, w_2) \supset$
$\forall w_3(K(a_1, w_2, w_3) \equiv ((H(w_2, :Open(x_2)) \equiv H(w_3, :Open(x_2))) \wedge$
$\exists w_4(K(a_1, w_1, w_4) \wedge R(:Do(a_1, :Dial(x_1, x_2)), w_4, w_3)))))$

D4. $\forall a_1, x_1, x_2, w_1, w_2$ 122
$(R(:Do(a_1, :Dial(x_1, x_2)), w_1, w_2) \supset$
$(\forall int.trm_1(V(w_1, int.trm_1) = V(w_2, int.trm_1)) \wedge$
$\forall int.p_1((int.p_1 \neq :Open(x_2)) \supset (H(w_1, int.p_1) \equiv H(w_2, int.p_1)))))$

ABV1. $\forall w_1, trm.x_1, trm.x_2$ 107
$(T(w_1, Above(trm.x_1, trm.x_2)) \equiv$
$(T(w_1, On(trm.x_1, trm.x_2)) \vee$
$\exists x_3(T(w_1, Above(trm.x_1, @(x_3))) \wedge T(w_1, Above(@(x_3), trm.x_2)))))$

A1. $\forall w_1, a_1, x_1(H(w_1, :At(a_1, x_1)) \supset \forall w_2(K(a_1, w_1, w_2) \supset H(w_2, :At(a_1, x_1))))$ 114

INF1. $\forall w_1, trm.x_1, exp_1$ 127
$(T(w_1, Info(trm.x_1, exp_1)) \equiv (exp_1 = V(w_1, :Info(D(w_1, trm.x_1)))))$

RD1. $\forall a_1, x_1, w_1, w_2$ 127
$(\exists w_2(R(:Do(a_1, :Read(x_1)), w_1, w_2)) \equiv$
$(H(w_1, :Reads(a_1)) \wedge H(w_1, :At(a_1, x_1))))$

RDS1. $\forall w_1, a_1(H(w_1, :Reads(a_1)) \supset \forall w_2(K(a_1, w_1, w_2) \supset H(w_2, :Reads(a_1))))$ 127

RD2. $\forall a_1,x_1,w_1,w_2$ 127

$(R(:Do(a_1,:Read(x_1)),w_1,w_2) \supset$

$\forall w_3(K(a_1,w_2,w_3) \equiv ((V(w_2,:info(x_1)) = V(w_3,:info(x_1))) \wedge$

$\exists w_4(K(a_1,w_1,w_4) \wedge R(:Do(a_1,:Read(x_1)),w_4,w_3)))))$

RD3. $\forall a_1,x_1,w_1,w_2$ 127

$(R(:Do(a_1,:Read(x_1)),w_1,w_2) \supset$

$(\forall int.trm_1 (V(w_1,int.trm_1) = V(w_2,int.trm_1)) \wedge$

$\forall int.p_1 (H(w_1,int.p_1) = H(w_2,int.p_1))))$

Appendix B: Procedurally Interpreted Axioms for Knowledge and Action

L1. $True(p_1) \Leftrightarrow T(W_0,p_1)$ 156

L2. $T(w_1,(And(p_1,p_2)) \Leftrightarrow (T(w_1,p_1) \wedge T(w_1,p_2))$ 156

L3. $T(w_1,(Or(p_1,p_2))) \Leftrightarrow (T(w_1,p_1) \vee T(w_1,p_2))$ 156

L4a. $T(w_1,(p_1 \rightarrow p_2)) \Leftrightarrow (T(w_1,p_1) \rightarrow T(w_1,p_2))$ 156

L4b. $T(w_1,(p_1 \leftarrow p_2)) \Leftrightarrow (T(w_1,p_1) \leftarrow T(w_1,p_2))$ 156

L4c. $T(w_1,(p_1 \Rightarrow p_2)) \Leftrightarrow (T(w_1,p_1) \Rightarrow T(w_1,p_2))$ 156

L4d. $T(w_1,(p_1 \Leftarrow p_2)) \Leftrightarrow (T(w_1,p_1) \Leftarrow T(w_1,p_2))$ 156

L5a. $T(w_1,(p_1 \Leftrightarrow p_2)) \Leftrightarrow (T(w_1,p_1) \Leftrightarrow T(w_1,p_2))$ 156

L5b. $T(w_1,(p_1 \Leftrightarrow p_2)) \Leftrightarrow (T(w_1,p_1) \Leftrightarrow T(w_1,p_2))$ 156

L6. $T(w_1,Not(p_1)) \Leftrightarrow \neg T(w_1,p_1)$ 156

L7. $T(w_1,Exist(?S_i,P)) \Leftrightarrow \exists s_i(T(w_1,P[@(s_i)/?S_i]))$ 156

L8. $T(w_1,All(?S_i,P)) \Leftrightarrow \forall s_i(T(w_1,P[@(s_i)/?S_i]))$ 156

L9a. $T(w_1,P(trm_1,...,trm_n)) \Leftrightarrow H(w_1,:P(D(w_1,trm_1),...,D(w_1,trm_n)))$ 156
if P is not an essential property of the things it is true of.

L9b. $T(w_1,P(trm_1,...,trm_n))] \Leftrightarrow :P(D(w_1,trm_1),...,D(w_1,trm_n))$ 156
if P is an essential property of the things it is true of.

L10a. $D(w_1,@(x_1)) = x_1$ 156

L10b. $(@(x_1) = @(x_2)) \Leftrightarrow (x_1 = x_2)$ 156

L11a. $D(w_1,Cnst) = V(w_1,:Cnst)$ if Cnst is not a rigid designator. 156

L11b. $D(w_1,Cnst)] = :Cnst$ if Cnst is a rigid designator. 156

L12a. $D(w_1,F(trm_1,...,trm_n)) = V(w_1,:F(D(w_1,trm_1),...,D(w_1,trm_n))$ 156
if F is not a rigid function.

L12b. $D(w_1,F(trm_1,...,trm_n)) = :F(D(w_1,trm_1),...,D(w_1,trm_n))$ 157
if F is a rigid function.

L13. $T(w_1,Eq(trm_1,trm_2)) \langle\Rightarrow\rangle (D(w_1,trm_1) = D(w_1,trm_2))$ 157

K1. $T(w_1,Know(trm.a_1,p_1)) \langle\Rightarrow\rangle \forall w_2(K(D(w_1,trm.a_1),w_1,w_2) \rightarrow T(w_2,p_1))$ 157

K2. $K(a_1,w_1,w_1)$ 157

K3. $K(a_1,w_1,w_2)/[w_1 \neq w_2] \rightarrow (K(a_1,w_2,w_3)/[w_2 \neq w_3] \rightarrow K(a_1,w_1,w_3))$ 157

R1. $T(w_1,Res(trm.ev_1,p_1))/$ 169
$[(trm.ev_1 \neq Do(trm.a_1,(trm.act_2; trm.act_3))) \wedge$
$(trm.ev_1 \neq Do(trm.a_1,If(p_2,trm.act_2,trm.act_3))) \wedge$
$(trm.ev_1 \neq Do(trm.a_1,While(p_2,trm.act_2)))] \langle\Rightarrow\rangle$
$\exists w_2(R(D(w_1,Do(trm.a_1,trm.act_1)),w_1,w_2) \wedge T(w_2,p_1))$

R2. $T(w_1,Res(Do(trm.a_1,(trm.act_1; trm.act_2)),p_1)) \langle\Rightarrow\rangle$ 169
$T(w_1,Res(Do(trm.a_1,trm.act_1),Res(Do(@(D(w_1,trm.a_1)),trm.act_2),p_1)))$

R3. $T(w_1,Res(Do(trm.a_1,If(p_1,trm.act_1,trm.act_2)),p_2)) \langle\Rightarrow\rangle$ 169
$((T(w_1,p_1) \wedge T(w_1,Res(Do(trm.a_1,trm.act_1),p_2))) \vee$
$(\neg T(w_1,p_1) \wedge T(w_1,Res(Do(trm.a_1,trm.act_2),p_2))))$

R4. $T(w_1,Res(Do(trm.a_1,While(p_1,trm.act_1)),p_2)) \langle\Rightarrow\rangle$ 169
$T(w_1,Res(Do(trm.a_1,If(p_1,(trm.act_1; While(p_1,trm.act_1)),Nil)),p_2))$

R5. $R(Do(trm.a_1,Nil),w_1,w_2) \langle\Rightarrow\rangle (w_1 = w_2)$ 169

R6. $T(w_1,Res1(trm.ev_1,p_1))/$ 170
$[(trm.ev_1 \neq Do(trm.a_1,(trm.act_2; trm.act_3))) \wedge$
$(trm.ev_1 \neq Do(trm.a_1,If(p_2,trm.act_2,trm.act_3))) \wedge$
$(trm.ev_1 \neq Do(trm.a_1,While(p_2,trm.act_2)))] \langle\Rightarrow\rangle$
$\forall w_2(R(D(w_1,Do(trm.a_1,trm.act_1)),w_1,w_2) \rightarrow T(w_2,p_1))$

C1. $T(w_1,Can(trm.a_1,trm.act_1,p_1))/$ 171
$[(trm.act_1 \neq (trm.act_2; trm.act_3)) \wedge$
$(trm.act_1 \neq If(p_2,trm.act_2,trm.act_3)) \wedge$
$(trm.act_1 \neq While(p_2,trm.act_2))] \langle=$
$T(w_1,Know(trm.a_1,And(Eq(@(D(w_1,trm.act_1)),trm.act_1),$
$Res(Do(@(D(w_1,trm.a_1)),trm.act_1),p_1))))]$

C2. $T(w_1,Can(trm.a_1,(trm.act_1; trm.act_2),p_1)) \langle\Rightarrow\rangle$ 171
$T(w_1,Can(trm.a_1,trm.act_1,Can(@(D(w_1,trm.a_1)),trm.act_2,p_1)))$

C3. $T(w_1,Can(trm.a_1,If(p_1,trm.act_1,trm.act_2),p_2)) \langle\Rightarrow\rangle$ 171
$((T(w_1,p_1) \wedge T(w_1,Can(trm.a_1,trm.act_1,p_2))) \vee$
$(\neg T(w_1,p_1) \wedge T(w_1,Can(trm.a_1,trm.act_2,p_2))))$

C4. $T(w_1,Can(trm.a_1,While(p_1,tr.act_1),p_2)) \langle=\rangle$ 171
$T(w_1,Can(trm.a_1,If(p_1,(trm.act_1; While(p_1,tr.act_1)),Nil),p_2))$

D1a. $R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \Rightarrow$ 172
$(\exists w_3(V(w_3,:Comb(x_2)) = x_1) \wedge :Safe(x_2) \wedge H(w_1,:At(a_1,x_2)))$

D1b. $R(:Do(a_1,:Dial(x_1,x_2)),w_1,F_1(a_1,x_1,x_2,w_1)) \Leftarrow$ 172
$((V(w_3,:Comb(x_2) = x_1) \wedge :Safe(x_2) \wedge H(w_1,:At(a_1,x_2)))$

D2. $R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \rightarrow$ 172
$((H(w_2,int.p_1) \langle=\rangle$
$(((int.p_1 = :Open(x_2)) \wedge$
$((V(w_1,:Comb(x_2)) = x_1) \vee H(w_1,:Open(x_2)))) \vee$
$((int.p_1 \neq :Open(x_2)) \wedge H(w_1,int.p_1)))) \wedge$
$(V(w_2,int.trm_1) = V(w_1,int.trm_1)))$

D3. $R(:Do(a_1,:Dial(x_1,x_2)),w_1,w_2) \rightarrow$ 173
$(K(a_1,w_2,w_3)/[w_2 \neq w_3] \langle\rightarrow\rangle$
$(\exists w_4(K(a_1,w_1,w_4)/[w_1 \neq w_4] \wedge$
$R(:Do(a_1,:Dial(x_1,x_2)),w_4,w_3)) \wedge$
$(H(w_2,:Open(x_2)) \langle=\rangle H(w_3,:Open(x_2)))))$

A1. $K(a_1,w_1,w_2)/[w_1 \neq w_2] \rightarrow$ 180
$(H(w_2,:At(a_1,x_1)) \vee \neg H(w_1,:At(a_1,x_1)))$

INF1. $T(w_1,Info(trm.x_1,exp_1)) \langle=\rangle (V(w_1,:Info(D(w_1,trm.x_1))) = exp_1)$ 191

RDS1. $K(a_1,w_1,w_2)/[w_1 \neq w_2] \rightarrow$ 191
$(H(w_2,:Reads(a_1)) \vee \neg H(w_1,:Reads(a_1)))$

RD1a. $R(:Do(a_1,:Read(x_1)),w_1,w_2) \Rightarrow$ 191
$(H(w_1,:Reads(a_1)) \wedge H(w_1,:At(a_1,x_1)))$

RD1b. $R(:Do(a_1,:Read(x_1)),w_1,F_2(a_1,x_1,w_1)) \Leftarrow$ 192
$(H(w_1,:Reads(a_1)) \wedge H(w_1,:At(a_1,x_1)))$

RD2. $R(:Do(a_1,:Read(x_1)),w_1,w_2) \rightarrow$ 192
$(K(a_1,w_2,w_3)/[w_2 \neq w_3] \langle\rightarrow\rangle$
$\exists w_4(K(a_1,w_1,w_4)/[w_1 \neq w_4] \wedge$
$(V(w_4,:Info(x_1)) = V(w_1,:Info(x_1))) \wedge$
$R(:Do(a_1,:Read(x_1)),w_4,w_3)))$

RD3. $R(:Do(a_1,:Read(x_1)),w_1,w_2) \rightarrow$ 192
$((V(w_2,int.trm_1) = V(w_1,int.trm_1)) \wedge$
$(H(w_2,int.p_1) \langle=\rangle H(w_1,int.p_1))$


4-
DT