October 1993

Search Plans

Michael B. Moore
University of Pennsylvania

Search Plans

Abstract
People often do not know where things are and have to look for them. This thesis presents a formal model suitable for reasoning about how to find things and acting to find them, which I will call "search behavior". Since not knowing location of something can prevent an agent from reaching his desired goal, the ability to plan and conduct a search will be argued to increase the variety of situations in which an agent can succeed at his chosen task.

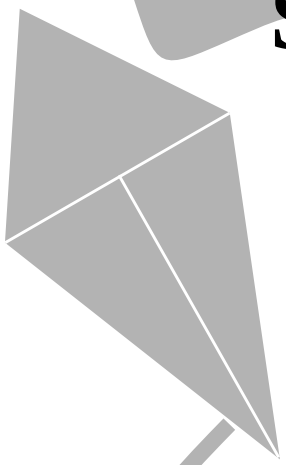
Searching for things is a natural problem that arises when the blocks world assumptions (which have been the problem setting for most planning research) are modified by providing the agent only *partial* knowledge of his environment. Since the agent does not know the total world state, actions may *appear* to have nondeterministic effects. The significant aspects of the search problem which differ from previously studied planning problems are the acquisition of information and iteration of similar actions while exploring a search space.

Since introduction of the situation calculus [MH69], various systems have been proposed for *representing and reasoning* about actions which involve knowledge acquisition and iteration, including Moore's work on the interaction between knowledge and action [Moo80]. Such systems can be used to infer properties of plans which have already been constructed, but do not themselves *construct plans* for complex actions. My concern with searching has to do with a sense that Moore's knowledge preconditions are overly restrictive. Morgenstern [Mor88] examined ways to weaken knowledge preconditions for an individual agent by relying on the knowledge and abilities of other agents. Lesperance's research [Les91] on indexical knowledge is another way of weakening the knowledge preconditions. I am trying to reduce the *amount* of information an agent must know (provided he can search a known search space). If you dial the right combination to a safe it will open, whether or not you knew in advance that it *was* the right combination. Search is a way to guarantee you will eventually dial the right combination. So what I am exploring is how to systematically construct a search that will use available knowledge to accomplish something the agent does not currently know enough to do directly.

I claim it is possible for automated agents to engage in search behavior. Engaging in search behavior consists in recognizing the need for a search, constructing an effective plan, and then carrying out that plan. Expressing such a plan and reasoning about its effectiveness requires a representation language. I will select a representation language based on criteria derived from analyzing the search planning problem. Each of the three components of a system for engaging in search behavior will be designed and implemented to demonstrate that an automated agent can find things when he needs to.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-93-29.

The Institute For Research In Cognitive Science



Search Plans
(Dissertation Proposal)

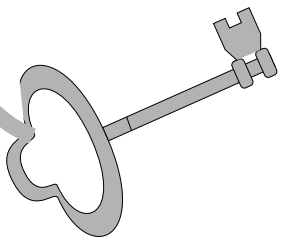
by

Michael B. Moore

University of Pennsylvania
Philadelphia, PA 19104-6228

October 1993

Site of the NSF Science and Technology Center for Research in Cognitive Science



University of Pennsylvania
Founded by Benjamin Franklin in 1740

IRCS Report 93-29

Search Plans

Michael B. Moore[1]

July 8, 1993

[1]This research was supported by the following grants: DARPA no. N00014-90-J-1863, ARO no. DAAL 03-89-C-0031, NSF no. IRI 90-16592, and Ben Franklin no. 91S.3078C-1.

A DISSERTATION PROPOSAL
in

COMPUTER AND INFORMATION SCIENCE

July 8, 1993

Abstract

People often do not know where things are and have to look for them. This thesis presents a formal model suitable for reasoning about how to find things and acting to find them, which I will call "search behavior". Since not knowing location of something can prevent an agent from reaching his desired goal, the ability to plan and conduct a search will be argued to increase the variety of situations in which an agent can succeed at his chosen task.

Searching for things is a natural problem that arises when the blocks world assumptions (which have been the problem setting for most planning research) are modified by providing the agent only *partial* knowledge of his environment. Since the agent does not know the total world state, actions may *appear* to have nondeterministic effects. The significant aspects of the search problem which differ from previously studied planning problems are the acquisition of information and iteration of similar actions while exploring a search space.

Since introduction of the situation calculus [MH69], various systems have been proposed for *representing* and *reasoning* about actions which involve knowledge acquisition and iteration, including Moore's work on the interaction between knowledge and action [Moo80]. Such systems can be used to infer properties of plans which have already been constructed, but do not themselves *construct plans* for complex actions. My concern with searching has to do with a sense that Moore's knowledge preconditions are overly restrictive. Morgenstern [Mor88] examined ways to weaken knowledge preconditions for an individual agent by relying on the knowledge and abilities of other agents. Lesperance's research [Les91] on indexical knowledge is another way of weakening the knowledge preconditions. I am trying to reduce the *amount* of information an agent must know (provided he can search a known search space). If you dial the right combination to a safe it will open, whether or not you knew in advance that it *was* the right combination. Search is a way to guarantee you will eventually dial the right combination. So what I am exploring is how to systematically construct a search that will use available knowledge to accomplish something the agent does not currently know enough to do directly.

I claim it is possible for automated agents to engage in search behavior. Engaging in search behavior consists in recognizing the need for a search, constructing an effective plan, and then carrying out that plan. Expressing such a plan and reasoning about its effectiveness requires a representation language. I will select a representation language based on criteria derived from analyzing the search planning problem. Each of the three components of a system for engaging in search behavior will be designed and implemented to demonstrate that an automated agent can find things when he needs to.

Acknowledgement

I owe debts of gratitude to many teachers and colleagues who have helped shape this research. I especially appreciate the help of my advisor Bonnie Lynn Webber and fellow students who have contributed to the AnimNL project, including: Chris Geib, Breck Baldwin, Libby Levison, Mike White, and Barbara Di Eugenio. This work has benefited from discussion with other members of Penn's Computational Linguistics Feedback Forum (CLiFF) and our informal planning discussion group, including Charlie Ortiz and Ron Rymon. Finally, the comments and advice of my dissertation committee, Henry Kautz (of AT&T), Norm Badler, Dale Miller, and Greg Provan (all of Penn), have been invaluable in helping me clarify the direction in which this research will evolve.

Contents

List of Tables

List of Figures

Chapter 1

Finding something

People often do not know where things are and have to look for them. Or they are given a generic description of where something is and have to look for it. Even if robots, like elephants, never forget, our ways of communicating locations are such that robots instructed by people are going to have to look for things.

This thesis presents a formal model suitable for reasoning about how to find things and acting to find them, which I will call "search behavior". Since not knowing something's location can prevent agents from reaching their desired goals, the ability to plan and conduct a search will be argued to increase the variety of situations in which an agent can succeed at his chosen task.

Search is concerned with increasing an agent's awareness of his environment – the agent acts to acquire knowledge of the location of an object. Conducting the search may also require changing the state of the world, including agents moving themselves, or removing various obstacles to their perception. Thus in search, two kinds of state information must be represented and monitored: the agent's internal epistemic state and the external environment state.

My research on search plans differs from the considerable work that has been done on efficient algorithms for search problems (see for example [Knu68]) in that it is aimed at identifying the *reasoning process* required of an automated agent to decide to conduct a search and to monitor its progress. In contrast, work on search algorithms aims to achieve efficiency by compiling out the reasoning involved with monitoring a search and leave to the programmer the decision of *when* to search. Planning has often been construed as a search problem [Kor87]. Here I construe physical search – that is, the process of looking for things – as a planning problem.

1.1 The search planning problem

The research problem I am addressing is a planning problem. An agent has a goal to achieve in an environment. The agent is able to perform several different actions and has information about the effects of different actions on the environment.

The original planning problem was to discover a sequence of actions which result in the goal being satisfied when executed in the current environment. The planning problem for *search* will require plans which have more complex structure than sequential ordering

(as will be argued in this chapter). Therefore, the agent is provided with additional plan construction operators for forming sequential, conditional, or iterated plans.

Unlike the original planning problem, search involves discovering information about the environment. The notion of environment must be enriched to distinguish what information is known to the agent at any given time and what is unknown.

In its most general form, the search planning problem can be stated as discovering a plan structure which will determine the truth (or falsity) of a goal proposition when executed in the current environment. The search planning problem can also be viewed as discovering a plan which will determine the identity of an object in the environment which satisfies a goal property when executed in the current environment.

The rest of this chapter discusses these novel aspects of the search problem in more detail. The requirement for additional control structure in plans is derived from attempts to use existing planning techniques to solve the search planning problem.

One example of the search planning problem which is used throughout this proposal to illustrate search behavior is an environment with a variety of containers which hide a ball from the agent. The agent has the goal of locating the ball. In this example, the containers provide the environmental device which is the basis for distinguishing what the agent knows about the world from what it does not know.

To give the reader a sense of the class of search problems, a variety of other examples are presented.

An *area search* uses the device of sweeping a limited perceptual sensor over an area of the environment (larger than the area immediately perceivable to the agent) to explore a search space. An illustration of area search is the behavior that a Coast Guard Search and Rescue mission might undertake to scan a large area of ocean for missing boaters.

A kind of search which might be called a *theory search* might be used by an agent attempting to answer the question, "Is it feasible for me to buy a house?" Information which the agent can use to develop a theory for answering this question is initially hidden from the agent in books. Reading various texts may provide direct answers to the goal proposition, indirect answers which may be used to infer the truth of the goal proposition, or secondary information about other texts.

An agent may be faced with a search problem when in a maze. The goal proposition might be, "Can I exit this maze?" The walls immediately surrounding the agent in this environment hide information about the rest of the maze. Similarly, when searching for an object in a cluttered environment, obstacles other than containers may block the agent's perception.

The rest of this proposal will focus on searches where the goal is to determine the location of an object. Most examples will involve searching through containers for a ball, which is one of the simplest kinds of search problems. The advantage of this instance of the search problem is that geometric reasoning can be simplified if we consider that an agent can see any object that is not inside a closed container. The dissertation will present a general mechanism which can plan for all the above kinds of search.

1.2 Novel aspects of the search planning problem

From the perspective of planning, search plans constitute another step in the evolution of plans from simple sequences of actions to structures which more closely resemble computer programs. The planning problem for search differs from earlier planning problems. The earliest recognition of the distinct character of plans for search in the cognitive science literature is in Miller *et al.* [MGP60].

The original planning problem assumed certain properties about the behavior of the environment in which an agent was acting and the agent's relation to that environment. These assumptions are called the *blocks world assumptions*.

- Only one agent is active in the environment – nothing else causes change.
- That agent has total knowledge of the environment.
- Actions are deterministic in their effects

Searching for things is a natural problem that arises when the blocks world assumptions are modified by providing the agent only *partial* knowledge of his environment.

Rather than relax the requirement that actions have deterministic effects, I consider the case that the agent may not have total knowledge of the environment. Since the agent does not know the total world state, actions may *appear* to have nondeterministic effects. There is a subtle difference between these two planning problems. Conditional planning is an appropriate response to nondeterminism, but something more is necessary for epistemic nondeterminism.

1.3 A scenario calling for a search plan

To introduce the notions of search knowledge and search plans, consider an agent who knows that one effect of opening a container is to reveal things inside that container. The agent might use this information to find a ball, b, that unbeknownst to him is hidden in some container c, by reasoning as follows.

> I do not currently see the ball, but I do know of an action which will make the ball visible, *opening the container it is in*. The obvious container to open is the one containing the ball, but the reason I am searching is because I do not know what container that is. Perhaps I have an independent way to know where the ball is, e.g. by shaking the container it is in. This is no solution, since I still do not know which container to shake.
>
> If I now see seven containers, then I may consider that either it is in the first, the second, ..., or the seventh. Each of these things I am able to assume in turn. Acting on those assumptions will exhaustively search the space of containers and will find the ball. Fortunately, the nature of this problem is that opening an empty container is useful as it permits me to eliminate the possibility that the ball is there.
>
> Whenever a new container is discovered inside an old one, that discovery can be easily accommodated since I will do the same thing to check inside it as with all the other containers.

The agent knows the disjunctive proposition that the ball is in the first container, or the second, ..., or the seventh. That information is insufficient for the agent to know he can find the ball by opening the first container (or any of the other containers). Section 3.2 presents this formal characterization of ability. However, it is sufficient for the agent to know that the effect of conducting a search through those containers is to know the location of the ball (or know that the ball is not located in any of the containers).

1.4 Previous approaches to related problems

The relationship between an agent and his environment determines the kind of plans he will need to form in order to act effectively. Initial planning research assumed that the agent was acting in a blocks world. This section considers planning research which shares characteristics with mine. None of these research directions adequately addresses the problem of planning to conduct a search.

1.4.1 Classical planning

Search is a purposeful and conscious activity in which agents engage. A major research theme in Artificial Intelligence (AI) is that purposeful behavior can be guided by deliberate plans. The prominent paradigm for such planned activity is planning as a state space search, with the resulting plan being subsequently carried out by the agent. This paradigm has its origins in the situation calculus of McCarthy [MH69] and was originally implemented as a planner in STRIPS by Fikes and Nilsson [FN71].

State space search planning considers an initial state and a goal state. The goal state describes some future time when the purpose of the plan has been achieved. The initial state describes the way the world is now. A planner searches for a sequence of actions which will get the agent from the initial state to the goal state. Following the development of a plan to traverse a path from the current world state to the desired goal state, the plan is executed. Because agents have total knowledge of their environment in this planning problem, the need for conducting a search never arises.

A sequential planner might be used as *part of a system* to produce search behavior. Let an agent *replan* when its environment does not respond as expected. Let it *guess* which container the ball was located in. It can plan to find the ball by opening that container. If the assumption is correct, the agent discovers the ball when it executes that plan. If the ball is not discovered in the container, the agent can notice this and replan for the new environment.

Guessing and replanning are required to adapt sequential planning for use in solving the search problem. Guessing compensates the planner for not having complete knowledge of the environment. Replanning compensates for guessing incorrectly. Below I will argue that plans for search require conditional and iterative plans. The conditional nature of search behavior is still present in that the sequential planning agent conditionally decides to replan. The iterative nature is still there in the possible repeated invocation of the planner. Representing conditionals and iteration explicitly in the plan provides several advantages. It makes proving correctness of an agents behavior easier by making the agent modular. It provides a level of representation which completely specifies the interaction between the planner and the execution engine (since it is the plan for the entire search).

If the planner produces a correct plan and that plan is correctly executed, then the agent behaves correctly.

1.4.2 Informative actions

In his pioneering work in the theory of knowledge and action, Moore [Moo80] defined *informative actions* to be those which result in knowledge of the truth or falsity of a proposition. These actions are like tests with litmus paper. After the test, the tester either knows that the paper is blue or that it is not blue. From the test result, the agent can infer other information about the state of the world (whether the solution tested was acid or base). The nature of tests is that the outcome is not known in advance; they have nondeterministic effects.

Sequencing other actions after informative actions can be problematic, because informative actions have nondeterministic effects. In order to sequence one action after another, the earlier action should have effects which have more information than required for the the preconditions of the later action. However, for nondeterministic actions, effects are typically less informative than required by the preconditions of any single action. Combining actions using conditional operators or nondeterministic choice is one response to this problem. (I discuss informative actions further in Section 2.3.3.)

1.4.3 Conditional planning

Relaxing the assumption that actions are deterministic creates new problems for an agent. Research in AI planning systems has addressed the issue of planning for actions with nondeterministic effects by introducing conditional plans. Conditional plans were first introduced in WARPLAN-C by Warren [War76]. Recently an approach was suggested for developing conditional nonlinear plans [PS92]. Whenever a plan included an action which had a nondeterministic effect, the rest of the plan would be conditional on which state of the world actually occurred. When the conditional was reached, the agent would examine the world to determine which branch of the conditional to take.

Figure 1.1 illustrates the interaction with nondeterministic effects. Supposing action 2 has as its preconditions $A \vee B$. This can be sequenced in a plan after action 1 which has effect A. However, to sequence action 4 (with precondition C) after action 3 (with effect $C \vee D$) requires first combining it with another action in a conditional. A suitable combination would be to combine action 4 with action 5 which has precondition D. These short plans are shown in figure 1.1. Note that $A \vee B$ is less than A in the information ordering.

Conditional planning is not sufficient for agents with only limited knowledge of their environment, since the latter prevents them from being able to reliably evaluate the expressions which will decide which branch of a conditional to execute. Conditional planning might be considered as a solution to the search planning problem provided that additional restrictions are placed on the expression which decides the conditional. The simplest restriction is to require the expression to be a property of the agent's mental state. This will be effective provided that the agent has accurate introspection. The simplicity of this approach is that it can rely on the simple syntactic marking of a formula as being known to the agent. Under this regime, acceptable expressions are just those which are properties

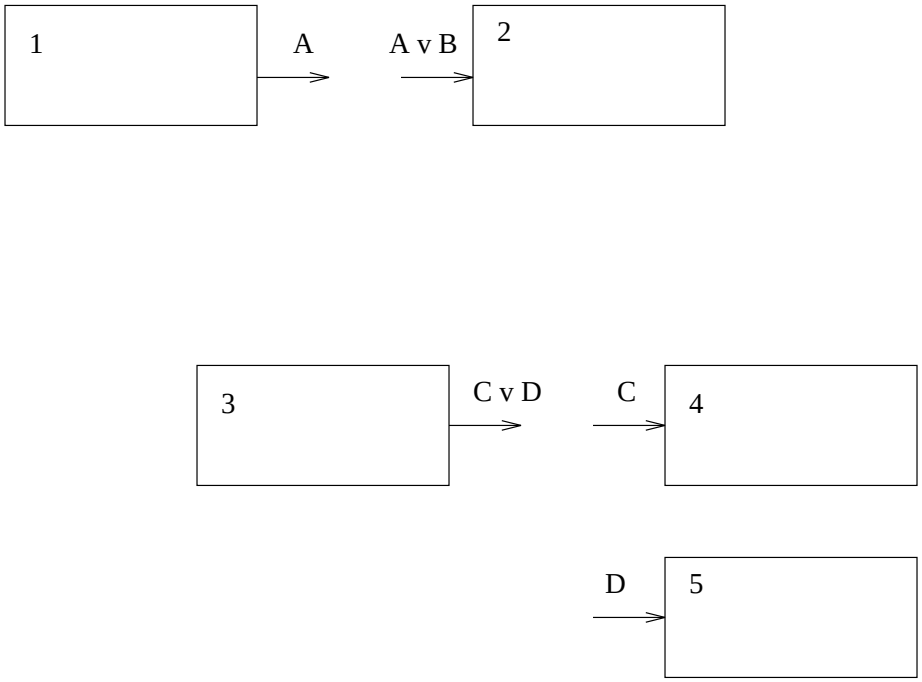


Figure 1.1: Nondeterministic effects and the need for conditional plans

of the agent's knowledge.

An alternate approach to adopting conditional planning solutions to the search planning problem is to insert plan steps to acquire information to decide the branch to take. Under this approach, the planner must be invoked to attempt to construct a plan to determine if the expression deciding the condition is acceptable. Acceptable expressions are those which the agent can reliably plan to know.

Since none of the work to date on conditional planning has distinguished the agent's epistemic state from the actual state of the world, neither of these two approaches to restricting the conditional expression has yet been explored.

1.4.4 Diagnosis

The goal of a search plan is the acquisition of information about the environment rather than some modification of the environment. This is a feature that search plans share with plans for diagnosis [RWC92, SW92, MR92]. That actions may both be informative and effect changes in the world has been recognized as early as McCarthy and Hayes [MH69] and used in recent work on medical decision support [RWC92]. That one may have to take multiple knowledge-acquisition actions before one's knowledge is sufficient to take decisive action has been recognized in work by McIlraith and Reiter [MR92] on incremental diagnosis.

Plans for incremental diagnostic tests such as proposed by McIlraith and Reiter are similar to search plans, except that they specifically exclude the possibility that the test actions will change the state of the world. As a consequence of expressing expected test

outcomes in a propositional logic, their diagnostic approach does not permit taking advantage of serendipity and finding things other than what one was looking for.

The literature on diagnosis has not addressed the issue of deciding branches of conditional action. However, if one chose to adopt the more general approach to extending conditional planning to the search problem, existing research on diagnostic plans provides good models for how to acquire information needed to decide what action to take.

1.4.5 Abstract search actions

While it would be possible to model searches at a more abstract level as individual actions (as suggested by Schoppers [Sch92]), there are at least two reasons for constructing detailed plans to drive search behavior. First, searches for different kinds of objects share a common control structure which I propose to use as the basis for my representation of search plans. Second, the actions which must be taken to conduct the search are not different from actions the agent takes to achieve other goals; having two different ways of representing the same actions is undesirable.

Schoppers presents a modal logic of time and mental state for application to planning. He also presents a *single-action* search operator for locating an object. The application domain he considers is that of a space-borne robot assisting a human EVA. He represents abstract search operations. In his planning language, the robot may perform a complete rotation to acquire an arbitrary object in the field of view of its camera.

```
rotating3(X) -- %
   b soon sust in-camera-fov(X) <+ true
```

A rough gloss of the above formula for the `rotating3` action is that an unconditional effect of this action is that the agent believes that, at some time in the future, the object X will enter (and remain in) the field of view of the camera. This is a compact description of a search.

Clearly, a simple way to incorporate search behavior into existing planning systems is for the designer of the system to characterize the preconditions and effects of the entire search. This abstracts away from the details of how the search is planned and conducted and the changes in mental state of the agent during the search. It is these issues that I am addressing in this work.

1.4.6 Planning for iteration

Tate *et al.* [THD90] claim that planning systems exist which plan for iteration. They cite NOAH [Sac75] and SIPE [Wil83] as taking a *black box* approach to construct iterative plans. To the extent that iterative parts of plans can be abstracted to single actions they can be incorporated into these plans. The representation system used does not permit reasoning about the details of the iterative part of the plan.

However, both these planning systems use a procedural net representation which prevents repetition of actions. Drummond [Dru85] extends procedural nets to plan nets which can represent iteration.

1.4.7 Reactive systems or situated agents

Since the mid 1980's there has been a counter-current in AI planning research which advocates reactive behavior in response to the immediate situation over execution of deliberate plans. A feature of reactive systems which is emphasized in the literature is that agents exploit features of their environment to reduce their need to deliberate about actions. While search can exploit features of the environment, the environment is not enough: the environment may be identical at different stages of a search. The choice of search action thus depends on the agent's *awareness* of what progress has been made in the search. Therefore search can be used to argue for the need for more internal state.

Current reactive systems and situated agents (with the exception of Rosenschein's work) are constructed manually for a particular task. When that task necessitates a search, the designer recognizes this need and manually encodes a procedure for a search.

1.4.8 Monitoring plan execution

The environments where agents carry out plans are frequently unpredictable. This has led to the suggestion that plans be augmented with sensor actions to ensure that expected states of the world occur at the appropriate times [DAD86]. This is not a different planning problem, simply a response to the inappropriate characterization of an unpredictable environment by blocks world assumptions.

1.4.9 A new approach to planning a search

This exploration of related planning systems leaves me with the conclusion that no existing system can be used for planning search behavior. The significant aspects of the search problem which differ from previously studied planning problems are the acquisition of information and iteration of similar actions while exploring the search space. Various systems have been proposed for *representing* and *reasoning* about actions which involve knowledge acquisition and iteration, including [Moo84, Les91]. Such systems can be used to infer properties of plans which have already been constructed, but do not themselves *construct plans* for complex actions.

By incorporating information about the agent's epistemic state into situation descriptions and action descriptions, means-end planning can be modified to plan for knowledge acquisition. Such a modified means-end planning system may be able to incorporate search into a plan if search is specified as a single action, with a set of preconditions and a set of effects. However, planning *how* to conduct the search requires representing and reasoning about actions whose outcome the agent does not know. Existing approaches to conditional planning do not sufficiently restrict the form of conditional expressions to guarantee they can be executed in the modified blocks world I consider. After a brief discussion of the application which motivates this planning problem, I state the claim of my thesis.

1.5 Application to the ANIMNL project

The problem of getting agents to look for things has come up in the process of developing the planning component of a system to produce human figure animation from instruction

texts. Details of this system, called ANIMNL (Animation from Natural Language), can be found in [WBB+92].

The problem raised in the ANIMNL project is that of *reference grounding*. Grounding natural-language referring expressions to objects in a (simulated) physical environment often requires exploration on the part of the simulated agent to locate them. Information from the interpretation of instructions will be used directly in forming search plans

In the application of this research to the ANIMNL project, the description of the search object is derived in part from the interpretation of natural language referring expressions. Descriptions can come from other parts of the interpretation as well, and also from common sense, world knowledge, etc.

Objects may also be introduced during planning or in reasoning about how to carry out actions that also need to be "grounded" in the specific environment (e.g. "remove the lid of the paint can" requires a flat-bladed tool to pry the lid off). Also, instructions may directly specify that the agent conduct a search to find an object (e.g. "Get me a screwdriver").

Search plans are developed in response to an *inability* of the agent to act. In the ANIMNL project this is detected after the planner has produced an atomic action, but before the agent can commit to performing it. Commitment is only possible when the objects required to perform the action are all perceived by the agent. Without the required objects, a search must be undertaken to locate them.

1.6 Thesis claim

I claim it is possible for automated agents to engage in search behavior. Engaging in search behavior consists in recognizing the need for a search, constructing an effective plan, and then carrying out that plan.

Further, I claim that explicitly representing a plan for an entire search in advance has several advantages for an agent. The agent has more information about its intentions and can thus provide better explanations of its behavior. The agent need not replan during the course of a normal search.

Finally, even constructing a search plan can be more efficient when the entire search is represented. This is because features common to different sites in the search space can be exploited to avoid redundant planning effort. Agents who represent search plans as I suggest can plan and act more efficiently than agents that construct only sequential plans

Expressing such a plan and reasoning about its effectiveness requires a representation language. I will select a representation language based on criteria derived from analyzing the search planning problem. Each of the three components of a system for engaging in search behavior (recognizing a search problem, planning, and executing a plan) will be designed and implemented to demonstrate that an automated agent can find things when he needs to.

The problem of planning search behavior differs from the planning problems that have previously been studied. In particular, I do not assume that agents have total knowledge of their environment.[1] I propose a planning system for this modified problem that includes

[1] This assumption is shared with some planners for diagnostic actions, see [RWC92]. Rymon's system does not construct a complete plan at once, and information seeking actions do not need to be repeated.

a logic for reasoning about the interaction between an agent and his environment. I claim that this system permits an agent to plan effective behavior in situations that call for a search to be conducted.

The need for a search plan is recognized automatically when knowledge about an object's location is needed. The abstract characterization of a search plan as a single action is expanded into a plan that has an iterative control structure. This detailed search plan specifies exploring different sites in the search space until the desired object is found or the search space is exhausted. A logic for reasoning about the effects of various actions on the agent's internal epistemic state and on his surrounding environment is used to guide construction of this detailed search plan and to insure its correctness. Correctness for a search plan requires that no search site is explored twice and that the exploration of all sites precedes abandoning the search in the plan. (Correctness is discussed in Chapter 3.)

There may arise situations where some element of information is needed by the agent and is not known to him. This problem has been discussed before in the planning literature as the problem of knowledge acquisition. Knowledge acquisition actions are inherently nondeterministic in their effects on the agent's knowledge state. (If they were not, then it could only be because the effects were already known to the agent.) I am restricting the problem of planning for knowledge acquisition to circumventing obstacles to direct perception of objects. Actions such as opening doors or lifting the lid of a box have the potential to reveal objects that were previously hidden. While any action that moves either the agent or the object being sought or the obstacle that stands between them may potentially reveal the object to the agent, I am primarily interested in operations on objects that are conventionally understood to function as obstructions (doors and lids). I call plans that manipulate these conventional obstructions systematically to reveal the location of an unknown object *search plans*.

When the agent believes that a desired object is located in one of a finite set of sites and needs to gain access to that object for some purpose, a search plan may be employed by the agent to achieve his goal. The two properties of non-repetition and eventual termination of the search ensure that when the object exists in the search space, behaving as specified in the search plan will satisfy the goal of locating the search object.

Reasoning about the effectiveness of an iterative plan requires reasoning about the termination of the iteration. Agents do frequently have some information about the location of a desired object which acts to delimit or order the space they are willing to search. This location information is less than what is needed by the agent. The combination of partial information about the location of the object and the nondeterministic knowledge effects of manipulating obstacles in that search space suggests repeatedly going to some different obstacle in the search space and moving it until they have all been tried – exhaustive search. Placing some constraint on the search space plays an important role in ensuring that the search will eventually terminate.

Iteration of the only action whose successful performance cannot be assumed (chest tube insertion) is handled by an ad hoc mechanism.

1.7 Plan of proposal

The remainder of this proposal will present the general problem of automating planning, reasoning about and carrying out physical searches. I present a formalism for representing search plans which is derived from recent research in reasoning about knowledge and action. The unique requirements on a representation for search plans are identified and used to argue against adopting existing formalisms. Chapter 2 presents a formalism for reasoning about search plans. Chapter 3 discusses criteria for evaluating the correctness of search plans. Chapter 4 describes constructing search plans by instantiating a generic search plan schema. Chapter 5 discusses the execution of these plans. Finally, I conclude with discussion of the remaining work to complete my thesis.

Chapter 2

Plan representation

Since the previously discussed planning approaches are insufficient for planning searches, I turn to an analysis of the representational needs a search planner. This chapter presents an analysis of the search planning problem, framed largely in terms of criteria for a representation. After presenting criteria for a logic in which to express search plans, I use these criteria to analyze previous formalisms.

2.1 Criteria for a representation

A plan for conducting a search should call for repeated selection of some action which may change the agent's knowledge about the location of an object. The previous discussion showed some of the failings of existing planning systems in attempting to represent plans for search behavior.

I argue that the following are required to plan, conduct, and reason about search plans:

1. The representation system should include terms for various objects, to represent the search object and distinguish it from other objects which might be encountered.

2. Objects are located at sites which should also be represented.

3. Agents perform actions to explore sites. It is useful for the actions to have parametric description so that an abstract plan for searching an abstract site can be represented.

4. Some representational mechanism is necessary for expressing the selection of a site to explore and an action to explore that site.

5. Some representational device is required to distinguish what the agent knows and perceives from what is actually the case (but not known to him).

6. Due to the epistemic nondeterminism inherent in the search problem, some means of expressing conditional plans is necessary. It is also necessary to represent the iterative control strategy of a search plan.

2.2 Indexicality

It is also desirable that the representation of search plans be indexical for efficiency and to reflect the appropriate knowledge preconditions of actions. Lesperance [Les91] argues that knowledge requirements for actions are largely indexical knowledge: knowledge of the environment from the perspective of the agent. In order to open a container an agent need only know that the container is *here* (i.e. at the agent's own location). The agent does not need to know where "here" is in some external coordinate system. Similarly, the agent need not know the identity of the container except that it is the one that is here.

Lesperance provides indexical functions and operators to change the context of interpretation, which permits him to characterize the difference between indexical knowledge and objective knowledge. He further shows that indexical knowledge is usually required as a precondition for action rather than objective (non-indexical) knowledge.

Lesperance presents an example plan which involves search. He proves that an agent is able to move to where an object is located if he knows that the object is within some bounded distance in front of him. The search space for this action is the finite collection of locations in front of the agent. The action terminates when the agent arrives at the search object after iterating the action of moving forward one location at a time.

It is the nature of moving forward that changes the agent's physical position. Thus the nature of the (indexically specified) action means that the agent explores a different portion of the search space each time. Few actions have the property that repeating them will constitute exploring different parts of the search space, and it is only for such actions that Lesperance's approach will adequately represent a search.

According to Subramanian and Woodfill [SW89] indexical representations can lead to more efficient implementations. In particular, they can be implemented in combinatorial circuits which can execute in constant time. So an indexical representation is advantageous since it both represents knowledge accurately and affords an efficient implementation.

Ensuring that the plan is represented in an indexical representation and makes use of indexical information when necessary is an ultimate goal of my research in search planning. However, given the existence of a mechanism for translating from a situation calculus notation to an indexical representation, I consider this of secondary importance.

2.3 Candidate formalisms

There is no lack of formal theories of action suitable for use in this thesis. There are also several competing formal theories of internal states of intelligent agents. This section presents an analysis of their relative merits for purposes of representing, reasoning, and proving correctness of search plans.

In general both logics of action (or change) and logics of epistemic state extend first order logic with additional mechanism to make inference depend on a particular time or on a particular belief state of some individual.

2.3.1 Time and causation

Among logics of change, one can distinguish between logics of time and logics of causation. Although both have to do with the changes in context as time passes and events occur,

	change		context		duration	
	causal	temporal	explicit	implicit	interval	point
situation calculus	X		X			X
dynamic logic	X			X		X
event calculus	X	?	X		X	
temporal logic		X	X	X	X	X

Table 2.1: Properties of logics of change

logics of time take times as primitives, while logics of causation take events as primitives and often do not include times at all. When they do, time is often just an index into a sequence of events which determines how context changes. Although logics of time can describe the temporal relation between a cause and its effect, they require some additional mechanism to distinguish causation from other reasons for temporal ordering. Conversely, logics of causation require additional mechanism to express temporal relations between non-causally related events.

Logics of change may consider times to have no duration (points) or some duration (intervals), or they may combine point and interval times. The relevance of this distinction is in specifying the duration (or lack thereof) over which formulas are evaluated. In point logics, formulas are evaluated with respect to an instantaneous state. In interval logics, formulas are properties of a span of time.

There are, roughly speaking, two approaches to logics of time or causation. Either times may be added to the ontology of a first-order theory or time may be considered an element of the context which determines the interpretation of formula. The first approach results in a first-order logic, the second in a modal logic.

Having identified relevant distinctions, I will now use them in characterizing the relative merits of candidate formalisms for representing and reasoning about search plans.

Dynamic Logic Dynamic logic is a causal logic that was invented by Pratt [Pra79] for use in analyzing programs. Dynamic logic is a modal logic, with modal operators for each action. Because of the way actions are interpreted as modal operators, there are no event individuals and consequently no quantification over actions in dynamic logic. It is therefore unsuited for representing plans where the selection of actions is expressed by quantification over the action.

Event calculus Kowalski's event calculus [KS86] permits quantification over events and relates the truth of propositions and the occurrence of events to intervals. In the event calculus, intervals are described relative to events, instead of indexically related to the current time. Kowalski does not include operators to express conditionals or iteration of events. In general, the only events which are considered are those which instantaneously achieve a change in the world. Representing complex actions (such as needed in search plans) in such a framework would be difficult. Thus, it is not appropriate for my purposes.

Temporal Logic Dynamic logic and the situation calculus have been criticized by many researchers for restricting the possible temporal relations between actions and between causes and their effects. Allen's use of an interval-based temporal logic is one attempt to overcome these limitations [All84].

Temporal logics are a family of logics rather than a single formalism. (For a recent survey see Van Bentham [vB88].) In different temporal logics, the world may be described at instants or over intervals. Both modal and non-modal formalisms have been studied.

However, additional mechanism is needed to express causality in temporal logic, since it primarily concerned with reasoning about *when* things happen instead of reasoning about the effects of actions. Rather than complicate a temporal logic with additional mechanism for reasoning about causation, I deem it appropriate to use a causal logic based on the situation calculus.

Situation Calculus The situation calculus was invented by McCarthy and Hayes [MH69] for representing intelligent actions. In it, doing an action is a relation (or function) between two situations. Situation calculus is therefore classified as a causal logic of change. In contrast to dynamic logic, it is possible to quantify over actions in the situation calculus. However, it is a (sorted) first-order logic not an indexical representation. This makes it difficult to capture the correct knowledge preconditions of actions.

I expect to overcome this difficulty using a technique proposed by Subramanian and Woodfill [SW89] to derive an indexical-functional representation from a restricted form of the situation calculus.

2.3.2 Attitudes

There are first order logics of mental attitudes, just as there are first order logics of change. Such logics formalize the objects of mental attitudes as strings which represent propositions. Similarly, modal logics have been proposed to capture the context dependent nature of reasoning about mental states. A recent alternative to these two approaches departs markedly from first order logic to take arbitrary sets of propositions as constituting *situations*.

Although there are arguable differences between these differing approaches to reasoning about mental attitudes, they are all more or less suitable for use in conducting searches. The important point is that *some* means to distinguish the state of the world from the mental state of the agent is required.

Syntactic Epistemic Logic A first-order logic was used by Morgenstern in her work on reasoning about knowledge and action [Mor88]. In her formalism, objects of epistemic predicates are strings. Functions are included in the logic to manipulate those strings.

Syntactic epistemic logic is most compatible with a first-order logic of change, such as the situation calculus. This was the combination used by Morgenstern.

This approach is unwieldy as the functions which manipulate strings must be defined within the logic, and reasoning about inferences drawn by an agent requires appeal to the definition of these string manipulation functions. A modal alternative to this will be discussed below.

Situation Theory Situation theory was invented by Barwise and Perry [BP83] and extended by Keith Devlin [Dev91] and others. Situations are arbitrary collections of propositions and hence may be inconsistent or incomplete. This contrasts with both modal epistemic logics and syntactic epistemic logics which start from complete and consistent possible worlds and achieve incompleteness through quantification. Since arbitrary collections of propositions are considered, situation theory does not require that tautologies be believed nor does it require that beliefs are closed under logical consequence. Indeed, situations may contain inconsistencies.

Situation theory contains mechanism for representing spatio-temporal locations, so it alone might be sufficient for the purpose of representing search. However, it is fairly new and less well understood than the alternatives. Since there are no readily available automated inference engines for situation theory, it cannot be considered a viable alternative on engineering grounds.

Epistemic Logic Hintikka [Hin62] proposed modal logics of knowledge and belief. In epistemic logics, modal context is used to prevent substitution into expressions which describe the internal state of agents. This approach has been extended to multiple agents.

Epistemic logic is most compatible with modal logics of change, such as the modal temporal logic argued for above. A consequence of adopting modal logics for mental attitudes is that tautologies are automatically included as objects of belief, knowledge, and perception. This is a consequence of the necessitation rule of inference which is part of all common modal logics. Also, a common modal axiom specifies consequential closure of mental attitudes. This is the approach I propose to take for representing the knowledge of the agent.

2.3.3 Combined theories of change and mental state

Robert C. Moore [Moo80] first provided a logic of knowledge and action along the lines suggested by McCarthy and Hayes [MH69]. His logic combined a modal logic of knowledge with a situation calculus logic of action. To a basic situation calculus of actions he added action constructors which build sequences, conditional actions, and iterated actions from other actions.

Moore's theory of *informative actions* illustrates the knowledge effects of an action. An action is said to be informative about proposition P if, after the action, the agent knows P and every world accessible (via the knowing accessibility relation) from the world resulting from the action is the result of doing a similar action in a possible world which was accessible (again via the knowing relation) in the initial world. To express this formally, Moore appeals to a first-order meta-language for his logic in which $\mathbf{R}$ is the accessibility relation which expresses that world w_2 results from the occurrence of $Event$ in w_1 and **Know** is the accessibility relation which expresses that worlds w_2 and w_3 are indistinguishable given what $Agent$ knows. The expression $\mathbf{True}(w, P)$ indicates that P is true at world w. The condition for informative actions about P is expressed as

$$\begin{aligned}\forall w_1, w_2.(&\mathbf{R}(Event, w_1, w_2)\\ &\rightarrow \forall w_3.(\mathbf{Know}(Agent, w_2, w_3)\\ &\qquad \leftrightarrow (\exists w_4.\mathbf{Know}(Agent, w_1, w_4)\end{aligned}$$

$$\wedge \mathbf{R}(Event, w_4, w_3))$$
$$\wedge(\mathbf{True}(w_2, P) \leftrightarrow \mathbf{True}(w_3, P))))$$

Not included in the above formula is the characterization of the lack of knowledge of P in the initial world w_1. When an informative action is successful, there are two possible outcomes: either the agent knows the truth of P or knows that P is false.

As has been argued by Lesperance, it is *indexical* knowledge which is the prerequisite for most action, rather than the objective knowledge which Moore uses in his knowledge preconditions. Aside from that objection, Moore's formalism addresses most of the requirements of a representation for search plans.

Another approach to combining representation of actions and epistemic state is taken by Drummond [Dru86]. This approach is based on procedural net descriptions of actions [Sac75]. Drummond presents a Petri-net like graph representation of plans which distinguishes between effects on the external environment and effects on the agent's internal state. However, all the preconditions on actions in his representation are on the agent's belief state. He distinguishes between beliefs which must be present and those that must be absent in order to permit an action. The approach I propose is more general in permitting preconditions on either the external environment, or on the agent's internal state.

2.4 Meeting representation criteria

Here I argue that the criteria for a search plan representation system are largely met in the formalism I propose. The proposed formalism is a combination of a modal logic of epistemic state (beliefs, knowledge, and perceptions) (Section 2.3.2) with the situation calculus (Section 2.3.1).

Search plans are represented in a modal predicate logic for time and mental state. This approach is similar to that taken by Schoppers [Sch92]. Using this formalism, I can show how search plans increase the ability of the agent. In particular, they permit agents to act in the face of knowledge of disjunctive formulas.

Action selection is managed through a two-stage approach that uses conditional actions and an action of selecting a site to search from a set of alternatives. Modal operators maintain a distinction between mental state and actual truth. (Axioms are presented in Section A.3 to formalize their interaction.) Iteration is represented using an operator which combines a condition and an action into a while-statement. The formalism is presented as a logic complete with inference rules suitable for constructing proofs of the ability of agents to act effectively. (See Appendix A for more details.)

2.5 Situation calculus examples

The situation calculus is useful for reasoning about situations before and after actions occur. To represent that ball b is in container c in situation s I will use the formula:

$$in(b, c, s)$$

To represent that the ball is in the container after doing action a in situation s, I use:

$$in(b, c, do(a, s))$$

Plans such as search plans are expressed as actions in this extended situation calculus. From disjunctive information about the location of the ball in the initial situation s we want to determine a search plan p such that after doing p in s, the agent will know the location of the ball. The initial situation might be represented with an existential quantifier as:

$$\exists c.in(b, c, s)$$

The expression of the goal for search plan p is that the agent know which of the containers the ball is in. (The next section will explore details of the form which the search plan p might take to make it possible to achieve this goal.) This might be expressed as:

$$\exists c.\mathbf{K}(in(b, c, do(p, s)))$$

Information may be passed from one action to another by storing it in a variable. Thus an important kind of action is assigning a value to a variable.[1] Assigning the value 3 to variable x is represented as:

$$x := 3$$

Descriptions of actions may be atomic (such as p or a above), parameterized (such as $open(x)$ below), or they may be complex actions constructed from other actions using sequencing, conditional selection, or iteration. The system of inference for reasoning about various actions is presented in Section A.3.1.

2.6 Example search plan

As an example expression of a search plan in this representation, consider the problem of an agent carrying out the instruction step, "Go into the kitchen and get me the coffee urn." Suppose, unbeknownst to the agent, the coffee urn is stored in one of several closed cabinets in the kitchen. When the agent enters the kitchen, he will not be able to see the urn and will have to look for it, in order to get it for the speaker.

Figure 2.2 shows a search plan for finding a coffee urn (urn) when $open(x)$ is an action which constitutes steps in exploring the search space. The plan consists of a while loop followed by a conditional statement. The while loop calls for iteration until either the object of the search is found or the search space is exhausted.

The success condition is that the urn is perceived to be in some location, c. The term now in the condition is interpreted indexically with respect to the situation when the plan is executed. When this condition holds, the plan proceeds to the remaining actions in the plan which is here described as win.

Similarly, the failure condition expresses that the search space has been exhausted; there are no further actions to explore unexamined parts of the search space. When the failure condition holds, the plan proceeds to actions which should be taken under those circumstances (described as $lose$).

When the search is incomplete (neither success nor failure hold), then actions are selected and taken, and the search plan repeats. Here the container x is selected to open.

[1] An earlier representation formalism used logic variables to pass information between actions. The resulting formalism required a modal temporal logic. The most straightforward way to pass information in the situation calculus is to introduce variables specifically for that purpose.

```
while (¬∃c(See(in(urn, c, now))) ∧ B(¬empty(sites, now)))
 do
   begin
     x := select(sites);
     sites := remove(x, sites);
     open(x)
   end
if ∃c(See(in(urn, c, now)))
 then
  win
 else
  lose
```

Figure 2.2: Example search plan

The while-statement expresses repeated execution of the *open* action. The plan also includes information which constrains the selection of the container to be one the agent has not previously opened. Once a site to explore is selected, it is removed from the candidate sites. The *open* action may have a side effect of adding new sites to the *sites* variable. An initial value for *sites* is established by examining the agent's perceptions at planning time.

The various operators used in the above expressions include quantifiers ($\exists$), logical connectives (conjunction $\wedge$, negation $\neg$,), and modal operators (perception **See**, belief **B**). The modal operators are given a possible worlds semantics with accessibility relations for epistemic contexts (knowledge, perception) and temporal relations.

2.7 A representation language for search plans

I have argued that a modal logic (extending the situation calculus with epistemic operators) is an adequate representation for the search planning problem. An initial specification is given in Appendix A. Subsequent discussion of plan construction and execution will make use of this notation. The point of giving a precise formal semantics for search plans is that it permits me to be precise about how to automatically reason about how the complex activity of search is related to its component acts.

Chapter 3

Properties of search plans

An important aspect of reasoning about plans of all kinds is being able to determine if a given plan will achieve given desired effects. I am interested in two different properties of each search plan: that sites in the search space are not redundantly explored (a safety property, nothing bad happens), and that the search object is eventually found if it exists in the search space (a liveness property, something good eventually happens). Safety properties can be verified by invariance arguments as shown in [Lam84]. Liveness properties can be verified by well-foundedness arguments [MP82, OL82]. Plans which satisfy these safety and liveness properties will be considered *correct*.

3.1 Avoiding repetition

As mentioned earlier, the invariance to prove for the iterated step of a search plan is that each site is explored at most once. A single execution establishes conditions which prevent a repeated execution and no other action changes those conditions. Each iteration of the site exploration action in a search plan should be different from all the previous occurrences, so that the agent does not keep searching the same space. This can be accomplished in search plans by preventing an agent from exploring a site which he knows does not contain the search object.

Frame axioms ensure that information obtained earlier in the search will still be known by the agent at a later time. To illustrate this reasoning, consider that an agent's knowledge about the location of an object is not lost by opening or closing another container (for all actions a, situations s, containers c, and objects b):

- No actions change location or containment. This is expressed in the form of a successor state axiom:

  $$Poss(a, s) \rightarrow in(b, c, do(a, s)) \leftrightarrow in(b, c, s)$$

  The notation $Poss(a, s)$ represents that the preconditions for action a are satisfied in situation s.

- This is known due to the necessitation rule for $\mathbf{K}$:

  $$\mathbf{K}(Poss(a, s) \rightarrow in(b, c, do(a, s)) \leftrightarrow in(b, c, s))$$

- Knowledge is consequentially closed (axiom K for $\mathbf{K}$):

$$\mathbf{K}(\varphi \rightarrow \psi) \rightarrow \mathbf{K}(\varphi) \rightarrow \mathbf{K}(\psi)$$

- Assuming knowledge before action a (and a's preconditions hold):

$$\mathbf{K}(Poss(a, s)) \qquad \mathbf{K}(in(b, c, s))$$

- *Conclude:* knowledge after action (a):

$$\mathbf{K}(in(b, c, do(a, s)))$$

3.2 Ability

The literature on knowledge and action formalizes the above liveness property with a technical definition of *ability*. This section discusses proofs of ability for various search plans. The formal definitions of ability which have been proposed by Moore [Moo84], Morgenstern [Mor87], Lesperance [Les91] share the idea that an agent has the ability to achieve an effect by some action when he knows that every way in which he does that action results in that effect being the case.[1] When "ability" is used in this sense it will appear in italics (*ability*). In formulas, the modal operator **can**() will be used.

An agent is *able* to bring about a situation which has property φ by taking an action a in situation s ($\mathbf{can}(\alpha, \varphi, s)$) when the agent knows that φ is an effect of action α. This is expressed as follows:

$$\mathbf{can}(\alpha, \varphi, s) \stackrel{def}{=} \mathbf{K}(\varphi(do(\alpha, s)))$$

3.2.1 Loop termination

Given a search plan, proving that an agent is *able* to accomplish some task using that plan relies on an assumption that the search plan terminates successfully. There are two avenues I am exploring for insuring that these plans terminate. The first approach uses a finite bound on the search space. The second approach places restrictions on the structure of time, such that only finitely many steps of the iteration are executed. (Lesperance [Les91] takes the first approach.) These two approaches result in two different kinds of well-foundedness arguments: either the search space is well-founded or the relevant part of the agent's resources (specifically, the amount of time the agent devotes to the task) is some abstract well-founded structure.

Finite search space It is correct to assume that a search plan terminates when the search space is finite and the agent does not repeat actions already performed. Finite search spaces can usually be known in advance before performing the first step of the search (even if they are large, as in the case of a combination safe). With nested containers, though, the fact that an agent perceives finitely many containers at any given moment does not

[1] Note that an agent may still be considered *able* to achieve some effect φ if φ is already true and the action did not make φ false. φ is simply true when the action is done.

```
while (¬∃c(See(in(b, c, now))) ∧ B(¬empty(sites, now)))
 do
    begin
       x := select(sites);
       sites := remove(x, sites);
       open(x)
    end
if ∃c(See(in(b, c, now)))
 then
  win
 else
  lose
```

Figure 3.3: Search plan for finding a ball

guarantee termination of the search. Another argument for termination is needed in this case.

In particular, one can appeal to the fact that the *in* relation between containers is transitive and well-founded (meaning that there can be no infinite nesting of containers). König's lemma states that any finitely branching well-founded tree can only have finitely many nodes. So if finitely many containers are perceivable to the agent in the search space and each contains only finitely many other containers, we can conclude that the search space in this case is finite. Unfortunately, well-foundedness is not first-order expressible, so this deduction will not be automated.

Finite iteration The alternate approach of constraining the number of iterations to be finite by definition will not be taken in this dissertation. The problem with this approach is that it is difficult to construct an execution engine for search plans which will respect this definition.

3.2.2 Ability and search plans

To find a ball which is hidden inside one of several (possibly nested) containers, an agent might adopt the plan in Figure 3.3. Here the site exploration action in the search is to open a new container, x. This step is repeated until a ball b is found or the agent runs out of containers to open (*sites* becomes empty).

To prove that agents are *able* to use the Figure 3.3 search plan to find a ball hidden in some container, more must be said about the *open* action and about the nature of the search space. To complete the proof requires frame axioms and a description of the current situation. Frame axioms are needed which specify that opening containers does not cause the ball to move. The approach to the frame problem I am considering is discussed in Appendix Section A.3.1.

3.3 Efficient strategies

A minimal requirement of search plans is that they be correct. Better search plans are not only correct, but also make good use of time and other resources. Search plans will also be evaluated for their efficiency.

A good strategy for searching is to look in the most likely places first. An important area of remaining research on this thesis is to determine sources of information about ordering the sites in a search space and how that information gets used to ensure efficient searches.

One potential advantage of using an indexical representation is that it might simply specify that the search proceed to the nearest unsearched site if the ordering is a "greedy" decision based on closeness.

Once a site is selected for exploration, it remains to select an action for exploring that site. The best that may be possible at planning time is to arrange for the type of site to determine the action.

Future research will explore whether site ordering is best done eagerly at planning time, or delayed until immediately prior to site selection. I will also study what impact the timing of these different choices has on decisions about actions to explore the selected site.

An obvious goal for research on planning efficient searches would be to determine when sufficient information was available to permit known search strategies (such as A*) to be employed. The description of the search planning system I propose is modular, in that it specifies where site selection information is incorporated into the plan. More work must be done to determine precisely what that site (and action) selection information is.

3.4 Summary: Correctness of a search plan

A search plan is correct when its execution leads to the discovery of the desired object when that object is present within the space described by the search space constraint. Two components of correctness have been described and methods for proving a plan both avoids repetition and terminates were discussed. The central concept of deducing *ability* (or knowledge of the effects) of a search plan was also presented.

A search plan is efficient when its execution leads *quickly* to the discovery of the object. Determining the order in which sites are explored has a large impact on this efficiency and will be a major topic of future research.

Chapter 4

Plan construction

Having settled on a representation language to use in the search plans system, I now present my design for a planner which can construct these plans. Discussion of the correctness of plans developed by this system was already discussed in Chapter 3.

Iteration is introduced in a plan by schema instantiation. The iterated actions must be planned for, so the planner is recursively invoked to plan the inside of the loop. Occasionally, there are cases where a search is not yet complete, yet some other action is appropriate. Such cases can be grouped into two general classes: interruptions where the search is later resumed, and relativized searches where the search is prematurely terminated. The term "relativized" is borrowed from Cohen and Levesque [CL90] where they consider *relative* persistent goals. Davis [Dav92] discusses an alternative representation for plans which also supports these kinds of alternatives to completing a search.

4.1 Iteration schemata for search

Generally speaking, search plans should be constructed when the knowledge requirements for direct action are not met, but there is sufficient knowledge to achieve the desired goal via a search.

Search plans are constructed by schema instantiation. Selecting an algorithm for search amounts to deciding which search plan schema to use as the basis for constructing the search. That schema is then instantiated with a description of the search object, a specification of the search space, and actions to take to explore the search space.

Constructing search plans via a schema is similar to instantiating other action descriptions as is commonly done in other planners. The substantive difference is that only some of the information is substituted, while the rest is constructed by re-invoking the planner. (The correctness of the resulting search plan can be verified with an inductive proof. Whether or not this proof is carried out at planning time is yet to be determined.) The substitution also involves substituting formula instead of simply instantiating variables to terms. This formula substitution occurs with the search object description and the search space constraint.

Once the search space constraint and the description of the search object have been incorporated into the search plan, the incremental actions must be specified. This is done by setting the search space constraint as the initial situation and the search object

```
while (¬found ∧ ¬exhausted)
 do
   step
if found
 then
   win
 else
   lose
```

Figure 4.4: Search schema

description as the goal situation and constructing a conditional plan which achieves the goal. The conditions on which each branch depends are taken as successive assumptions which the plan execution engine must make.

The process of creating a search plan is simplified through the use of a generic search plan schema (see Figure 4.4) that is instantiated to form different search plans. Two conditions are tested to determine what action occurs next. Condition *found* indicates that the object of the search has been located. When it is true, the plan proceeds to a successful conclusion. Typically this conclusion, *win*, will be replaced with whatever action sequence was delayed, pending the agent's finding the object. The failure condition, *exhausted*, expresses that the entire search space is known not to contain the object. When this is the case, the plan continues with an alternate course of action (whatever is substituted for *lose*). Otherwise, the schema specifies that it is appropriate to continue to search. At each iteration of the search, the exploration of some site in the search space is specified by whatever is substituted for *step*.

4.1.1 Interrupting a search

Search plans may be interrupted and resumed due to two aspects of a search plan. Each iteration of a search plan may be begun in any state. Also, progress in conducting a search is recorded in the agent's knowledge about the world independently of the search plan.

There are no preconditions on a search plan, except those which may propagate forward from the actions substituted for *step*, *win*, and *lose*. Preconditions from *step* may be discounted if they are incorporated into the *exhausted* condition. As a consequence, any iteration of a search plan may be begun in any state since every state will satisfy one of the three conditions that the basic search plan specifies (when $(\neg found \wedge \neg exhausted)$ explore a site, when *found* terminate successfully, otherwise (*exhausted*) terminate unsuccessfully).

Progress in a search is recorded in the agent's knowledge of the world. When an agent opens a container, his knowledge about the world is increased by discovering the contents. If a search is halted, and some other action is performed before the search resumes, the agent will not have lost his knowledge about the world due to the intervening action. He may even have gained knowledge about the world. When the search is resumed, his knowledge of the world determines the course of the search.

Recall that an important assumption about the environment in which search plans are

```
while (¬found ∧ ¬exhausted)
 do
   if interrupt
    then
     alternate
    else
     step
if found
 then
  win
 else
  lose
```

Figure 4.5: Interrupted search schema

executed is that the agent is the only source of change in the world. Therefore, even though the intervening actions may drastically alter the world state (e.g., if the agent set fire to the house he's been searching through) the agent can use knowledge of the effects of his actions to update his knowledge of the world. This is the sense in which an agent does not lose knowledge about the world.

An agent may *plan* for a search to be interrupted using a different schema. Figure 4.5 depicts a schema for an interruptible search plan which takes *alternate* actions when the *interrupt* condition holds and then resumes the plan.

4.1.2 Abandoning a search

Cohen and Levesque [CL90] provide a markedly different approach to rational action than researchers in the AI planning paradigm. For them, the important issues are committing to intentions to act in the future and dropping those intentions when appropriate. I take their point that it may be important to be able to relativize a commitment to any arbitrary condition and show how to do that in search plans.

Relativization similar to persistent, relativized goals could be formalized by adding another condition to the existing conditions of a search plan schema. Figure 4.6 illustrates a search plan with an alternative exit condition labeled *relative* which when true permits the agent to exit the search and perform the steps here schematized as *exit*.

4.2 Recursive planner invocation

A generic plan for exploring an arbitrary search site is constructed by calling the planner (when the search plan schema is instantiated). This generic plan is substituted for the *step*

```
while (¬found ∧ ¬exhausted ∧ ¬relative)
 do
   step
if found
 then
   win
 else
    if relative
     then
       exit
     else
       lose
```

Figure 4.6: Relativized search schema

action in the schema. For example, the actions

```
begin
   x := select(sites);
   sites := remove(x, sites);
   open(x)
end
```

are substituted for *step* when forming the example search plan of Figure 2.2 (page 19).

The initial state description for this recursive invocation of the planner is an assumption that the search object is actually located in the site to be explored. The goal state for the recursive invocation is that the agent has confirmed this assumption. To the extent that search sites require similar exploration, the planner can plan for exploration of an arbitrary site in the search space and leave management of the assumptions (one approach to determining which specific site will be explored) to the plan execution engine. Actions proposed by this recursive invocation of the planner must respect the non-repetition property of the search plan.

Calling the planner again is motivated by the potential that site exploration may require a complex plan (perhaps even a nested search plan). For a given container in a search, it may be necessary to remove something from the top of the container before it can be opened.

4.3 Planning for a search

This section has developed an approach for constructing search plans when the need to do so is recognized. An iterative control structure is assumed as part of the schema-instantiation construction technique.

The modifications to a planning system I describe show how to incorporate iteration into a plan in a principled way. This approach may generalize to principled introduction

of iteration for different purposes, but I am only interested in the principled introduction of iteration for the purposes of representing a search.

The planning process must also determine an order which will be used at execution time to select the next site to explore. Section 3.3 described desirable properties of orderings on the sites in the search space. Just how such an ordering is chosen has yet to be determined. Because it has an impact on the efficiency of the resulting search, it is necessary research to complete my thesis.

Chapter 5

Plan execution

5.1 Description of execution of a search plan

To illustrate the execution of a search plan, consider the previous example of an agent searching for an urn in a collection of (possibly nested) containers. The search plan for this task is shown in Figure 2.2 (page 19). Initially, the agent does not see the urn ($\neg\exists c(\mathbf{See}(in(urn, c, now)))$) and there is some unexplored search site ($\mathbf{B}(\neg empty(sites, now))$). Thus, the search plan directs the agent to proceed with the search. That is to say, the operational semantics of the search plan specifies first checking the condition and then executing the action if the condition holds. Part of the condition is expressed as a description of the agent's perceptions. This description is checked against input from the perceptual system of the agent. The other part of the condition for the while loop is a property of an internal data structure. It is important that only conditions which can be verified by examining the internal state of the agent (i.e. perceptions, knowledge, or beliefs) appear in while statements.

The next action in the plan calls for the agent to select some container which he has not yet visited and open it. The *select* operation chooses from the available sites in the remaining search space. The next assignment statement updates the *sites* variable to reflect this selection. Then the selected site is examined. For the agent to take this action, he must believe that he can achieve perception of the object by the action $open(x)$. However, his knowledge is only of the search space, which has less information than will support that conclusion. To conclude that he can find the object by this plan, he must assume that the urn is in the selected container. This assumption is added to the agent's belief space and belief revision is performed if necessary to maintain consistency of beliefs.

The action *open* has deterministic effects on the world – the container is now opened – and nondeterministic perception effects on the agent. After this action, the agent either perceives the urn or does not. The first case calls for the successful termination of the search plan. The second case introduces a contradiction in the agent's belief space since $\mathbf{See}(\neg in(urn, c, now))$ implies $\mathbf{B}(\neg in(urn, c, now))$. (See axioms for the perception operator, Section A.3.2.)

The only remaining case of the while loop condition is when the search space has been completely explored. In such a situation, the variable *sites* will be empty. In this case, the plan calls for the unsuccessful termination of the search. The while loop terminates,

and the succeeding conditional is evaluated. Since the urn is not seen, the *lose* actions are selected.

5.2 Evaluating conditional expressions

Conditional expressions are restricted to be properties of the agent's internal state (i.e. perceptions, knowledge, or beliefs). This ensures that the agent will always be able to decide which branch of the conditional to take at run time.

5.3 Selecting obstacle/site to search

For the second component of action selection, a search strategy amounts to deciding on the order in which different sites in the search space will be visited. From the perspective of belief revision, this ordering establishes an order on sets of beliefs which will be considered. In the process of nonmonotonic revision of the agent's beliefs, a consistent extension of the agent's knowledge will be selected in such a way as to include at least one of the assumptions about the site containing the search object. This idea of expressing control information through justifications recorded for the purpose of belief revision is due to Doyle [Doy79].

In the previous sections, I have described physical search in terms of the actions an agent executes to explore a search space. Another way of viewing it is in terms of the belief revision process. Having a search space in mind can be explained as believing a disjunctive statement about each of the possible locations of the object. For each iteration, some part of the search space disjunct (usually just one location) is assumed to be actually true. Then the agent acts so as to reveal the truth of that assumption. If as a result of that action the object is discovered, the search is successfully concluded. Otherwise, the world (and agent's perceptions) are in conflict with its expectations, and belief revision must take place.

In general, belief revision is difficult,[1] but the structure of search plans helps to *encapsulate* a set of assumptions for which I believe revision is much easier. For conducting searches I identify two disjoint sets of relevant assumptions:

1. Internal assumptions over which the search exerts control

2. Supporting assumptions on which the ability to successfully search is based

Assumptions about the location of the search object are easily revised by selecting other locations in the remaining search space. Revising supporting assumptions which underlie the search itself may require abandoning the search. This class of assumptions contain things like the agent's ability to recognize the search object and beliefs about the effects of actions used to explore the search space. (Other assumptions may be revised independently from the search since they are irrelevant to carrying out the search.) I am currently implementing a belief maintenance system to attempt to exploit this classification of assumptions.

[1] Kean and Tsiknis [KT90] show that approaches based on computing prime implicants have at least exponential worst case complexity.

5.4 Executing a search plan

This section has illustrated the interaction between an execution engine for a search plan and its environment. The search plan execution engine will make use of a belief revision system to control the order in which sites are selected for exploration.

Chapter 6

Thesis

6.1 Review of current state

The problem of planning for and carrying out a search has been presented. Difficulties using existing formalisms for planning and reasoning about conducting searches were discussed. Analysis of those difficulties suggested a novel logic which was presented. A simple approach to constructing plans using this formalism was described. Reasoning about the plans shows how they can effectively plan for goals which can not be planned for by other planners. Finally, a belief revision system was discussed that supports efficient execution of these plans.

A theorem prover for the modal logic has been implemented. Public domain ATMS code can be used to support recording dependencies between modal formulas. Earlier versions of plan execution engines for recursive plans have been implemented (but without mechanisms to manage agent beliefs and memory).

6.1.1 System Description

The system for planning and conducting searches consists of three functional components and three databases. Input to the system is a goal to achieve. Output from the system is search behavior. A first-level decomposition of this system is depicted in Figure 6.7.

The three functional components include a planner (MODIFIED PLANNER) which constructs plans, a plan executive (EXECUTIVE) which generates behavior (based on the plan, world state and agent state), and a nonmonotonic revision component (ATMS) which maintains consistency among the agent's beliefs.

The three databases include world state, agent knowledge, and agent assumptions. The agent's internal state is a combination of his knowledge and assumptions. An agent's perceptions (also part of his internal state) are part of his knowledge. An agent's beliefs are the combination of his assumptions and his knowledge (his entire internal state). A database of the world state is maintained, and this is the environment in which the behaviors are executed. Part of the world state database is available to the agent as knowledge. This knowledge partition increases monotonically over time. An collection of the beliefs of the agent is maintained separate from (but consistent with) the agent's knowledge.

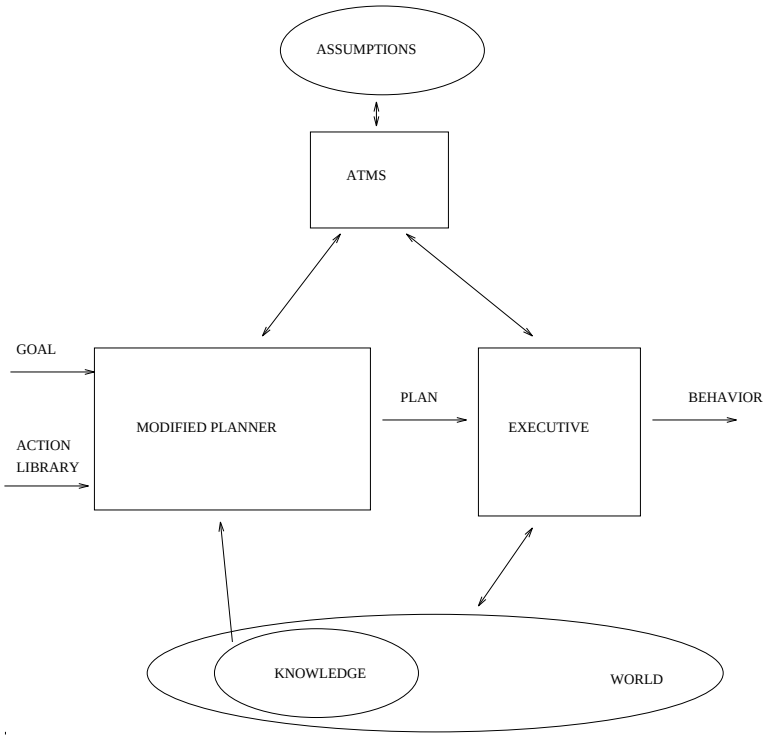


Figure 6.7: System Diagram

Planning system The planning component is modified from an existing hierarchical planner. There are two modifications which are required. First, the feasibility of an action is determined by the agent's state instead of the world state (by PLAN EVALUATION). Second, hierarchical plan expansion is modified to include expanding an abstract search into a more detailed search plan (by SCHEMA INSTANTIATION). Figure 6.8 depicts this organization of the functional component for planning.

Feasibility of actions is evaluated by comparing their preconditions to the agent's knowledge and beliefs. Actions may be known to be possible, known impossible, or have unknown feasibility. This last category, where the agent does not know the feasibility of an action, may be believed to be feasible or not (or the agent may have no beliefs as to the feasibility of the action).

Expanding an abstract search plan is accomplished by schema instantiation and recursive invocation of the planner.

Belief revision and plan execution Nonmonotonic belief revision is managed by an assumption-based truth-maintenance system (ATMS). Minimally, this system records assumptions and contradictions, and selects among alternative consistent extensions of the assumptions.

Plan execution is managed by combining an existing plan execution engine with a programming language interpreter. This interpreter has access to the world state and the agent state. Only agent state may be used to determine the value of conditional expressions in the plan. World state is used to determine the effects of behavior, and those effects then

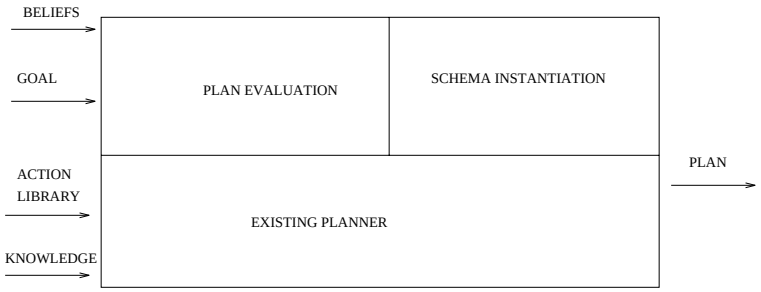


Figure 6.8: Planner System Diagram

become part of the world state.

6.2 Remaining open issues

Whether the complexity of the search planning problem requires the use of an ATMS for belief revision needs to be more fully explored. Alternatives to managing disjunctive contexts derived from the search space will be examined, including reasoning by cases as in a theorem prover.

How the truth maintenance system (if used) will be integrated in the search plan execution is yet to be completely specified. The options range from using it only to maintain consistency of beliefs to using it to control selection of the next search site by enforcing sequential selection of assumptions. In addition, various criteria for selecting among alternative next sites to explore and alternative actions for exploring the selected site will be examined.

A major topic of remaining research is determining the ordering on sites in the search space. What information about the search sites or the object determines which ordering will be used, and when that ordering is applied to determine which site to explore are questions which the dissertation must answer. The timing of site ordering and selection relative to other tasks (namely selecting an action to explore a site) must be determined.

Time permitting, the claim that this representation can be easily converted to an indexical representation will be demonstrated in the dissertation.

Beyond these technical details, the approach taken to search in this proposal is claimed to generalize to other kinds of search. The dissertation will demonstrate this generality by exploring the unmodified mechanism to other domains (such as the search problems sketched on page 2).

6.3 Remaining software development

Implementing the planning and execution engines for search plans requires modifying an existing planner and plan executing engine according to the design described in this proposal. An initial attempt at implementing limited search plans used the ItPlans planner in the AnimNL project [WBB+92]. To the extent, possible I will continue to use ItPlans as the underlying planner and plan execution engine which is modified to construct and execute search plans.

6.4 Proposed work

To support my thesis claims, I will construct search plans by schema instantiation with object descriptions, search space constraints, and appropriate actions for exploring the search space. I will build an execution engine for these search plans to support claims for improved execution.

For the thesis, I will explore the interaction of a belief revision system with a system for constructing and conducting search plans. I may use an existing belief revision system for this purpose, or implement a new one if necessary. Other ways of managing the selection of alternative sites to explore will also be examined.

The entire process of selecting sites to explore and selecting actions to explore those sites will be a focus of my remaining research. Information about what sites are likely to contain the object will be exploited by the planner to attempt to construct efficient searches.

To support my proposed architecture, I will examine the inherent computational complexity of the search planning problem and the complexity of the system architecture. I will compare the efficiency of my system to a system based on a replanning architecture which produces only sequential plans.

I will also clarify what inference is done at plan construction time and what (if any) inference must be done at plan execution time.

6.5 Conclusion

This proposal has identified the changes to the blocks world assumptions which naturally give rise to the search planning problem. A solution to planning for searches has been proposed which will permit agents to act effectively when faced with a need to find some object.

Appendix A

Representation for search plans

A.1 Syntax – first order modal logic

Individual variables	$\{v_1, v_2, v_3, \ldots\}$
Functions	$\{f_1^0, f_2^0, \ldots, f_1^1, f_2^1, \ldots\}$
Predicates	$\{p_1^0, p_2^0, \ldots, p_1^1, p_2^1, \ldots\}$
Terms	$T ::= v_i \mid f_i^n(t_1, \ldots, t_n)$ for $t_i \in T$
Propositions	$P ::= p_i^n(t_1, \ldots, t_n) \mid \neg P_i \mid P_i \wedge P_j \mid \forall v_i(P_j) \mid \Box_{\mathbf{Know}}(P_j) \mid \Box_{\mathbf{Perceive}}(P_j) \mid \Box_{\mathbf{Believe}}(P_j)$ for $t_i \in T$ and $P_i, P_j \in P$

Note that the action construction operations of the extended situation calculus are merely functions, and atomic action descriptions are 0-ary functions (constants). No new syntactic devices are required for the situation calculus.

Abbreviations I adopt the standard definitions for the remaining logical operators, disjunction ($\vee$), material conditional ($\rightarrow$), and biconditional($\leftrightarrow$). Existential quantification is the dual of universal quantification, $\exists x(\varphi)$ iff $\neg\forall x(\neg\varphi)$. Similarly, the diamond operator is the dual of the box operator, $\Diamond_R(\varphi)$ iff $\neg\Box_R(\neg\varphi)$.

I use the following common single letter abbreviations for temporal and epistemic modal operators:

K	$\Box_{\mathbf{Know}}$
B	$\Box_{\mathbf{Believe}}$

In addition I will use **See** as an abbreviation for $\Box_{\mathbf{Perceive}}$.

A.2 Semantics – possible worlds

A model for this logic is an 8-tuple

$$M = (W, \mathbf{Believe}, \mathbf{Know}, \mathbf{Perceive}, I, D, F, R)$$

where W is a set of worlds, **Believe**, **Know**, and **Perceive** are relations on W. I is a set of individuals. D is a mapping from elements w of W to subsets of individuals for that world D_w. F and R interpret functions and predicates.

My logic has a possible worlds semantics with a different world for each way an agent considers the world might be.

There are three epistemic accessibility relations. The **Know** accessibility relation is an equivalence relation. **Perceive** is reflexive and contains the **Know** relation, **Know**$\subseteq$**Perceive**. Similarly, **Know** contains the **Believe** relation, **Believe**$\subseteq$**Know**.[1]

A.2.1 Situation calculus

The situation calculus is a first order logic. Individuals may denote situations and actions. A function $do()$ is used to denote the situation which results from doing an action in a situation.

Action operators are provided which express control flow among several actions. A sequencing operator constructs a compound action from two actions.

$$s = do((a;b), s_0) \leftrightarrow s = do(b, do(a, s_0))$$

A conditional operator constructs a conditional branch action.

$$e(s_0) \rightarrow do(a, s_0) = do\left(\left(\begin{array}{c}\texttt{if } e(now)\\ \texttt{then}\\ a\\ \texttt{else}\\ b\end{array}\right), s_0\right)$$

$$\neg e(s_0) \rightarrow do(b, s_0) = do\left(\left(\begin{array}{c}\texttt{if } e(now)\\ \texttt{then}\\ a\\ \texttt{else}\\ b\end{array}\right), s_0\right)$$

An iteration operator constructs loops.

$$e(s_0) \rightarrow do\left(\left(\begin{array}{ll}\texttt{while } e(now)\\ \texttt{do}\\ \quad a\end{array}\right), s_0\right) = do\left(\left(\begin{array}{ll} & \texttt{while } e(now)\\ a; & \texttt{do}\\ & \quad a\end{array}\right), s_0\right)$$

$$\neg e(s_0) \rightarrow do\left(\left(\begin{array}{ll}\texttt{while } e(now)\\ \texttt{do}\\ \quad a\end{array}\right), s_0\right) = s_0$$

[1] As odd as this direction of containment may seem for the *accessibility relations*, it restricts models to those which support the axioms given later.

Note that *now* in the expressions of the conditional and iterative plans is interpreted as referring to the situation in which that part of the plan is executed (s_0 in these examples).

A.3 Inference – modal axioms

The inference system is presented as a Hilbert-style calculus. In addition to the inference rule of modus ponens, there is an inference rule of necessitation for each modal operator. The schematic presentation of this rule for the schematic modal operator (for an arbitrary accessibility relation R) $\Box_R$ is:

Schema 1 (Modus Ponens)

$$\psi, \psi \rightarrow \varphi \vdash \varphi$$

Schema 2 (Necessitation rule)

$$\vdash \varphi$$

implies that

$$\vdash \Box_R(\varphi)$$

Proofs about the effects of executing finitely many iterations of a search plan requires some form of induction. Section 3.2.1 discusses alternatives for the structure over which induction is performed. The two options considered there are the structure of time and the structure of the search space.

The remainder of the inference system is a collection of axioms and axiom schema.[2] Following pages discuss the various axiom collections in more detail.

A.3.1 Inferring properties of situations

Actions in the situation calculus have preconditions and effects. A predicate *Poss* of an action a and a situation s holds when the preconditions for that actions are satisfied in that situation. This is expressed as

$$Poss(a, s)$$

. For example, a precondition for the action of opening a container c is that the container is closed.

$$closed(c, s) \rightarrow Poss(open(c), s)$$

Effects of actions are described using successor state axioms. The idea of using successor state axioms is not a new one, and their particular form in this proposal is similar to that described in [Rei92]. Successor state axioms are used to infer for any action and any property of a situation, whether or not that property holds of the situation following that action. An example of a successor state axiom is shown for the property of container c being open:

$$Poss(a, s) \rightarrow open(c, do(a, s)) \leftrightarrow a = open(c) \vee a \neq close(c) \wedge open(c, s)$$

This can be glossed as stating that container c is open after doing some action a provided that action was either an *open* action, or it was not a *close* action and c was open before a. (Providing that action a was possible at the time.) Note that $open(c)$ is the action of opening container c and $open(c, s)$ is the property of container c in situation s.

The iteration and conditional action operators require expressions to be evaluated as either true or false. Each of these expressions corresponds to a formula in the situation calculus. The expression evaluates to true in a situation when the corresponding situation calculus formula holds of the situation. For example, the conditional expression

$$do\left(\left(\begin{array}{l}\texttt{if } in(b, c, now)\\ \quad \texttt{then}\\ \quad\quad first\\ \quad \texttt{else}\\ \quad\quad second\end{array}\right), s\right)$$

[2] Lowercase Greek symbols, like φ and ψ, are used in schemata to stand for arbitrary formulas.

will result in the agent taking action $first$ if $in(b, c, s)$ holds or action $second$ if $\neg in(b, c, s)$ holds.

A.3.2 Static Epistemic axioms

Static epistemic

Knowledge The S5 axioms are validated by the modal operator **K**.

1. $\mathbf{K}(\varphi \rightarrow \psi) \rightarrow \mathbf{K}(\varphi) \rightarrow \mathbf{K}(\psi)$
2. $\mathbf{K}(\varphi) \rightarrow \varphi$
3. $\mathbf{K}(\varphi) \rightarrow \mathbf{K}(\mathbf{K}(\varphi))$
4. $\neg\mathbf{K}(\varphi) \rightarrow \mathbf{K}(\neg\mathbf{K}(\varphi))$

Epistemic logic Since Hintikka [Hin62] modal operators have been used to represent epistemic attitudes of "knowing that" and "believing that". Frequently, the S5 or S4 axioms are selected to axiomatize the knowledge operator. The **Know** accessibility relation is reflexive under both these schemes, so the formulas known are always a subset of those actually true. An intuitive interpretation of the formula $\mathbf{K}(\varphi)$ is that in every possible world which the agent currently considers a valid alternative to the way the world actually is, φ is true. Another way to see this is that all the **Know**-related worlds are indistinguishable to the agent given what they know.

Common to this approach to formalizing knowledge is an extension to knowing the identity of objects represented as quantification into a modal context. Thus knowing the identity of some object uniquely describable as a is represented as

Schema 3

$$\exists x \mathbf{K}(x = a)$$

Perception Since the search plans I am concerned with involve the physical world, knowledge of successful search completion is typically obtained through the agent's perceptions. Search is not successful if the agent deduces the location of the object based on having opened all the containers but the last one, since the assumption that the object is in one of them may prove false. Since containers may be nested, "last" is really "last currently visible". Search is only successful when the agent sees the object. I adopt from Davis [Dav88] the principle that perceived facts are known, formalized as

5. $\mathbf{See}(\varphi) \rightarrow \mathbf{K}(\varphi)$

Belief

6. $\mathbf{K}(\varphi) \rightarrow \mathbf{B}(\varphi)$
7. $\mathbf{B}(\varphi) \rightarrow \neg\mathbf{B}(\neg\varphi)$

Belief is managed outside the system of logic presented here since it must be nonmonotonically revised (see Section 5.3). Beliefs are constrained to contain what is known and be consistent. Consistency is expressed by the **D** modal axiom (numbered 7 above).

A.3.3 Dynamic Epistemic axioms

Dynamic epistemic

8. $\mathbf{can}(\alpha, \varphi, s) \stackrel{def}{=} \mathbf{K}(\varphi(do(\alpha, s)))$

The definition of the modal operator **can** is given in Section 3.2 where reasoning about the agent's abilities to act effectively is discussed.

Bibliography

[All84] James Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[BP83] Jon Barwise and John Perry. *Situations and Attitudes*. Bradford Books, Cambridge, MA, 1983.

[CL90] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[DAD86] Richard Doyle, David Atkinson, and Rajkumar Doshi. Generating perception requests and expectations to verify the execution of plans. In *Proceedings AAAI*, pages 81–88, 1986.

[Dav88] Ernest Davis. Inferring ignorance from the locality of visual perception. In *Proceedings AAAI*, pages 786–791, St. Paul, MN, 1988.

[Dav92] Ernest Davis. Semantics for tasks that can be interrupted or abandoned. In *Artificial Intelligence Planning Systems*, pages 37–44, 1992.

[Dev91] Keith Devlin. *Logic and Information*. Cambridge University Press, 1991.

[Doy79] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[Dru85] Mark Drummond. Refining and extending the procedural net. In *IJCAI-85*, pages 1010–1012, 1985.

[Dru86] Mark Drummond. A representation of action and belief for automatic planning systems. In Georgeff and Lansky, editors, *Reasoning About Actions and Plans*, pages 189–211. Morgan Kaufmann, 1986.

[FN71] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Hin62] J. Hintikka. *Knowledge and belief*. Cornell University Press, 1962.

[Knu68] Donald E. Knuth. *The art of computer programming: sorting and searching*, volume 3. Addison-Wesley, 1968.

[Kor87] Richard E. Korf. Planning as search: A quantitative approach. *AI Magazine*, 33(1):65–88, 1987.

[KS86] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[KT90] Alex Kean and George Tsiknis. An incremental method for generating prime implicants/implicates. *Journal of Symbolic Computation*, 9:185–206, 1990.

[Lam84] L. Lamport. *Advanced Course on Distributed Systems – Methods and Tools for Specification*, volume 190 of *Lecture Notes in Computer Science*, chapter Basic Concepts. Springer, 1984.

[Les91] Yves Lesperance. *A Formal Theory of Indexical Knowledge and Action.* PhD thesis, Computer Systems Research Institute, University of Toronto, 1991. also Technichal Report CSRI-248.

[MGP60] George Armitage Miller, Eugene Galanter, and Karl H. Pribram. *Plans and the structure of behavior.* Holt, New York, 1960.

[MH69] John McCarthy and Pat Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[Moo80] Robert C. Moore. Reasoning about knowledge and action. Technical Report 191, SRI International, Menlo Park, October 1980.

[Moo84] Robert C. Moore. A formal theory of knowledge and action. In R.C. Moore and J. Hobbs, editors, *Formal Theories of the Commonsense World.* Ablex Publishing, Norwood NJ, 1984.

[Mor87] Leora Morgenstern. Knowledge preconditions for actions and plans. In *Proceedings IJCAI*, pages 867–874, Milano, Italy, 1987.

[Mor88] Leora Morgenstern. *Foundations of a Logic of Knowledge, Action, and Communication.* PhD thesis, Department of Computer Science, New York University, 1988.

[MP82] Z. Manna and A. Pneuli. Verification of concurrent programs: Proving eventualities by well-founded ranking. Technical Report STAN-CS-82-915, Dept. of Computer Science, Stanford, 1982.

[MR92] Sheila McIlraith and Raymond Reiter. On tests for hypothetical reasoning. Technical report, University of Toronto, 1992.

[OL82] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Programming Languages and Systems*, 4(3):455–495, 1982.

[Pra79] V. Pratt. Models of program logics. In *20th Symposium on foundations of Computer Science*, San Juan, October 1979.

[PS92] Mark A. Peot and David E. Smith. Conditional nonlinear planning. In *Aritificial Intelligence Planning Systems*, 1992.

[Rei92] Raymond Reiter. The projection problem in the situation calculus. In *Proceedings AIPS*, pages 198–203, 1992.

[RWC92] Ron Rymon, Bonnie L. Webber, and John R. Clarke. Progressive horizon planning – planning an exploratory-corrective behavior. *IEEE Transactions on Systems, Man, and Cybernetics*, To appear(Special Issue on Planning, Scheduling, and Control), 1992.

[Sac75] E. Sacerdoti. The non-linear nature of plans. In *Advance papers of IJCAI-75*, 1975.

[Sch92] Marcel Schoppers. Building plans to monitor and exploit open-loop and closed-loop dynamics. In *Aritificial Intelligence Planning Systems*, 1992.

[SW89] Devika Subramanian and John Woodfill. Making situation calculus indexical. In *Knowledge Representation and Reasoning*, pages 467–474, 1989.

[SW92] Ying Sun and Daniel S. Weld. Beyond simple observation: Planning to diagnose. In *Proceedings of the Third International Workshop on Principles of Diagnosis*, 1992.

[THD90] Austin Tate, James Hendler, and Mark Drummond. A review of AI planning techniques. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 26–49. Morgan Kaufmann, 1990.

[vB88] Johan F. A. K. van Bentham. *A Manual of Intensional Logic*. Number 1 in Lecture Notes. CSLI, Stanford, second edition, 1988.

[War76] D. Warren. Generating conditional plans and programs. In *Proceedings of the Summer Conference on AI and the Simulatino of Behavior*, Edinburgh, 1976.

[WBB+92] Bonnie Webber, Norman Badler, F. Breckenridge Baldwin, Welton Becket, Barbara Di Eugenio, Christopher Geib, Moon Jung, Libby Levison, Michael Moore, and Michael White. Doing what you're told: Following task instructions in changing, but hospitable environments. Technical Report ms-cis-92-74/linc lab 236, CIS, University of Pennsylvania, 1992.

[Wil83] D. E. Wilkins. Representation in a domain-independent planner. In *IJCAI-83*, pages 733–740, 1983.