Chapter 9

Plans and Goals

Groucho: You say you're going to go to everyone in this house and ask them if they took the painting. Suppose nobody in the house took the painting.

Chico: Go to the house next door.

Groucho: That's great. Suppose there isn't any house next door.

Chico: Well, then, of course, we got to build one.

Animal Crackers

At the bottom line, knowledge and reasoning are valuable insofar as they enable a creature to accomplish its goals through appropriate action. Finding such appropriate actions in a complex world often requires forethought; the creature must think through a course of action before executing it. Carrying out such forethought effectively under a range of circumstances requires the ability to represent and reasoning about one's plans and goals explicitly. Similarly, understanding or predicting the behavior of another intelligent creature requires reasoning about its plans and goals.[1] Accordingly, *plan construction*, the task of finding a plan to accomplish a goal, has been the most extensively studied application of commonsense reasoning. *Motivation analysis*, the task of inferring an agent's goals and plans from his actions, has also been much studied as a major component of understanding narrative text.

[1] These statements may seem to be truisms, but they have been debated. The behaviorists [Skinner 1971], of course, rejected them entirely. More recently, Agre and Chapman [1987] have argued that intelligent behavior consists largely of "situated activity," actions performed in direct response to situations, with little long-term planning.

Humans deal with many different types of plans and goals, and they reason about them in many different ways. The following examples illustrate some of these issues that a complete theory of plans and goals must address:

1. It is raining outside, and you have to bring a book home from the library without getting it wet. Infer that you can do this by carrying the book inside a plastic bag. This is a problem of plan construction. The goal is a conjunction of the physical state, "The book is at home," with the physical constraint, "At no time is the book wet." The plan is a description of a physical action. Note that an efficient planner will not plan the route home step by step; it will use general knowledge that it will be able to get home on foot while keeping the book protected.

2. Elly tells you that she is planning to travel to Bosnia via Herzogovinia. You know that the two countries went to war this morning and that there is no way of getting into Bosnia. Infer (a) that Elly does not know about the war; and (b) that Elly will have to drop or postpone her goal of going to Bosnia.

3. Joe is at his workshop. He has to build a desk, but his only record of the dimensions is at home. His customers, who are the only people who know the dimensions, are out of town. Infer that Joe will have to go home to get the dimensions. The problem here is to infer the actions of an agent from his goals. It is similar to plan construction, but, while the problem in plan construction is to find some plan satisfying the goal, here the problem is to characterize all plans that satisfy the goal. (All such plans involve going home.) The constraint here arises because of an informational requirement of a plan. In order to build the desk properly, he must know the desired dimensions.

4. The only thing that you know how to cook is oatmeal, and it is getting monotonous. Infer that one possible plan is to buy yourself a cookbook and learn some recipes. This is a plan-construction task. The goal has a complex structure: to satisfy your hunger at regular intervals in the future, subject to the constraint that the types of food vary sufficiently. To carry this out, it is necessary to know many different ways of preparing food. This requirement can be satisfied by using a cookbook.

5. Ed is driving at a leisurely pace up a one-lane highway. Matilda drives up close behind him and honks. Ed pulls over to the shoulder. Infer that Ed has inferred that Matilda is in a hurry and wants to pass, and that he has courteously cooperated. This is a problem

of motivation analysis: Given an agent's actions, infer his motives. Note that this problem involves reasoning about communication and cooperation between agents.

6. On a dark night, your horse refuses to cross a familiar bridge. Infer that the horse may sense that there is something wrong with the bridge. Here we are inferring something about the mental state of an agent in order to connect its actions with its goals. The goal-plan structure here has some particular points of interest. The underlying goal is one of preventing, or at least postponing, a state (death) rather than achieving one. The plan inferred consists of not doing an action rather than doing it. The plan is adopted in response to a particular circumstance (the bridge being washed out) rather than generated when the goal is adopted.

Most of the work on planning in AI has focused on problems of search, particularly the problem of finding a successful plan given a starting state and a goal. This search problem has generally been studied in the context of rather simple ontologies for plans, in order to concentrate on the search problems. By contrast, we will focus in this chapter on the ontological and representational issues that arise in reasoning about plans and goals.

9.1 Plans as Sequences of Primitive Actions

Many AI planners operate under the following assumptions:

1. The situation calculus model of time as a branching, discrete structure is appropriate. That is, only one primitive event occurs at a time, and the states of the world in the middle of events are unimportant.
2. There is only one agent, and the only events are his actions.
3. A goal is a desired state of the world.
4. All relevant aspects of the world are known to the planner from the start.

Under these circumstances, the execution of a plan will consist of the performance of a sequence of primitive actions. A plan is a complex event type. A *complete* plan description specifies a sequence of particular primitive actions; a *partial* plan description is a set of constraints on sequences of actions. A planning program is given a goal and a description of a starting situation; its task is to find a sequence

of actions that will be feasible in the starting situation and will terminate in a situation satisfying the goal.

Figure 9.1 illustrates three plans that accomplish the goal "Send Mr. Jones his bill." In plan A the sequence of actions is fully determined. In plan B the order is underdetermined; the printing of the bill can come before or after the printing of the address label, the sticking of the address label, or the stamping of the envelope. In plan C both the order of the actions and the binding of certain variables is underdetermined. A is a complete plan description; B and C are partial plans. B and C can be converted to complete plans by finding some total ordering obeying the partial ordering and some binding of the variables satisfying the restrictions.

There are two significant advantages to using partial plans. First, it may be worthwhile to leave certain decisions until execution time, since the best choice may be determined by minor considerations that do not arise until execution. For example, there is probably no point in deciding in advance which envelope or which label to use; one may as well just let the agent pick up the most convenient one. Second, in forming the plan, it is often easier to control a search through the space of partial plans than through a space of complete plans. By looking at partial plans, one can concentrate on constraints on the plan that are known to hold, thus avoiding arbitrary choices that are later found to be incorrect. Such a planner, which searches through a space of partial plans, is known as a "nonlinear" planner. We will present the details of one particular nonlinear planner in Section 9.2.

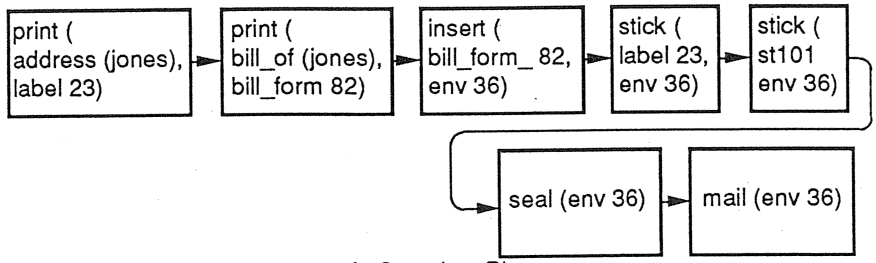
We introduce three fundamental predicates on plans and goals in this restricted model. A plan may be *feasible* in a given situation; it may *lead to* a given goal; and it may be a *valid* way to accomplish a given goal in a situation. These can be easily defined for complete plans using a branching temporal ontology. Plan P is feasible in situation S if P occurs in some interval starting in S. P leads to G if G is true following any execution of P. P is a valid plan to accomplish G in S if P is feasible and P leads to G.

PL.1. $\text{true\_in}(S,\text{feasible}(P)) \Leftrightarrow \exists_{S2}\ \text{occurs}([S,S2],P)$.

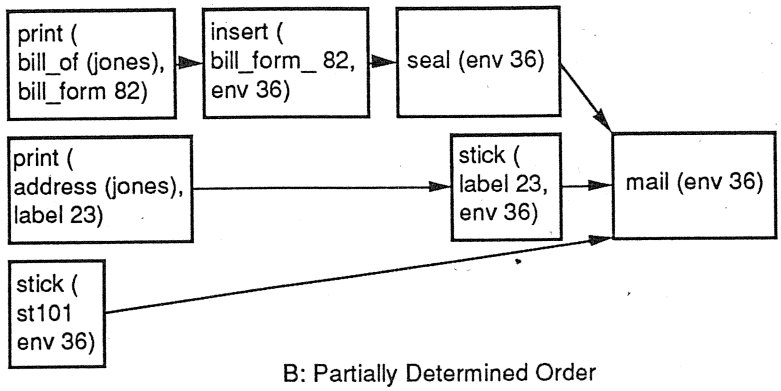
PL.2. $\text{true\_in}(S,\text{leads\_to}(P,G)) \Leftrightarrow$
$[\ \forall_{S2}\ \text{occurs}([S,S2],P) \Rightarrow \text{true\_in}(S2,G)\ ]$.

PL.3. $\text{true\_in}(S,\text{valid}(P,G)) \Leftrightarrow$
$[\text{true\_in}(S,\text{feasible}(P)) \wedge \text{true\_in}(S,\text{leads\_to}(P,G))]$.

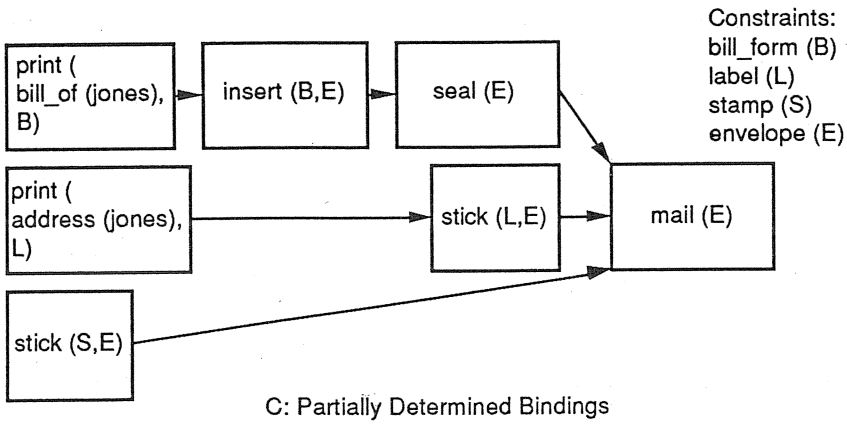
However, these definitions PL.1, PL.2, and PL.3 can lead to counterintuitive results when applied to partial plans. Consider, for example, the partial plan shown in Figure 9.2, consisting of two unordered



A: Complete Plan



B: Partially Determined Order



C: Partially Determined Bindings

Figure 9.1 Partially determined bindings

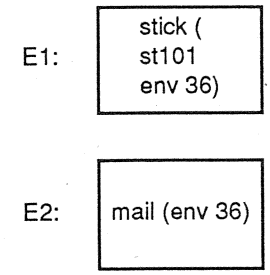


Figure 9.2 Invalid partial plan

events: ($E1$) Put stamp ST101 on envelope ENV36; ($E2$) Mail envelope ENV36. We would like to say that this is not a valid plan for the goal of having the envelope posted because if $E2$ is executed first, then there is no way to execute $E1$. However, it does satisfy our definition of validity; it can occur in the starting situation, and, if it does occur, then the goal is satisfied. In fact, if we view plans as event types and event types as sets of intervals, then this plan is exactly equivalent to the plan "First put the stamp on; then mail the envelope," since there are no intervals in which the reverse order occurs. In other words, axiom PL1 characterizes whether it is possible that events corresponding to the plan occur. With partial plans, this is not a sufficient condition to establish that the plan is feasible.

The simplest way around this difficulty is to change the way in which the causal theory treats impossible events. Rather than say that an event cannot occur if its preconditions are unsatisfied, we will say that if it does occur, it results in a distinguished impossible state, which we will call "twilight_zone." The following changes must be made to the causal theory to accommodate this approach:

- Replace each precondition axiom, "Event E can only occur if its preconditions are satisfied," with the weaker axiom "If E occurs and its preconditions are unsatisfied, then E leads to the twilight zone." For example, in the blocks-world axioms of Table 5.2, we would replace axiom BW13 by the new axiom

  [occurs(I,pickup) $\wedge$ $\neg$true_in(end(I),twilight_zone)] $\Rightarrow$
  [true_in(start(I),clear(hand)) $\wedge$
  $\exists_X$ true_in(start(I),under_hand(X))].

- For each causal axiom governing event type E, add the preconditions of E and the statement that the start state is not the twilight zone to the antecedents of the axiom, and add the statement that the final situation is not the twilight zone to its consequences. For example, axiom BW15 would be changed to

  [occurs(I,pickup) $\wedge$ true_in(start(I),clear(hand)) $\wedge$
  true_in(start(I),under_hand(X)) $\wedge$
  $\neg$true_in(start(I),twilight_zone)] $\Rightarrow$
  [true_in(end(I),beneath(hand,X)) $\wedge$
  $\neg$true_in(end(I),twilight_zone)].

- Add the rule that one cannot escape the twilight zone.

  true_in(start(I),twilight_zone) $\Rightarrow$ true_in(end(I),twilight_zone).

- Add the rule that the twilight zone cannot really come about.

  $S \in$real_chronicle $\Rightarrow$ $\neg$true_in(S,twilight_zone).

We can now modify our definition of feasibility to apply to partial plans in a more reasonable way: A plan is feasible if it does not lead to the twilight zone. (Note: For certain powerful representations of partial plans, this definition leads to counter-intuitive results.)

PL1.a. true_in(S,feasible(P)) $\Leftrightarrow$
[$\forall_{S2}$ occurs([$S, S2$],P) $\Rightarrow$ $\neg$true_in($S2$,twilight_zone)].

A complete plan can be verified by chaining forward in time, step by step. Beginning with the starting situation S, for each action of the plan in turn, verify that the preconditions of the action are satisfied in the situation where the action is to be done, and use the causal theory to predict the situation that will follow the action. When the effects of the last action have been computed, verify that the goal holds in the final situation.

Reasoning about partial plans can be more difficult. This is the subject of the next section.

9.1.1 TWEAK—a Nonlinear Planner

David Chapman's TWEAK program [1987] is a nonlinear planner. It is simple and somewhat limited in scope, but its construction is exceptionally clean and well analyzed. We present it here as an example of the use of nonlinear representations for plans in plan construction.

TWEAK uses a representation of action slightly modified from the STRIPS representation (see Section 5.7). A state type is represented either in the form $p(c_1 \ldots c_k)$ or in the form $\sim p(c_1 \ldots c_k)$, where p is a state function, $c_1 \ldots c_k$ are constants, and $\sim$ represents state negation. The only state-coherence axiom is that $p(c_1 \ldots c_k)$ and $\sim p(c_1 \ldots c_k)$ cannot hold simultaneously. The causal structure of an event type is defined by its effects and its preconditions, each of which is a set of state types. For example, the event type "puton(a,b,c)" (Put a onto b from c) would have effects { clear(c), on(a,b), ~clear(b) ~on(a,c)} and preconditions { clear(a), clear(b), on(a,c) }. An event template is like an event except that free variables may be used instead of constants. An event template may also contain constraints stating that two variables are unequal. For example, the event template "puton(X, Y, Z)" is defined to have effects { clear(Z), on(X, Y), ~clear(Y), ~on(X, Z) }, preconditions { clear(X), on(X, Z), clear(Y) }, and constraints { $X \neq Y$, $Z \neq Y$, $X \neq Z$ }.

The definition of the event template corresponds to a causal axiom that, if the preconditions and constraints hold in the starting situation, and the event takes place, then the effects will hold in the final situation. Also implicit is a frame axiom, stating that any state not on the effects list is not changed by the action. For example, the above definition of the event type "puton(X, Y, Z)" corresponds to the axioms

$$\forall_{S,S1,X,Y,Z}\ [\ \text{true\_in}(S,\text{clear}(X)) \wedge \text{true\_in}(S,\text{on}(X,Z)) \wedge \text{true\_in}(S,\text{clear}(Y)) \wedge X \neq Y \wedge Z \neq Y \wedge X \neq Z \wedge S1{=}\text{result}(S,\text{puton}(X,Y,Z))\ ] \Rightarrow [\text{true\_in}(S1,\text{clear}(Z)) \wedge \text{true\_in}(S1,\text{on}(X,Y)) \wedge \neg\text{true\_in}(S1,\text{clear}(Y)) \wedge \neg\text{true\_in}(S1,\text{on}(X,Z))].$$

$$\forall_{S,X,Y,Z,A}\ [\ \neg\text{true\_in}(S,A) \wedge \text{true\_in}(\text{result}(S,\text{puton}(X,Y,Z)),A)] \Rightarrow [A{=}\text{clear}(Z) \vee A{=}\text{on}(X,Y) \vee A{=}\sim\text{clear}(Y) \vee A{=}\sim\text{on}(X,Z)].$$

(In the second formula above, the variable A ranges over primitive state types and their negations.)

A trace of a plan is a linear sequence of variable-free events. Let AA_i be the set of states that hold at the start of event E_i; let EE_i be the set of effects of E_i, and let $\sim EE_i$ be the set of negations of states in EE_i. Then AA_{i+1}, the set of states that hold at the end of E_i, may be computed as $AA_{i+1} = (AA_i \cup EE_i) - \sim EE_i$. For uniformity of description, we treat the starting situation of a planning problem as the effects of an initial event *START* with no preconditions, and treat the goal as the preconditions of a final event *GOAL* with no

effects. A trace is feasible if every precondition of each event E_i is satisfied at its start; that is, each precondition of E_i is an element of AA_i.

A plan in TWEAK consists of a collection of partially or fully instantiated event templates, called *steps*. The plan structures the steps in a partial temporal ordering and records constraints on the bindings of the variables in the steps (Figure 9.3). A variable can be constrained to be equal or to be unequal to another variable or to a constant. Constants with distinct names are assumed to be unequal.

A plan subsumes a trace if the events in the trace correspond to the templates in the plan for some linear ordering consistent with the partial ordering of the plan and for some binding of variables consistent with the constraints in the plan. A plan is consistent if it subsumes at least one trace. (The ordering constraints must constitute a partial ordering, and the binding constraints must be consistent.) A plan is feasible if every trace subsumed by the plan is feasible. Given a consistent plan, it is possible to find a trace that it subsumes by finding bindings for the variables that satisfy the constraints, and by doing a topological sort on the steps. (We assume that all variables range over an infinite set of possible values. If so, it is trivial to satisfy the binding constraints, using a greedy algorithm. If variable bindings are restricted to a finite set, then the problem of satisfying the bindings becomes NP-complete, and the conditions for the correctness of the plan, below, are no longer necessary and sufficient.)

The central question in evaluating TWEAK plans, then, is what is needed to guarantee that a given precondition of an event is satisfied at its starting situation. In order to answer this, we introduce some additional technical terms. A step $E1$ *achieves* a state A under a binding if some effect of $E1$ is equal with A under the binding. $E1$ *necessarily* achieves A if it achieves A under all bindings consistent with the constraints. $E1$ *possibly* achieves A if it achieves A under some binding consistent with the constraints. A step $E1$ necessarily *establishes* a precondition A of step E if $E1$ necessarily achieves A, and $E1$ is constrained to occur before E. $E1$ possibly establishes a precondition A of step E if $E1$ possibly achieves A, and $E1$ is allowed to occur before E. A step $E1$ (necessarily / possibly) *clobbers* precondition A of step E if $E1$ (must / may) occur before E and $E1$ (necessarily / possibly) achieves the negation of A.

Chapman shows in his analysis that the following conditions are necessary and sufficient to guarantee that a precondition A of step E will be satisfied:

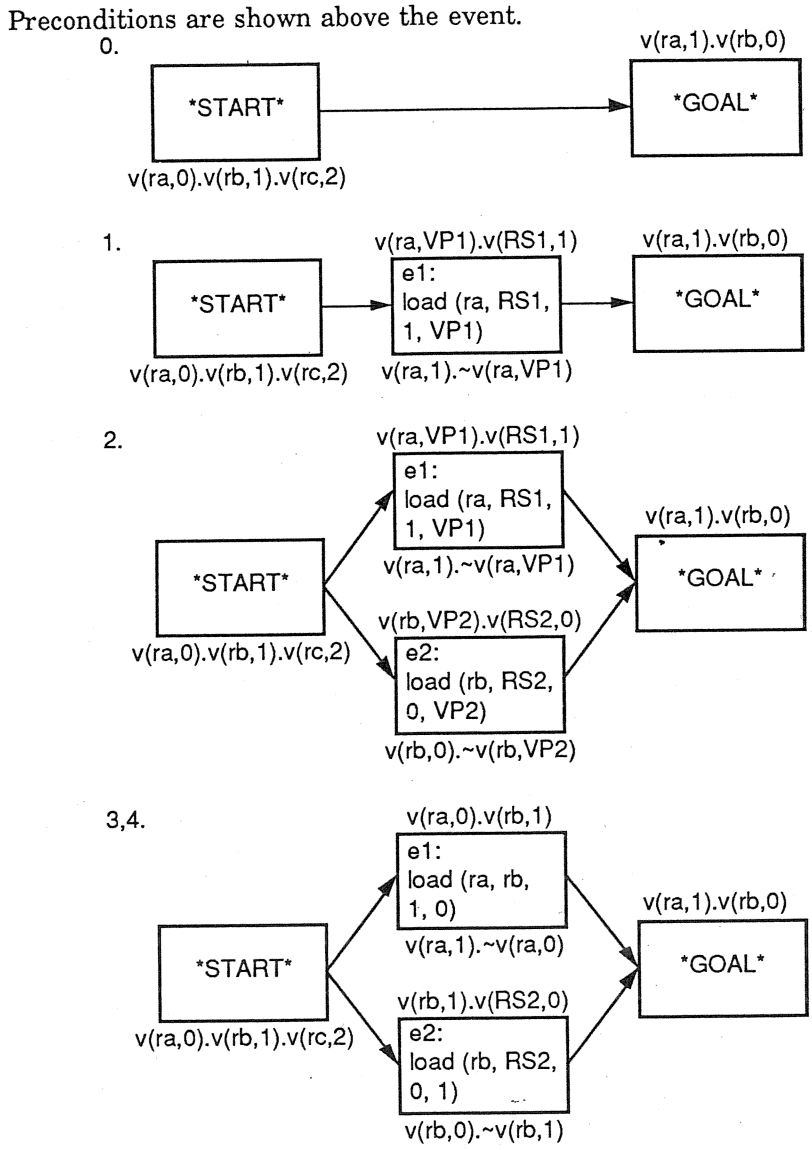


Figure 9.3 Successive states of the TWEAK planner

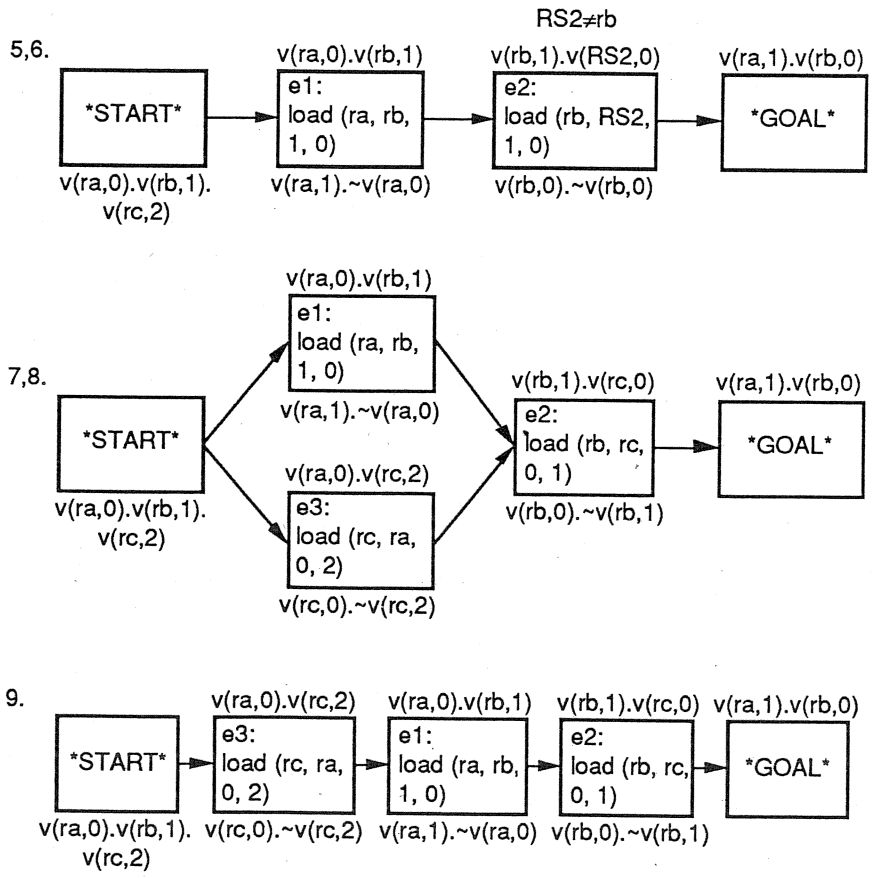


Figure 9.3 Successive states of the TWEAK planner (Continued)

Precondition A of step E is satisfied in all traces of the plan iff:
i.) There exists a step $E1$ that necessarily establishes A.
ii.) For each step C that possibly clobbers A, there exists a step W (called a *white knight*) that comes between C and E such that, for any allowable binding, if C achieves $\sim A$, then W achieves A.

Using this condition, Chapman presents the following algorithm for constructing TWEAK plans:

Algorithm 9.1: TWEAK Algorithm

Input: The effects of step *START* and the preconditions of step *GOAL*. *START* is constrained to precede *GOAL*.

Output: A feasible TWEAK plan including *START* and *GOAL*.

repeat until no changes occur in an iteration
  for each step E do
    for each precondition A of E do
      begin (1) if A is not established then either
        (1.a) Find some step $E1$ in the plan that possibly establishes A. Add temporal constraints so that $E1$ precedes E and binding constraints so that $E1$ necessarily achieves A; or
        (1.b) Find an event template $E1$ that possibly achieves A. Add $E1$ to the plan, constrained to precede E, with bindings constrained so that $E1$ achieves A;
      (2) for each possible clobberer C of A do either
        (2.a) constrain C to follow E; or
        (2.b) Add binding constraints so that C does not achieve $\sim A$; or
        (2.c) Find a step W in the plan that possibly achieves A. Constrain W to come between C and E. Add binding constraints so that W achieves A for every binding under which C achieves $\sim A$; or
        (2.d) Find an event template W that possibly achieves A. Constrain W to come between C and E. Add binding constraints so that W achieves A for every binding under which C achieves $\sim A$.

Chapman gives the following names to the various plan operators:

1.a: Simple establishment
1.b: Step addition
2.a: Promotion
2.b: Separation of variables
2.c: White knight insertion
2.d: White knight addition

For each precondition of every step, either (1.a) or (1.b) must be performed, if it is not established, and either (2.a), (2.b), (2.c), or (2.d) must be accomplished for each of its clobberers, if any. The choice between different possible modifications is performed nondeterministically, either by backtracking or by parallel search. (In Chapman's implementation, it was performed using breadth-first search with data-dependency maintenance, so as to achieve completeness and efficiency.) It can be shown that this algorithm is complete. That is, given any feasible trace T with no pointless steps — i.e., where every step satisfies some precondition of a later step — there is a plan subsuming T that the above algorithm will find for some choice of modification operators.

In general, there may be several ways of carrying out the operation "Add binding constraints so that W achieves A for every binding under which C achieves $\sim A$" in (2.c) and (2.d). For example, suppose that A is the state p(a), that C achieves $\sim$p(X), and that W achieves p(Z). The condition can be achieved either by adding a constraint that Z =a or by adding a constraint that $X = Z$. (It can also be achieved by adding a constraint that $X \neq$a, but that case reduces to separation of variables; the white knight is unnecessary.)

For example, consider register swapping, a problem that is beyond the capacity of the STRIPS planning system. The primitive states in this system have the form "value(R, V)", meaning that register R has value V. The events have the form "load(RD, RS, VN, VP)", the action of loading new value VN from source register RS to destination register RD, overwriting the previous value VP in RD. The preconditions of the event "load(RD, RS, VN, VP)" are { value(RS, VN), value(RD, VP) }. The effects of the event are { value(RD, VN), $\sim$value (RD, VP) }. There are constraints $RS \neq RD, VP \neq VN$. Let *START* have the effects { value(ra,0), value(rb,1), value(rc,2) } and let *GOAL* have the preconditions { value(ra,1), value(rb,0) }. Table 9.1 shows one series of operations that brings TWEAK to success (Figure 9.3).

Note that the only links between components of this plan are binding constraints between variables and precedence relations between steps. In particular, the concept of one state being a *subgoal* of another plays no significant part in this view of planning.

Table 9.1 Register Swapping in TWEAK

1. To achieve the precondition value(ra,1) of *GOAL*, add the step e1=load(ra,$RS1$,1,$VP1$) between *START* and *GOAL*. (Step addition)

2. To achieve the precondition value(rb,0) of *GOAL*, add the step e2=load(rb,$RS2$,0,$VP2$) between *START* and *GOAL*. (Step addition)

3. To achieve the preconditions value($RS1$,1) and value(ra,$VP1$) of e1, use the effects of *START* by binding $RS1$ to rb and $VP1$ to 0. (Simple establishment)

4. To achieve the precondition value(rb,$VP2$) of e2, use the effects of *START* by binding $VP2$ to 1. (Simple establishment)

5. The effect ~value(rb,1) of e2 is now a potential clobberer of precondition value(rb,1) of e1. Therefore, promote e2 to come after e1. (Promotion)

6. The effect ~value(ra,0) of e1 now potentially clobbers the precondition value($RS2$,0) of e2. Therefore, impose the constraint $RS2 \neq$ra.

7. To establish the precondition value($RS2$,0) of e2, add the step e3=load($RS2$,$RS3$,0,$VP3$) before e2. (Step addition)

8. To achieve the preconditions value($RS2$,$VP3$) and value($RS3$,0) of e3, use the effects of *START* with the bindings $RS2$=rc,$VP3$=2, $RS3$=ra. (Simple establishment)

9. The effect ~value(ra,0) of e1 now potentially clobbers the precondition value(ra,0) of e3. Therefore, promote e1 to come after e3. (Promotion)

The plan is now complete. It contains the three actions, load(rc,ra,0,2), load(ra,rb,1,0), load(rb,rc,0,1), in sequence.

9.2 Extensions

We now consider a number of simple extensions to the theory developed so far. These extensions involve only fairly straightforward changes to the ontology, the representation, or the axiomatics of planning, but they may make the search problem very much more difficult.

Goals of maximizing an objective: Many natural goals, such as "Build a tower as high as possible" or "Make as much money as possible," have no fixed criterion of success, but can be achieved to greater and lesser degree. Such a goal may be represented by a fluent rather than a state. A plan achieves a goal to degree D if the goal has value D when the plan is complete. We replace the Boolean state "valid(P, G)" with the fluent "success(P, G)". Formally, P is guaranteed to succeed to degree D in goal G in situation S if D is the minimum value of G after P occurs starting in S. We replace axiom PL2 above by the new rule:

PL2A. value_in(S,success(P, G)) $\geq D \Leftrightarrow$
[true_in(S,feasible(P)) $\wedge$
$\forall_I$ S=start(I) $\wedge$ occurs(I, P) $\Rightarrow$ value_in(end(I),G) $\geq D$].

Many complex goals can be handled in this format, by encoding the entire goal in a single complex fluent. For example, the goal "Bring all the blocks to $L1$ as quickly as possible" can be expressed as maximizing a fluent whose value is $-\infty$ if the blocks are not all at $L1$, and $-$clocktime if the blocks are all at $L1$.

Interaction with external events: Events other than the agent's own actions may be relevant to the achievement of the goal. These external events may either aid or hinder the agent. Consider, for example, the following plan for preparing tea:

Fill the kettle with water;
Put the kettle on the stove;
Turn on the burner;
Get the tea cup from the shelf;
Put the tea bag in the cup;
Turn off the burner;
Put on a kitchen glove;
Pour the water from the kettle into the tea cup.

The success of this plan depends on the event of the water becoming hot while it is on the stove. The need to put on a glove before pouring the water is due to the event of the kettle becoming hot. Representing such plans requires a model in which external events can occur.

Once external events are admitted, it becomes important for an agent to distinguish his own actions, which are dependent on his will, from other events, which are not. A plan can contain only the agent's own actions; these may give rise to other significant events, such as the water becoming hot in the above example. (Multiagent plans will be discussed in Section 10.2.) We introduce the function "actor_of(E)" mapping an event E onto the agent, if any, who performs E. The actions of agent A are often represented in the form "do(A, ACT)," where ACT is an action type. For example, the term "do(linda,puton(kettle1,stove8))" would be the event of Linda putting the kettle on the stove. We have the general axiom "actor_of(do(A, ACT)) = A".

An integral part of such plans is waiting for external events to bring about a desired result. We introduce three waiting actions: "wait_until(Q)," the action of waiting until state Q becomes true; "wait_while(E)," the action of waiting while event E takes place; and "wait(T)," the action of waiting for a time duration of length T. These may be defined as follows:

occurs(I,do(A,wait_until(Q))) $\Leftrightarrow$
[$\forall_{S \in I}$ true_in(S, Q) $\Leftrightarrow$ S=end(I)].

occurs(I,do(A,wait_while(E))) $\Leftrightarrow$
$\exists_{I1}$ end($I1$)=end(I) $\wedge$ occurs($I1$, E).

occurs(I,do(A,wait(T))) $\Leftrightarrow$
value_in(end(I),clock_time) $-$ value_in(start(I),clock_time) = T.

These axioms do not by themselves specify that the actor do anything or abstain from doing anything while he is waiting. Therefore, to reason deductively about the effect of a plan containing a "wait" action, it will generally be necessary to add axioms that assert that the only actions executed by the agent concurrently with a "wait" are those specified as concurrent in the plan. (See Section 5.11 for a discussion of how this is done.)

Goals over chronicles: Some natural goals, such as "Eat a turkey dinner," "Travel by boat to the Orient," or "Talk to everyone at the party," are not states or fluents that hold in a single final situation, but, rather are characteristics of a whole chronicle or interval. Such goals can be viewed as complex event types. We assert that goal G is accomplished in interval I in the formula "occurs(I, G)." Therefore, a plan P is valid for goal G in situation S if P is feasible in S and, for each interval I starting in S in which P occurs, there is an interval $I1$ with the same ending time as I in which G occurs.

P2B. true_in(S,valid(P, G)) $\Leftrightarrow$
[true_in(S,feasible(P)) $\wedge$
$\forall_I$ [start(I) = $S \wedge$ occurs(I, P)] $\Rightarrow$
$\exists_{I1}$ [end($I1$)=end(I) $\wedge$ occurs($I1, G$)].

A planner that reduces a goal involving complex events to a plan consisting of primitive events is known as a *task-reduction* planner. Such a planner constructs plans by combining task-reduction rules — rules stating how one action can be carried out in terms of other actions — with the techniques discussed in Section 9.1 for achieving precondition states. For example, the task "Give A a medical checkup" can be reduced to the conjunction of the tasks "Check A's temperature," "Check A's blood pressure," "Check A's weight," with no particular temporal ordering (Table 9.2). Each of these subtasks can be further reduced; for example "Check A's temperature" can be reduced to the sequence of steps "Sequence: (1) Place thermometer in A's mouth; (2) Wait three minutes; (3) Remove thermometer from mouth; and (4) Read temperature from thermometer." These substeps have preconditions and effects that the planner must reason about in the same way as the state-achievement planners discussed in Sections 9.1 and 9.2. For example, the step "Place thermometer in A's mouth" has preconditions that both the thermometer and A are available. It has the effect that A's mouth is occupied by a thermometer. The planner must be able to reason that it is not possible to have the patient drink or speak in between steps (1) and (3) of the temperature taking.

The operation of task reduction starts with relatively abstract descriptions of tasks and gradually makes the descriptions more concrete, ending with primitive robotic operations. However, the planner cannot as a whole consistently move down through levels of abstractions because achieving a precondition to a concrete goal may involve much planning at more abstract levels. Consider, for example, the planning involved in achieving the concrete goal "Walk on the moon" or, often, in achieving the goal "Make love to X."

Note that once planning is complete execution can proceed using only the primitive operations at the bottom level of the reduction; no reference to the higher-level tasks is needed. The task-reduction structure is now needed only in case of something unexpected occurring, which requires replanning. (See Section 9.3.3.)

Goals of prevention: In a world with external events, many of the most important goals are those of preserving a state rather than achieving it, or, more generally, of preventing a harmful state or event. The archetype of such goals is the goal of avoiding death or destruction (strictly speaking, this goal is one of postponement rather than of pre-

Table 9.2 Task-Reduction Axioms

Define the predicate "occurs_in(I, E)" (E occurs some time during I) by the axiom

$$\text{occurs\_in}(I, E) \Leftrightarrow \exists_{I1 \subset I} \text{occurs}(I1, E).$$

$$\begin{aligned}
&\text{occurs_in}(I,\text{checkup}(A)) \Leftrightarrow \\
&[\ \text{occurs_in}(I,\text{temperature_check}(A)) \wedge \\
&\ \ \text{occurs_in}(I,\text{blood_pressure_check}(A)) \wedge \\
&\ \ \text{occurs_in}(I,\text{weight_check}(A))\ ].
\end{aligned}$$

$$\begin{aligned}
&\text{occurs}(I,\text{temperature_check}(A)) \Leftrightarrow \\
&\exists_{TH}\ \text{thermometer}(TH) \wedge \\
&\quad \text{occurs}(I,\text{sequence}(\text{insert}(TH,\text{mouth}(A)), \\
&\qquad\qquad\qquad \text{wait}(3 \cdot \text{minute}), \\
&\qquad\qquad\qquad \text{remove}(TH,\text{mouth}(A)), \\
&\qquad\qquad\qquad \text{read_temperature}(TH))).
\end{aligned}$$

vention). Formally, these are a type of goal over chronicles; however, it is worthwhile to consider them independently on account of their frequency and importance. Some of these goals, such as "Stay alive," are always present, and plans are constructed to guarantee them whenever it seems that they will be jeopardized by the expected course of events. Others, such as "Avoid going to sleep (during a lecture)" are temporally limited; they arise in response to certain circumstances, and may disappear after time. The axioms that govern these goals are, on the whole, more concerned to describe the circumstances that threaten the goals and the actions that remove these threats, rather than the actions that achieve the goals and their preconditions.

Resource and timing constraints: Plans must often be carried out within constraints on resources and time. For example, carrying out a building project might involve constraints such as "The cost for supplies must not exceed $700," "At any instant, the total electric power used by active machines must not exceed 1250 watts," "It is necessary to wait a day between applying coats of paint to an object," "Power tools can only be used in the daytime," and so on. Formally, these can all be expressed as properties of the chronicle involved so that these constraints can all be incorporated as part of a goal over a chronicle. Computationally, even very simple constraints of this kind tend to

make the difficulties of finding a satisfactory plan much greater. Techniques for dealing with such constraints have been studied within operations research; the problem of incorporating these techniques into AI planners is a subject of current research.

Repeated goals: If it is expected that a goal, or a collection of similar goals, will often be repeated in the future, it may be worthwhile performing a relatively expensive operation that simplifies the performance of all the goals together. For example, an agent who knows that he will have to travel 30 miles every day may decide to invest in a car. Such planning is known as *goal subsumption* [Wilensky 1978].

Concurrent actions: If an agent is capable of performing more than one action at once, then he should be able to take advantage of this capacity in his plans. However, unless the actions are physically quite independent, it is not, in general, possible to predict either the feasibility or the effects of performing two actions together from knowing their properties singly. Rather, a rich physical model must be used that describes the interactions of the two activities. Little work has been done on such models.

9.3 Plans and Goals as Mental States

In the previous section we considered plans and goals purely as abstract physical constructs. Such a view is appropriate as a model for a single agent who is given a single goal from on high, and who must find a plan to accomplish his goal using just the information in his knowledge base. To go beyond this limited scenario, we must view plans and goals as mental constructs: aspects of an agent's mental state. Such a view will allow an intelligent creature to represent facts about the plans and goals of other creatures, and about his own plans and goals at different times. Such representations are needed to express theories that address questions like the following:

- How are beliefs and knowledge related to plans and goals? In particular, what information is required in order to carry out a specified plan?
- What is the life cycle of a goal or a plan? How are goals and plans adopted, maintained, achieved, modified, or abandoned?
- What goals are characteristic of humans?

The first problem in formalizing plans and goals as mental states is that, like knowledge and belief, having a plan P or a goal G is not an

extensional operator. In particular, the goal and plan operators are referentially opaque. Consider the following three sentences:

1. Oedipus has the goal of killing the traveler who is attacking him.
2. The traveler attacking Oedipus is his father.
3. Oedipus does not have the goal of killing his own father.

It is possible for all three statements to be true, as long as Oedipus does not know that the traveler is his father.

We will get around these problems by treating goals and plans as operators over quoted strings. We introduce three primitives. The predicate "goal(A, G, S)" means that, in situation S, agent A has the goal denoted by the quoted string G. The predicate "plan(A, P, S)" means that in situation S, A plans to carry out plan P. The function "deliberate(A, E)" denotes the event of agent A carrying out the action described by string E deliberately. (Like "plan" and "goal", "deliberate" is an opaque operator. Oedipus deliberately killed the stranger but did not deliberately kill his own father, even though the two were the same.) Thus, the statement that, in situation s0, Oedipus had the goal of being king of Thebes is represented

goal(oedipus,≺eql(oedipus,king_of(thebes))≻,s0)

The statement that Oedipus had the goal of knowing who killed Laius is represented

goal(oedipus,
≺{ S | $\exists_X$ know(oedipus, ≺$\exists_I$ end(I) < s0 ∧
occur(I,kill(@X@,laius))≻, S)
}≻,
s0).

(Note: The @ signs around X in the above formula are scoped to the inner set of string delimiters.)

The statement that Oedipus deliberately married Jocasta is represented as

occur(i0,deliberate(oedipus,≺do(oedipus,marry(jocasta))≻)).

An alternative solution to the problem of referential opacity is to treat a goal as a state ranging over possible worlds, and to treat a plan or a deliberate action as an event type ranging over possible chronicles. Thus, the statements "Oedipus deliberately married Jocasta," "Oedipus did not deliberately marry his mother," and "Jocasta was Oedipus' mother," are represented

$\exists_I$ occur(I,deliberate(oedipus,do(oedipus,marry(jocasta)))).
$\neg\exists_I$ occur(I,deliberate(oedipus,do(oedipus,
marry(mother_of(oedipus))))).
jocasta = value_in(w0, mother_of(oedipus)).

These three statements are mutually consistent because the "mother_of" function is made dependent on the possible world; Jocasta is Oedipus' mother in this world, but not in every possible world.

These primitives, together with the primitives of knowledge and belief, are the basic concepts in the theory of plans and goals as mental states. We will develop this theory in four parts. The first part describes what an agent knows or believes about his own plans and goals. The second describes what an agent needs to know in order to carry out his plans. The third describes how an agent gains, carries out, and abandons goals and plans. The last part of the theory describes what goals a human agent is likely to have.

9.3.1 Knowledge of Plans and Goals

A number of plausible axioms governing an agent's knowledge of his own plans and goals may be proposed:

KPG.1. Positive introspection: If A has a plan or a goal then he knows about it. If A performs a deliberate action, then he knows that he has done it when it is complete.
(a) goal(A, G, S) $\Rightarrow$ know(A,$\prec$goal(@A@, !G!, @S@)$\succ$,S)
(b) plan(A, P, S) $\Rightarrow$ know(A,$\prec$plan(@A@, !P!, @S@)$\succ$,S)
(c) occur(I,deliberate(A, E)) $\Rightarrow$
know(A,$\prec$occur(@I@, deliberate(@A@, !E!))$\succ$,end(I)).

KPG.2. Negative introspection: If A does not have a plan or a goal then he knows that he doesn't. If A does not perform a deliberate action, then he knows that he has not performed the action deliberately.
(a) $\neg$goal(A, G, S) $\Rightarrow$ know(A,$\prec\neg$goal(@A@, !G!, @S@)$\succ$,S)
(b) $\neg$plan(A, P, S) $\Rightarrow$ know(A,$\prec\neg$plan(@A@, !P!, @S@)$\succ$,S)
(c) $\neg$occur(I,deliberate(A, E)) $\Rightarrow$
know(A,$\prec\neg$occur(@I@, deliberate(@A@, !E!))$\succ$,end(I)).

KPG.3. A can deliberately perform only his own acts. (Of course, he can deliberately *trigger* other events, but only by performing an act of his own.)
occur(I,deliberate(A, E)) $\Rightarrow$ A=actor_of(denotation(E)).

(Keep in mind that "deliberate" takes a string as argument, while "actor_of" takes an actual event as argument.)

KPG.4. The deliberate performance of an action is an occurrence of the action.

$$\text{token\_of}(K,\text{deliberate}(A,E)) \Rightarrow \text{token\_of}(K,\text{denotation}(E)).$$

KPG.5. If A plans to perform P, then he believes that he will deliberately perform P.

$$\begin{aligned}&\text{plan}(A,P,S) \Rightarrow\\ &\text{believe}(A,\prec\exists_I\ \text{precede}(@S@,\text{start}(I)) \wedge\\ &\qquad\qquad \text{occur}(I,\text{deliberate}(@A@,!P!)) \wedge\\ &\qquad\qquad I \subset \text{real\_chronicle}\succ,\\ &\qquad S).\end{aligned}$$

KPG.6. If A plans to perform P, then A believes that P will be a valid plan to accomplish one of his goals.

$$\begin{aligned}&\text{plan}(A,P,S) \Rightarrow\\ &\text{believe}(A,\ \prec\exists_{G,S1}\ \text{precede}(@S@,S1) \wedge\\ &\qquad\qquad\qquad \text{goal}(@A@,G,S1) \wedge\\ &\qquad\qquad\qquad \text{true\_in}(S1,\text{valid}(@A@,\downarrow P\downarrow,G))\succ,\\ &\qquad\qquad S).\end{aligned}$$

KPG.7. Knowledge of the axioms: If ϕ is an instance of one of the axioms in this chapter, and P spells out ϕ, then "know(A,P,S)" is an axiom.

Axioms KPG.1 and KPG.2 state that an agent knows what are and are not his own goals, plans, and deliberate actions. These are plausible as long as we exclude unconscious goals from consideration. This exclusion is justifiable since robots presumably do not have two levels of goals, and the effect of humans' unconscious goals on their behavior and mental states is hard to characterize in any theory, let alone a formal one. See Section 10.3.3 for an example of a proof that uses KPG.2 on deliberate actions. Axioms KPG.3 and KPG.4, that an agent is the actor of his own deliberate actions and that the deliberate performance of an action is an occurrence of that action, are basic necessary properties of deliberate actions. Axiom KPG.5 states that an agent believes that he will deliberately execute his plans. The converse, that any deliberate action of the agent is part of some plan, likewise seems plausible, and is important for motivation analysis. However, it seems to be tricky to state this rule correctly; it would not be correct to say that if A performs E deliberately, then E is part of some plan P that A is performing deliberately, since A may end up not being able to execute later parts of P. Axiom KPG.6, that an agent only adopts plans that he believes to be valid for some future goal, is obviously an approximation. A more accurate statement would be that the agent

believes that the plan has some reasonable chance of forwarding his goals as a whole; however, this would be difficult to represent. Note that we have added the agent A as an argument to the state function "valid." Similarly, we will add the agent A as an argument to "feasible." Axiom KPG.7, that an agent knows the axioms of plans and goals, allows us to infer that the agent can reason about plans and goals; it is analogous to axiom KNOW.4 from Chapter 8.

These axioms can be used to justify inferences like example 2.a in the chapter introduction, in which we infer that Elly believes that the border between two countries will be open from the fact that she is planning to do so. Table 9.3 shows a precise statement of the inference.

9.3.2 Knowledge Needed for Plan Execution

In many cases, an agent is initially ignorant of information that he needs to achieve a given goal, but he knows how to acquire the information. In that case, he may plan to acquire the necessary information and then to use that information in further steps of planning. For example, if Debby wants to read *Moby Dick* but does not know where her copy is on her bookshelf, then she can form the plan "Look through the bookcase; take the book from its place; read it." Here the purpose of the first step of looking though the shelf is to determine the place of the book, a datum needed for the second step of the plan, grasping the book.

A theory of such plans must address the following issues:

- What information is necessary to carry out a primitive action? This is known as the *knowledge-preconditions* problem for actions.[2]
- What information is necessary to carry out a complex plan? This is the knowledge-preconditions problem for plans.
- What actions of the agent provide him with information? This is the information-acquisition problem.

In this section we will discuss the knowledge-preconditions problems for actions and plans. We have considered information acquisition through perception in Section 8.7; we will discuss information acquisition through communication in Section 10.3.

It should be noted, at the outset, that the failure of a knowledge precondition has different consequences than a failure of a physical

[2]This term was introduced in [McCarthy and Hayes 1969]

Table 9.3 Inferring Beliefs from Plans

Given:

- In situation s0, Elly plans to cross the border from Bosnia to Herzogovina.
  plan(elly,$\prec$cross(elly,bosnia,herzogovina)$\succ$,s0).
- Elly believes that it is only possible to cross from X to Y if the borders are open.

  believe(elly, $\prec\forall_{S1,A,X,Y}$ true_in($S1$,feasible(cross(A,X,Y))) $\Rightarrow$ true_in($S1$,open(border(X,Y)))$\succ$,s0).

Conclude:

- Elly believes that the border between Bosnia and Herzogovina will be open at some future time.
  believe(elly,$\prec\exists_{S1}$ precede(s0,$S1$) $\wedge$
  true_in($S1$, open(border(bosnia,herzogovina)))$\succ$,s0).

Proof: From axiom KPG.5, Elly believes that she will eventually cross from Bosnia to Herzogovina deliberately. By axiom PL.1, this can only occur if crossing is feasible; and by axiom KPG.7, Elly knows that it can only occur if it is feasible. By hypothesis, Elly believes that the crossing is feasible only if the border is open. Using axioms KPG.7, KPG.4, KB.1, and consequential closure on belief (BEL.1), it follows that Elly believes that, at the time when she crosses, the border will be open.

precondition, and the logical treatment must therefore take a different form. If a physical precondition to an event is not satisfied in a situation, then the event cannot occur. If the event is perceived to occur, then we can infer that the physical precondition was satisfied at the start. If the knowledge preconditions of an action are not satisfied, the action may still occur, though it cannot be deliberately performed. For instance, an agent who does not know which U.S. city is largest can nonetheless perform the action of going to the largest city, but he cannot deliberately go to the largest city. Thus, knowledge preconditions, like deliberate actions, are properties of action descriptions rather than the actions themselves.

We will use the predicate "kp_satisfied(A, P, S)," meaning that the knowledge preconditions for action or plan description P (a quoted string) are satisfied for agent A in situation S. An agent can perform an action deliberately only if its knowledge preconditions are satisfied.

KPS.1. occurs(I,deliberate(A, E)) $\Rightarrow$ kp_satisfied(A, E,start(I)).

Two general types of axiomatizations have been proposed for knowledge preconditions for primitive actions. The first is simply to enumerate the knowledge preconditions for each type of action description. We would express the statement "To move to a place described as P, one must know where the place is" in the form

kp_satisfied(A,$\prec$travel_to($\downarrow P\downarrow$)$\succ$,S) $\Leftrightarrow$ know_val(A, P, S).

The statement "To grasp the object denoted by string O, one must know where that object is located" can be expressed in the form

kp_satisfied(A,$\prec$grasp($\downarrow O\downarrow$)$\succ$,S) $\Leftrightarrow$
know_val(A, $\prec$value_in(@S@,place($\downarrow O\downarrow$))$\succ$, S).

A second solution is that an agent knows how to perform a primitive action E if E is "directly executable"; that is, E can be used as a direct call to a robotic control system. For example, if the agent can execute an action routine "tap(N)", to tap the ground N times, then the action "tap(14)" is directly executable, and the agent knows how to do it. An agent A knows how to perform an action description E that is not in the form of a direct robotic call if he knows that E can be carried out by doing $E1$, where $E1$ is directly executable. For example, A knows how to perform the action e1 = tap(cardinality({ S | planet(S) })) if he knows that e1 is carried out through the action "tap(9)," but not if he is unsure whether e1 is the action "tap(9)" or "tap(5)". In general, agent A knows how to perform a primitive action described as $\prec$f(t1 ... tk)$\succ$ if f is a primitive routine for A and A knows the value of t1 ... tk.[3]

KPS.2. kp_satisfied(A,$\prec\downarrow ACT\downarrow$($\downarrow T1\downarrow$,... $\downarrow Tk\downarrow$)$\succ$,S) $\Leftrightarrow$
primitive_routine(ACT, A) $\wedge$
know_val($A, T1, S$) ... know_val(A, Tk, S).

Thus, we have reduced the problem of knowledge preconditions for action to the problem of knowing the values of terms. This solution

[3]This account follows [Morgenstern 1988]. The theory was originally proposed by Moore [1980] in the context of a possible-worlds theory of knowledge. In that context, an executable description of an action was considered to be a rigid designator for the action. The proposal in [McCarthy and Hayes 1969] pointed in a similar direction.

eliminates the need for many specialized knowledge precondition axioms: the knowledge preconditions can be derived directly from the syntactic form of the action description. It also provides an intuitive justification of the knowledge preconditions; they require just that the actor know precisely what action it is that he wants to do. The solution has its costs, however. First, knowing the value of a term is generally a less precise concept than knowing enough to perform an action; to a degree, we have reduced a relatively well-defined problem to a much vaguer one. Second, as we have seen in Section 5.2, the concept of a robotic primitive is not absolute. There are levels of descriptions of actions, and a primitive at one level may involve a number of steps, including the gathering of knowledge, at a lower level. This analysis of knowledge preconditions, therefore, must be considered as relative to a given level of robotic primitives. (The dependence of the representation on the level chosen for robotic primitives is an implicit issue throughout the analysis of planning, but it appears in a particular direct form here.)

Waiting actions require a separate definition. The knowledge precondition for "wait_until(A)" is that the robot knows that A will eventually hold and that he will know when it holds. The knowledge precondition for "wait_while(E)" is that the robot knows that E will eventually occur and that he will know when it ends. The action "wait(T)" is handled by axiom KPS.2; the knowledge precondition for "wait(T)" is that the value of T be known. (We assume that the robot has an internal clock.)

KPS.3. kp_satisfied(A,$\prec$do(A,wait_until($\downarrow Q\downarrow$))$\succ$, S) $\Leftrightarrow$
know(A, $\prec\exists_I$start(I)=@S@ $\wedge$
occurs(I,do(A,wait_until($\downarrow Q\downarrow$))) $\wedge$
$\forall_{S1\in I}$know_fluent(A, !Q!, $S1$)$\succ$,
S).

KPS.4. kp_satisfied(A,$\prec$do(A,wait_while($\downarrow E\downarrow$))$\succ$, S) $\Leftrightarrow$
know(A, $\prec\exists_I$ start(I)=@S@ $\wedge$
occurs(I,do(A,wait_while($\downarrow E\downarrow$))) $\wedge$
$\forall_{S1\in I}$ know_whether(A,
$\prec\exists_{I1}$ end($I1$)=@$S1$@ $\wedge$
occurs($I1$, $\downarrow\downarrow E\downarrow\downarrow$)$\succ$, S)$\succ$,
S)

(Note: In the third line of KPS.4, the doubly imbedded antiquotes of @$S1$@ are scoped to the internal string delimiters. The double antiquotes of $\downarrow\downarrow E\downarrow\downarrow$ are scoped to the external string delimiter.)

We now define the knowledge preconditions for a complex plan, with sequence and conditional operations. (Knowledge preconditions for plans with loops can be defined analogously.) An agent will be able to carry out a physically feasible plan if the knowledge preconditions for each primitive action are satisfied at the time when he has to perform it and he knows the value of each conditional when he has to compute it. Therefore, the knowledge preconditions for the plan at its start are that the agent must know now that he will know the knowledge preconditions for each primitive action by the time he must take it, and that he will know the value of each conditional. For example, the knowledge preconditions for performing the sequence "sequence($E1, E2$)" are satisfied in S if the knowledge preconditions of $E1$ are satisfied in S, and it is known in S that the knowledge preconditions of $E2$ will be satisfied after $E1$ has been executed.

KPS.5. kp_satisfied(A,$\prec$sequence($\downarrow E1\downarrow, \downarrow E2\downarrow$)$\succ$,$S$) $\Leftrightarrow$
[kp_satisfied($A, E1, S$) $\wedge$
know(A,$\prec\forall_{S2}$ occurs([@S@, $S2$], $\downarrow E1\downarrow$) $\Rightarrow$
kp_satisfied(@A@, $\downarrow E2\downarrow, S2$)$\succ$,$S$)].

The knowledge preconditions for A performing the conditional action "cond($P, E1, E2$)" are satisfied in situation S if the following conditions are satisfied: (i) A knows whether P is true or false in situation S; (ii) If P is true in S then the knowledge preconditions for $E1$ are satisfied; if P is false, then the knowledge preconditions for $E2$ are satisfied.

KPS.6. kp_satisfied(A,$\prec$cond($\downarrow P\downarrow, \downarrow E1\downarrow, \downarrow E2\downarrow$)$\succ$,$S$) $\Leftrightarrow$
[know_fluent(A, P, S) $\wedge$ [true_in(S,denotation(P)) $\Rightarrow$
kp_satisfied($A, E1, S$)] $\wedge$ [$\neg$true_in(S,denotation(P) $\Rightarrow$
kp_satisfied($A, E2, S$)]].

To employ these definitions in a nontrivial way, it is necessary that the agent have a theory allowing him to predict what he will know at future times. For example, suppose that, in situation s0, Leo is in room 1, he knows that block B is either in room 2 or in room 3, and he has the goal that block B should be in room 4. Leo has two primitive actions: The action "move(R)" moves himself and anything he is holding to room R. It has no preconditions. The action "pickup(X)" has the effect that Leo is holding X, and the precondition that Leo and X be in the same room. Leo knows all relevant physical and epistemic axioms. He also knows that, for any object, he knows whether the object is in the same room as himself. This last fact can be stated as follows:

know(leo, $\prec\forall_{S1,X}$ know_fluent(leo,
$\prec$eql(place(leo), place(@X@))$\succ$,$S1$)$\succ$, s0).

(Note: in the above formula, the doubly imbedded antiquoted expression @X@ is scoped to the internal string delimiters.)

Leo then knows that he can achieve his goal by executing the following plan:

```
sequence(move(room2),
         cond(eql(place(leo),place(blockb)),
              sequence(pickup(blockb),move(room4)),
              sequence(move(room3),pickup(blockb),
              move(room4)).
```

The physical axioms are sufficient to show that the plan is physically feasible and that if it is carried out, then the goal will be achieved. The conditions that Leo knows the physical axioms and knows that he will know whether the block is in the same room are needed for the knowledge preconditions, to guarantee that Leo will know which branch to take when he comes to the conditional.

An agent A is able to perform a plan P if he knows that P is feasible and that P's knowledge preconditions are satisfied. He is able to achieve goal G if there is a plan P that he is able to perform and that he knows will lead to G. Formally, we define the predicates "can_do(A, P, S)," meaning that A can perform plan P in situation S, and "can_achieve(A, G, S)," meaning that A can achieve goal G in situation S, using the following axioms:

KPS.7. can_do(A, P, S) $\Leftrightarrow$
know(A,$\prec$true_in(@S@,feasible(@A@, $\downarrow P\downarrow$)) $\wedge$
kp_satisfied(@A@, !P!, @S@)$\succ$,S).

KPS.8. can_achieve(A, G, S) $\Leftrightarrow$
can_do(A, P, S) $\wedge$
know(A,$\prec$true_in(@S@,lead_to($\downarrow P\downarrow, \downarrow G\downarrow$))$\succ$, S).

Note that the condition that A knows of the feasibility and success of the plan is a separate requirement than that the knowledge preconditions are satisfied. Consider, for example, the following example:[4] Nicholas has received two packages. He knows that one is a bomb and the other is innocuous, but he does not know which is which. He also knows that the bomb can be deactivated by putting it in the toilet, but unfortunately he has only one toilet available, and it will not hold both packages. We would like to be able to conclude that Nicholas is not able to deactivate the bomb, and indeed the above definition of

[4] This example is a modification of a problem proposed by Bob Moore, cited in [McDermott 1987a].

"can_achieve" will support that conclusion. There does exist a valid plan whose knowledge preconditions are satisfied; namely, either the plan "Put package A in the toilet" or the plan "Put package B in the toilet." However, Nicholas does not know which of these plans is valid. Conversely, there is a plan that Nicholas knows will work — the plan "Put the package with the bomb in the toilet" — however, the knowledge preconditions of this plan are not satisfied.

(It is possible to design a planner that solves this problem without explicitly reasoning about knowledge and knowledge preconditions; see [Pednault 1988]. However, such a planner implicitly uses meta-level (syntactic) categorizations of the plan involved. Certainly, in any theory that treats plans as event types rather than as descriptions of event types, it must be possible to prove the result, "There is a plan with a single action that defuses the bomb.")

9.3.3 Planning and Acting

All the planning we have discussed so far has been suitable for a setting where a top-level goal is presented in a starting situation, and the planner can find a plan, complete or partial, described entirely in terms of constraints on sequences of primitive actions, that is provably valid for the goal. Executing the plan then requires only finding a particular sequence of primitive actions that satisfies the constraints. In reality, it is only in rare, tightly controlled environments that it is possible or practical to plan with this degree of detail and certainty. Rather, a plan is partially developed at the start. Its full expansion into primitive actions is interspersed with its execution. Thus, the planner plans only primitive actions that will be executed soon; later parts of the plan are sketched only dimly. For example, an agent who is planning to go to a destination will not plan out every individual step, but only the basic outlines of his route; the individual steps will be planned only as they have to be taken. This is partly because individual steps must be chosen in response to circumstances that are unknown but very unlikely to affect the larger plan, and partly because even if all future steps could be planned with certainty, to do so is just a waste of computational resources. Interleaving planning with execution also makes it possible for the agent to respond more flexibly to errors — unanticipated obstacles or opportunities — discovered in his world model during execution.

This kind of planning has proven to be hard to implement and even harder to formalize. Here, we will only describe some of the central issues involved.

- *Plan representation*: In the plans we have discussed so far, it is necessary to describe only the primitive actions to be carried out, with constraints on when and whether each should be executed. By contrast, in representing a plan that will be expanded or modified at a later time, it is necessary to record a variety of information to guide the later stages of planning. In particular, it is necessary to represent the purpose of each part of the plan. Typically, these functions are represented in terms of a subgoal or subtask hierarchy. The purpose of a lower-level task is either as a component of a supertask or to achieve a precondition of some later task. (See Figure 9.4.) Other kinds of information to be recorded include constraints of various kinds among parts of the plan, and multiple alternative plans, to be chosen among at a later stage of expansion.

  It is also important, of course, to record which parts of the plan tree have already been accomplished and which are still pending. It may be desirable, for the sake of memory efficiency, to forget (i.e., drop from the plan tree) any portion of it that has already been accomplished.

- *Modification operators*: Additional modification operators are needed to maintain the plan during execution. In particular, a piece of the plan tree may disappear without ever being carried out, either because its purpose has been accomplished without it being necessary to carry it out, or because it has been determined to be impossible or impractical, or because all its supertasks have either been accomplished or disappeared.

- *Actions of general purpose*: An agent may wish to carry out an action whose purpose is not well specified but merely serves to put him in a stronger position. For example, an agent may wish to have more money available than he needs for planned activities, just in case additional expenses arise. An agent may answer the telephone without knowing which, if any, of his goals will be advanced by so doing.

- *Evaluation*: Open-ended planning typically takes place in an uncontrolled and partially known environment in which success cannot be predicted with absolute certainty; it can only be made more or less certain. The evaluation must thus be probabilistic.

- *Plan maintenance*: Finally, there is the central question of search and control, determining what parts of the planning tree to expand or modify and how far to expand them.

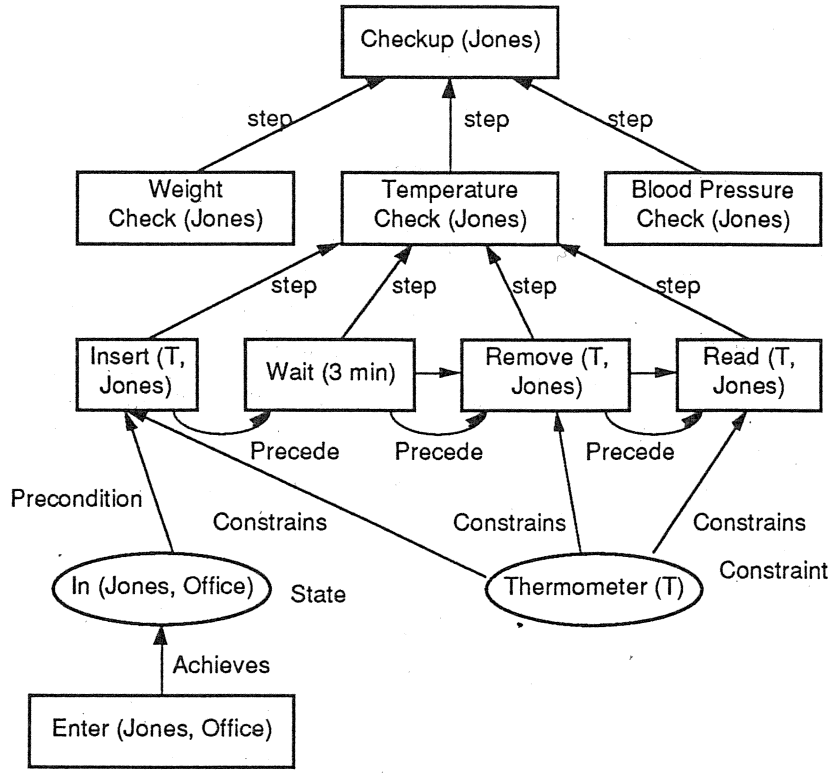


Figure 9.4 Hierarchy of tasks

9.3.4 Reactive Planning

A recent trend in planning research has been to study planning systems that perform relatively little inference or prediction of future states, but instead rely on a library of rules that specify an appropriate action given a goal and a current world state. There is wide variation among such systems in the richness of the world model maintained and of the inferences performed, if any. (Some systems, such as Agre and Chapman's PENGI [1987] have no internal world model whatever; they rely on immediate perceptions.) We will briefly discuss Firby's [1989] RAP system, which is notable among reactive planners for using a relatively high-level planning language and world model.

In Firby's system, a plan consists of a number of RAPs (Reaction Action Packages). A RAP consists of the following parts:

1. A *task* to be achieved by the RAP.
2. A *success* criterion, which may be verified when the RAP has succeeded in carrying out its task.
3. A number of *methods* for carrying out the task. Each method consists of
   (a) A *context*, a world state that must hold for the method to be appropriate.
   (b) A *task network* consisting of a set of subtasks or primitive actions with constraints on their performance.

Table 9.4 shows one simple RAP. (The LISP-like notation is Firby's. Symbols beginning with a question mark are variables. The example is slightly simplified from Firby's; the original had an additional clause whose significance depended on interpreting negation as failure. The logical translation in Table 9.5 is our interpretation, not Firby's.)

The RAP system maintains an agenda of tasks to be performed and a world model. The world model is updated by perceptions, which may be obtained as the direct results of a particular perception task, or may be side effects of the robot's activities. The task agenda is initialized to contain the robot's top-level goals. The robot then repeatedly chooses a task from the agenda to perform, based on scheduling criteria. If the task is primitive, it is executed by a direct command to the effectors. Otherwise, the robot chooses one of its methods whose context is currently satisfied and adds the task network for that method to the agenda. If all parts of a task network have been accomplished, the success criterion of the supertask is checked. The supertask has been successful if the criterion holds; otherwise it has failed. When a task fails, the method of which it was part is held to have failed, so that all other tasks associated with that method are removed from the task agenda.

We can characterize the above RAP in terms of a set of axioms like those of Table 9.5.

The formal interpretation of the central components of a RAP is thus quite straightforward. Such a characterization would be useful in trying to relate a RAP to a physical model of the domain. (The complete RAP language gets a great deal more complicated than this, with many programming bells and whistles. Formalizing these would be a lengthy project in robotic programming language semantics.) Characterizing or justifying the control structure in formal terms, by contrast,

Table 9.4 Sample RAP

```
(DEFINE-RAP
    (INDEX (load-into-truck ?object))
    (SUCCESS (location ?object in-truck))
    (METHOD
        (CONTEXT (and (size-of ?object ?size)
                      (<= ?size arm-capacity)))
        (TASK-NET
            (t1 (pickup ?object)
                ((holding arm ?object) for t2))
            (t2 (putdown ?object in-truck))))
    (METHOD
        (CONTEXT (and (size-of ?object ?size)
                      (> ?size arm-capacity)))
        (TASK-NET
                (t1 (pickup lifting-aid)
                    ((holding arm lifting-aid) for t2))
                (t2 (pickup ?object)
                    ((holding arm ?object) for t3))
                (t3 (putdown ?object in-truck)))))
```

The condition ((holding arm ?object) for t2) attached to t1 in the first method signifies that t1 should accomplish the state "(holding arm object)" and that this condition should be maintained until the beginning of t2. The conditions in the second method are analogous.

would be difficult. One can imagine a formal analysis of a RAP system analogous to the analysis of TWEAK in Section 9.2, which would show that a robot with a given set of RAPs could achieve the associated tasks, given certain constraints on the world, but no such analysis has been found, or, so far as I know, sought. Rather, the justification sought for systems like RAP would be empirical — that they do well enough under circumstances that arise in practice.

9.3.5 Characteristic Goals

In understanding the behavior of an agent, it is important to know what its top-level goals are likely to be. Motivation analysis depends

Table 9.5 Axiomatic Partial Characterization of a RAP

RAP.1. occurs(I, load_into_truck(X)) ⇒
in-truck(value_in(end(I),place(X))).
(Success condition)

RAP.2. occurs(I,load_into_truck(X)) ⇔
[occurs(I,lit1(X)) ∨ occurs(I,lit2(X))].
(The task is accomplished by one of two methods)

RAP.3. occurs(I,lit1(X)) ⇒ size_of(X) ≤ arm_capacity.
(Context for first method)

RAP.4. occurs(I,lit1(X)) ⇒
$\exists_{I1,I2}$ occurs($I1$,pickup(X)) ∧
occurs($I2$,putdown(X,in_truck) ∧
end($I1$) ≤ start($I2$) ∧
$\forall_S$ [end($I1$) ≤ S ≤ start($I2$) ⇒
true_in(S,holding(arm,X))].
(Task net for first method)

RAP.5. occurs(I,lit2(X)) ⇒ size_of(X) > arm_capacity.
(Context for second method)

RAP.6. occurs(I,lit2(X)) ⇒
$\exists_{I1,I2,I3}$ occurs($I1$,pickup(lifting_aid)) ∧
occurs($I2$,pickup(X)) ∧
occurs($I3$,putdown(X,in_truck) ∧
end($I1$) ≤ start($I2$) ∧ end($I2$) ≤ start($I3$) ∧
$\forall_S$ [end($I1$) ≤ S ≤ start($I2$) ⇒
true_in(S,holding(arm,lifting_aid))] ∧
$\forall_S$ [end($I2$) ≤ S ≤ start($I3$) ⇒
true_in(S,holding(arm,X))].
(Task net for second method)

entirely on such knowledge; without it, any action could be explained as a top-level goal in its own right. Knowledge of top-level goals is also important in interacting intelligently with other creatures and in predicting one's own future goals, so as to plan for them in advance.

A partial taxonomy of human high-level goals has been proposed by Schank and Abelson [1977]. They propose five general categories of top-level goals in humans:

1. *Satisfaction goals*: There are only a small number of these: hunger, thirst, sleepiness, sexual desire, and so on. A satisfaction goal generally persists until satisfied by an appropriate action (e.g., eating, drinking, sleep, sex), and then reappear after a characteristic time interval. Except for sex, the satisfaction of these goals cannot be indefinitely postponed, and they hold for all humans under all circumstances.

2. *Preservation goals*: The goals of preserving life, health, and property. These are almost always held. Schank and Abelson also distinguish a class of "crisis goals," preservation goals that are under direct threat and therefore must be addressed immediately, such as the preservation goals that are active when one is about to be run over or when one's house is on fire. Crisis goals generally take precedence over any other goal.

3. *Entertainment goals*: These are activities undertaken "for fun." Satisfying them may take from minutes to months to satisfy. They tend to be of lower priority than satisfaction or preservation goals. Examples: read a book, see a movie, visit the Himalayas.

4. *Achievement goals*: Large-scale ambitions whose accomplishment typically requires an extensive structure of actions extending over a long time.[5] Examples: become president, bring up children, write a novel.

5. *Delta goals*: Certain changes of state can be top-level goals in themselves. The most common of these are the gaining of money, property, or knowledge.

Schank and Abelson discuss a number of heuristic rules that characterize these categories of goals along a number of dimensions, such as the relative priority of the goals, the ease with which an agent will substitute one goal for another, and the emotional reaction of an agent to the failure of a goal. For instance, an agent will easily substitute one entertainment goal for another. This is more difficult, though sometimes possible, with achievement goals and delta goals. With satisfaction goals, it is easy to substitute a new argument (eat salmon instead of eat steak), but not to substitute a new type of goal (sleep instead of eat.) Goal substitution among preservation goals takes the form of giving up on less important goals in order to preserve more important goals; for example, spending one's savings to

[5]There are occasionally achievement goals that do not require many actions, such as Prince Charles's (presumptive) goal of being king of England, which requires on his part only that he outlive his mother and that he avoid serious offense to his country.

maintain one's health. (These rules can be used to predict behavior only in extreme cases, where one goal is overwhelmingly more important than others. Ordinarily, people continually choose between goals, and it is essentially impossible to know how they will choose without a detailed knowledge of the individual and the circumstances.)

The above taxonomy is not by any means complete. For example, people often have goals for others, such as the betterment of family members, friends, or society (or the injury of enemies). Given the diversity of human desires, it might seem hopeless to arrive at any taxonomy or any necessary conditions for human goals. On the other hand, there are certainly limits on top-level goals that are commonsensically known. A man with a top-level goal of putting a tulip in his glove compartment every day when the Mets score an odd number of runs would be recognizably peculiar.

Also important to an understanding of human behavior is a theory of interactions among goals and actions that will predict, or at least constrain, what actions an agent will perform given a situation and a collection of goals. Making such a choice often involves a complex evaluation, estimating the expected costs and the expected gain of achieving or postponing the various goals. Nonetheless, there is some commonsense consensus on how this calculation should be made. It would be generally agreed, for example, that a man who played chess while his house burned down around him was exercising poor judgment. A similar, deeper example would be to infer that a subsistence farmer in a hard winter will not eat his seed grain until he is in immediate danger.

A related, difficult, issue is the formulation of internal constraints on the set of goals held by an agent at a time, similar to the constraints on an agent's beliefs discussed in Chapter 8. For example, axioms such as the following might seem plausible:

- If G is a goal of A, and A believes that G implies Q, then Q is a goal of A.
- If G is a goal of A, then $\neg G$ is not a goal of A.
- If A believes that G will be a goal of A, then it is a current goal of A that he should be able to achieve G at the future time when he wants it.
- If A has a goal that G should be a goal of A's, then G is a goal of A.

9.4 References

Planners: Most of the AI analysis of plans and goals has, naturally, been in the context of planning programs. The following planning programs were particularly notable: GPS [Newell and Simon 1963] introduced the concepts of means-end analysis and of recursively calling the planner to achieve preconditions. QA3 [Green 1969] constructed plans by applying general theorem-proving techniques to situation-calculus axioms. STRIPS [Fikes and Nilsson 1971] used a GPS strategy applied to a fixed representation for actions. It also used a general-purpose theorem prover to compute dependencies among states in a single situation from state-coherence axioms. (The PLANNER language [Hewitt 1969; Sussman, Winograd and Charniak 1970] appears in retrospect as more an early version of a logic-programming language than a theory of planning.) HACKER [Sussman 1973] learned plans in the blocks world by a process of incremental debugging. BUILD [Fahlman 1974] was a powerful specialized planner for the blocks world. ABSTRIPS [Sacerdoti 1975] extended STRIPS by introducing task reduction and levels of abstraction. NOAH [Sacerdoti 1975] likewise used task reduction, and introduced nonlinear planning. Nonlinear planners with task reduction were further extended in NONLIN [Tate 1977]; in NASL [McDermott 1978b], which also studied the integration of planning and execution; and in DEVISER [Vere 1983], which incorporated a metric language of time and time intervals. MOLGEN [Stefik 1981], which constructed plans for biological experiments, highlighted the use of constraints on variable bindings in planning. TWEAK [Chapman 1987], as described in the text, provides an exceptionally clean theory of nonlinear planning, but does not incorporate task reduction. SIPE [Wilkins 1988] is a state-of-the-art planner that combines task reduction, nonlinear planning, and constraint posting. FORBIN [Miller, Firby, and Dean 1985] incorporates advanced techniques for temporal reasoning in the planner. In particular, FORBIN works in domains where it is possible that multiple events may occur simultaneously. [McDermott 1990] describes a complete linear planner for plans that may include actions with situation-dependent effects.

Theoretical analysis: [Georgeff and Lansky 1987] is a collection of a number of papers dealing with the logic and representation of plans. Particularly relevant to the issues discussed here are [Lifschitz 1987], which gives a formal account of the STRIPS program; [Manna and Waldinger 1987], which studies plan construction using methods from automatic programming, and, in particular, addresses the problem of constructing plans with conditionals; [Drummond 1987], which develops a plan representation capable of expressing conditionals and

loops; and [Cohen and Levesque 1987] which describes how an agent will maintain and pursue his goals. [McDermott 1985] is a (very difficult) study of the logical structure of a task-reduction planner that interleaves planning and acting. [Pednault 1988] provides an analysis for TWEAK-like plans that include actions, like toggling a switch, whose effect depends on the starting state. [Dean and Boddy 1988] demonstrate that predicting the effect of such a plan is NP-hard, and gives a polynomial-time partial solution.

Knowledge preconditions: Knowledge preconditions for plans are studied in [McCarthy and Hayes 1969], [Moore 1980], and [Morgenstern 1987].

Reactive planning: The RAP system discussed in the text is taken from [Firby 1989]. Other important studies include [Agre and Chapman 1987; Brooks 1986; Kaelbling 1987; Hendler 1989; Georgeff 1988; and Schoppers 1987].

Motivation analysis: The representation of plans and goals for motivation analysis has been studied in [Charniak 1975; Rieger 1975; Schank and Abelson 1977; Wilensky 1978; Wilensky 1980; Dyer 1985; and Kautz and Allen 1986].

9.5 Exercises

(Starred problems are more difficult.)

1. (a) Translate the STRIPS representation of blocks-world events in Table 5.11 into the TWEAK representation.
   (b) * Show how TWEAK could construct a plan to get from state A to state B in Figure 5-2 using the actions defined in part a.

2. Show that if A can achieve G in S, then A knows that he can achieve G in S.

3. * Formalize the following inference:

   Given that Nicholas knows the following:

   (a) Either package A or package B contains a bomb.
   (b) The package containing the bomb weighs 5 pounds while the other package weighs 3 pounds.
   (c) If he lifts the package, then he will know its weight.
   (d) If the bomb is put in the toilet, then it will be defused.
   (e) It is possible to put a package in the toilet iff the toilet is empty.
   (f) The toilet is currently empty.

Infer: Nicholas can achieve the goal of defusing the bomb.

4. * Formalize the following inference:

   Joe is at his workshop. He has to build a desk, but his only record of the dimensions is at home. His customers, who are the only people who know the dimensions, are out of town. Infer that Joe will have to go home to get the dimensions.

5. * Consider the following plan:

   Step 1. Take a taxi to the train station no later than 3:30.
   Step 2. Buy a ticket to Miami within 5 minutes of arriving at the station.
   Step 3. When the train to Miami is announced, go immediately to the track where it leaves, and get on the train.

   (a) Construct a representation for this kind of plan, and axiomatize the events that constitute carrying out such a plan.
   (b) Define the conditions for the physical feasibility of a plan described in the language constructed in part a. The definition should support inferences like "If the train leaves at 3:50 and the taxi takes longer than 30 minutes to get to the station, then the above plan is infeasible."
   (c) Define the knowledge preconditions of a plan described in the language of part a.