CHAPTER 18

Uncertainty with Neoclassical Techniques

18.1 Introduction

In Chapters 16 and 17, we discussed two approaches devised to solve the problem of planning under uncertainty: planning based on Markov Decision Processes and planning based on model checking. In addition, some approaches originally devised for classical planning problems have been extended to deal with some form of uncertainty. Some of them are based on plan-space planning, state-space planning, planning-graph techniques, and planning as satisfiability.

In this chapter, we focus on extensions to two neoclassical techniques, planning as satisfiability (Section 18.2) and planning-graph techniques (Section 18.3), while other approaches are mentioned in Section 18.4. We limit our discussion to the case of conformant planning, i.e., the case of planning in nondeterministic domains, for reachability goals, and with null observability, where the generated plans are sequences of actions. See Section 18.4 for a general discussion on the relevance and efficiency of the two neoclassical approaches.

18.2 Planning as Satisfiability

In this section, we first define the notion of weak and strong conformant solutions (Section 18.2.1). We show how planning problems can be encoded as propositional formulas: nondeterministic effects result in disjunctions of propositions (Section 18.2.2). Finally, we describe a planning algorithm based on satisfiability techniques (Section 18.2.3) and briefly discuss Quantified Boolean Formulas (Section 18.2.4).

18.2.1 Planning Problems

A nondeterministic action can be described with the following operator:

Action
  preconditions: P
  deterministic effects: E
  nondeterministic effects: $N_1, \ldots, N_n$

where P is the set of literals that represent the preconditions, E is the set of literals that represent *deterministic effects* (i.e., effects that hold deterministically after the action is executed), and $N_1, \ldots, N_n$ are n sets of literals that represent *nondeterministic effects*. The intended meaning is that some of the N_i will actually take place, but we do not know which ones.

Example 18.1 Here is a nondeterministic action $\mathsf{move}(r, l, l')$ that either succeeds and moves the robot from location l to destination l' or fails and leaves the robot in its original location l.

$\mathsf{move}(r, l, l')$
  preconditions: $\mathsf{at}(r, l)$
  deterministic effects: $\{\}$
  nondeterministic effects: $\{\mathsf{at}(r, l'), \neg\, \mathsf{at}(r, l)\}, \{\}$

■

In the rest of this section, we write $\mathsf{precond}(a)$, $\mathsf{deterministic\text{-}effects}(a)$, and $\mathsf{nondeterministic\text{-}effects}(a)$ to denote the sets $\{P\}$, $\{E\}$, and $\{N_1, \ldots, N_n\}$, respectively. We further assume that for any action a, any nondeterministic effect N_i is consistent with the deterministic effects E, and $N_i \cup E$ cannot contain both a literal and its negation.

A weak conformant solution is a sequence of actions that may achieve the goal but is not guaranteed to do so. A strong conformant solution is a sequence of action that is guaranteed to reach the goal.

Definition 18.1 Let $P = (\Sigma, S_0, S_g)$ be a planning problem, where $\Sigma = (S, A, \gamma)$ is a nondeterministic state-transition system, $S_0 \subseteq S$ is the set of initial states, and $S_g \subseteq S$ is the set of goal states. As in Chapter 7, let (P, n), where $n \in \mathcal{N}^+$, be a bounded planning problem of length n, i.e., the problem of finding a plan of length n that satisfies the original planning problem. Let π be a sequence of actions $a_1, \ldots a_n$.

1. π is a *weak conformant solution* to the bounded planning problem (P, n) iff there exists a sequence of states $\langle s_0, \ldots, s_n \rangle$ such that $s_0 \in S_0$, $s_{i+1} \in \gamma(s_i, a)$ for each $i = 0, \ldots, n-1$, and $s_n \in S_g$.
2. π is a *strong conformant solution* to the bounded planning problem (P, n) iff $s_n \in S_g$ for every sequence of states $\langle s_0, \ldots, s_n \rangle$ such that $s_0 \in S_0$ and $s_{i+1} \in \gamma(s_i, a)$.

■

Notice that the notion of a weak conformant solution given in this section is even weaker than the notion of a weak solution given for planning based on model checking (Chapter 17). Here we just require that there be *at least one state* in the set of initial states such that there exists a path to the goal. In planning based on model checking, we require that there be at *least one path for each initial state.*

18.2.2 Planning Problems as Propositional Formulas

A planning domain Σ of a planning problem $P = (\Sigma, S_0, S_g)$ can be encoded as a propositional formula $\Phi(\Sigma)$. $\Phi(\Sigma)$ is a conjunct of smaller formulas for each action and for each fluent. As in Chapter 7, a_i stands for an action at step i of the bounded planning problem, $0 \leq i \leq n-1$. For each a_i there is a formula to say that if action a takes place at step i, then its preconditions must hold at step i and its effects will hold at the next step $i+1$. This formula is the conjunct of two smaller formulas, one for the deterministic effects and one for the nondeterministic ones:

$$a_i \Rightarrow \left(\bigwedge_{p \in \text{precond}(a)} p_i \wedge \bigwedge_{e \in \text{deterministic-effects}(a)} e_{i+1} \right) \tag{18.1}$$

$$a_i \Rightarrow \bigvee_{N \in \text{nondeterministic-effects}(a)} \bigwedge_{e \in N} e_{i+1} \tag{18.2}$$

Formula 18.1 is the same as Formula 7.11. Formula 18.2 states that an action a executed at step i implies that *some* of its nondeterministic effects are true at step $i+1$.

Explanatory frame axioms have to take into account both deterministic and nondeterministic effects. In the case of deterministic effects, it is enough to require that one of the actions that causes the effect has been executed at step i, as in Chapter 7. In the case of a nondeterministic effect $e \in N$, where $N \in \text{nondeterministic-effects}(a)$, we must require that all the effects in N hold at step $i+1$. Formally, for each fluent f and for each $0 \leq i \leq n-1$:

$$\neg f_i \wedge f_{i+1} \Rightarrow (\bigvee_{a \in A | f_i \in \text{deterministic-effects}^+(a)} a_i) \vee \bigvee_{a \in A} \bigvee_{N | N \in \text{nondeterministic-effects}^+(a), e \in N} (a_i \wedge \bigwedge_{e' | e' \in N} e'_{i+1}) \tag{18.3}$$

We have a similar formula for negative effects.

The complete exclusion axioms are the same as in the case of deterministic domains, i.e., for each $0 \leq i \leq n-1$, and for each distinct $a_i, b_i \in A$:

$$\neg a_i \vee \neg b_i \tag{18.4}$$

Therefore we have propositional formulas that encode action effects (see Formula 18.2), explanatory frame axioms (Formula 18.3), and complete exclusion axioms (Formula 18.4). Let $\Phi_i(\Sigma)$ be the conjunction of Formulas 18.2, 18.3,

and 18.4. $\Phi_i(\Sigma)$ encodes the planning domain at step i. Let $\Phi(\Sigma)$ be the conjunction of $\Phi_i(\Sigma)$ for any $0 \leq i \leq n-1$:

$$\bigwedge_{i=0}^{n-1} \Phi_i(\Sigma)$$

$\Phi(\Sigma)$ encodes the planning domain. Given the planning problem $P = (\Sigma, S_0, S_g)$, the encoding of the initial states S_0 and of the goal states S_g is the same as for deterministic domains (see Chapter 7). Let $\Phi(S_0)$ and $\Phi(S_g)$ be the encodings of S_0 and S_g, respectively. Finally, Let $\Phi(P)$, or simply Φ, be the encoding of the bounded planning problem (P, n):

$$\Phi(S_0) \wedge \Phi(\Sigma) \wedge \Phi(S_g)$$

Example 18.2 Consider the planning problem P of Example 7.1 in Chapter 7. Suppose we modify this example so that the move operator is the one in Example 18.1. Suppose the robot is at l1 in the initial state and at l2 in the goal state. Consider the bounded planning problem $(P, 1)$. The action is encoded as the following formulas:

$$\mathsf{move(r1, l1, l2, 0)} \Rightarrow \mathsf{at(r1, l1, 0)} \tag{18.5}$$

$$\mathsf{move(r1, l1, l2, 0)} \Rightarrow (\mathsf{at(r1, l2, 1)} \wedge \neg\mathsf{at(r1, l1, 1)}) \vee \mathit{True} \tag{18.6}$$

$$\mathsf{move(r1, l2, l1, 0)} \Rightarrow \mathsf{at(r1, l2, 0)} \tag{18.7}$$

$$\mathsf{move(r1, l2, l1, 0)} \Rightarrow (\mathsf{at(r1, l1, 1)} \wedge \neg\mathsf{at(r1, l2, 1)}) \vee \mathit{True} \tag{18.8}$$

Notice that Formulas 18.6 and 18.8 are tautologies. Indeed, we have two states in the planning domain, and each action may lead to any state in the domain.

The explanatory frame axioms are:

$$\neg\mathsf{at(r1, l1, 0)} \wedge \mathsf{at(r1, l1, 1)} \Rightarrow \mathsf{move(r1, l2, l1, 0)} \wedge \neg\mathsf{at(r1, l2, 1)} \tag{18.9}$$

$$\neg\mathsf{at(r1, l2, 0)} \wedge \mathsf{at(r1, l2, 1)} \Rightarrow \mathsf{move(r1, l1, l2, 0)} \wedge \neg\mathsf{at(r1, l1, 1)} \tag{18.10}$$

■

Given a bounded planning problem (P, n), we can automatically construct the formula Φ encoding (P, n). The formula Φ can be used to find weak conformant solutions. Recall that, if Φ is satisfiable, then it is possible to extract a plan from one of its models μ (see Chapter 7).

Proposition 18.1 *Let (P, n) be a bounded planning problem. Let Φ be the encoding of (P, n). Let μ be a model of Φ. Then a plan extracted from μ is a weak conformant solution to the bounded planning problem (P, n).*

Now consider the problem of finding a strong conformant solution. This can be done as follows.

1. Generate weak plans by finding a model μ of Φ.
2. Check whether the weak plan is a strong plan.

The second of these two steps is a plan validation problem, i.e., the problem of determining whether a given plan satisfies some condition. A plan π extracted from a model μ can be represented by a propositional formula $\psi(\mu)$. For instance, given an assignment $\mu = \{\mathsf{move}(\mathsf{r1},\mathsf{l1},\mathsf{l2},0), \mathsf{move}(\mathsf{r1},\mathsf{l2},\mathsf{l1},1),\ldots\}$, $\psi(\mu)$ is $\mathsf{move}(\mathsf{r1},\mathsf{l1},\mathsf{l2},0) \wedge \mathsf{move}(\mathsf{r1},\mathsf{l2},\mathsf{l1},1)$. We have to find a formula Φ' such that for any μ' that is a model of $\psi(\mu) \wedge \Phi'$, the plan extracted from μ' is a strong solution to (P, n). In order to construct Φ', we start from the observation that any model of $\psi(\mu) \wedge \Phi$ is a strong solution if each action in A is executable in any state. Given a bounded planning problem (P, n) whose planning domain is the state-transition system Σ, we therefore construct a planning domain Σ' such that actions that are not executable in Σ are instead executable in Σ', and all of them lead to failure states. For each state s in Σ, Σ' contains both s and a failure state s' in which a special fluent *Fail* holds. $\Sigma' = (S', A', \gamma')$ is constructed from $\Sigma = (S, A, \gamma)$ as follows.

1. $S' = S \cup \{s \cup \mathit{Fail} \mid s \in S\}$.
2. A' is the same as A.
3. Let s'_1 and s'_2 be states of S'. Let $s_1 = s'_1 - \{\mathit{Fail}\}$ and $s_2 = s'_2 - \{\mathit{Fail}\}$ be two states of S. $s'_2 \in \gamma'(s'_1, a)$ iff one of the following is true.
   (a) *Fail* holds neither in s'_1 nor in s'_2, and $s_2 \in \gamma(s_1, a)$.
   (b) *Fail* holds only in s'_2, and a is not executable in s_1 (i.e., $\neg\exists\, s | s \in \gamma(s_1, a)$).
   (c) *Fail* holds both in s'_1 and in s'_2.

Each action is executable in Σ', and given a sequence $s_0, \ldots, s_n$ in Σ, where each $s_{i+1} \in \gamma(s_i, a_i)$, we have a corresponding sequence $s'_0, \ldots, s'_n$, where *Fail* does not hold in each s'_0, and $s'_{i+1} \in \gamma'(s'_i, a_i)$. Let $\mathsf{executable}_i$ be the conjunction of the formulas

$$a_i \rightarrow \bigwedge_{p \in \mathsf{precond}(a)} p_i$$

over all actions $a \in A$. $\mathsf{executable}_i$ represents the executable actions at step i. Then $\Phi_i(\Sigma')$ is:

$$(\Phi(\Sigma) \wedge \neg \mathit{Fail}_i \wedge \neg \mathit{Fail}_{i+1}) \vee (\mathit{Fail}_i \vee \neg \mathsf{executable}_i) \wedge \mathit{Fail}_{i+1})$$

Because each action is executable in Σ', then the plan extracted from the model μ of Φ is a strong conformant solution if:

$$\psi(\mu) \wedge \Phi(S_0) \wedge \bigwedge_{i=0}^{n-1} \Phi_i(\Sigma') \models \Phi(S_g) \wedge \neg \mathit{Fail}_n$$

We therefore define Φ' as:

$$\psi(\mu) \wedge \Phi(S_0) \wedge \bigwedge_{i=0}^{n-1} \Phi_i(\Sigma') \wedge \neg\Phi(S_g) \vee Fail_n \tag{18.11}$$

We call Φ' the *strong encoding* of the bounded planning problem (P, n).

Example 18.3 In Example 18.2, Σ has two states, s_1 and s_2. at(r1,l1) holds in s_1, while at(r1,l2) holds in s_2. Σ' has four states, s'_1, s'_2, s'_3, and s'_4. State s_1 corresponds to s'_1 and s'_3 (*Fail* holds in s'_3), while s_2 corresponds to s'_2 and s'_4 (*Fail* holds in s'_4).

executable$_i$ is the conjunction of:

(pre1) move(r1,l1,l2,i) $\Rightarrow$ at(r1,l1,i)
(pre2) move(r1,l2,l1,i) $\Rightarrow$ at(r1,l2,i)

From executable$_i$ it is easy to construct $\Phi(\Sigma')$. ■

The following proposition says that if Φ' is satisfiable, then the plan extracted from the model μ is a strong conformant solution.

Proposition 18.2 *Let (P, n) be a bounded planning problem, ϕ be the encoding of (P, n), μ be a model of ϕ, and ϕ' be the strong encoding (P, n). If ϕ' is unsatisfiable, then a plan extracted from μ is a strong conformant solution to the bounded planning problem (P, n).*

18.2.3 Planning by Satisfiability

Now that we have translated the problems of finding weak and strong conformant solutions into satisfiability problems, we can use standard satisfiability decision procedures, such as Davis-Putnam (see Chapter 7) to plan for strong conformant solutions. Figure 18.1 shows Strong-Conformant-Davis-Putnam, a planning algorithm based on the Davis-Putnam satisfiability decision procedure. Strong-Conformant-Davis-Putnam iteratively calls Generate-Weak, which generates weak conformant solutions, and then Test-if-Strong, which tests whether the weak solutions are strong. Generate-Weak is essentially a Davis-Putnam procedure that calls Test-if-Strong whenever the procedure finds a model. In Test-if-Strong, the first line takes into account that the assignments generated by Generate-Weak may be partial. It is possible to prove that Strong-Conformant-Davis-Putnam is a sound and complete planning algorithm for a bounded planning problem.

18.2.4 QBF Planning

Planning based on Quantified Boolean Formulas (QBFs) [456] is a generalization of the planning as satisfiability approach. A bounded planning problem is

```
Strong-Conformant-Davis-Putnam()
    return Generate-Weak(Φ,∅)
end

Generate-Weak(φ,μ)
    if φ = ∅ then return Test-if-Strong(μ)
    if ∅ ∈ φ then return False
    if a unit clause L occurs in φ then
        return Generate-Weak(assign(L,φ),μ ∪ {L})
    P ← an atom occurring in φ
    return Generate-Weak(assign(P,φ),μ ∪ {P}) or
        Generate-Weak(assign(¬P,φ),μ ∪ {¬P})
end

Test-if-Strong(μ)
    for each assignment μ' such that μ ⊆ μ' do
        if Φ' is not satisfiable then exit with μ'
    return False
end
```

Figure 18.1 Algorithm for generating and testing strong conformant solutions.

reformulated as a QBF rather than in propositional logic. QBF solvers are employed in place of SAT solvers. QBF logic is a definitional extension to propositional logic, where propositional variables can be universally and existentially quantified (see Appendix C).

This approach can tackle a wide variety of conditional planning problems. The user must provide the structure of the plan to be generated (e.g., the length in the case of conformant planning, the number of control points and observations in the plan in the case of conditional planning). This can provide a significant limitation of the search space. In QBF planning, the points of nondeterminism are explicitly codified in terms of environment variables, with a construction known as *determinization of the transition relation.* In general, in this approach, as in the planning as satisfiability approach, it is impossible to decide whether the given problem admits a solution because it works on bounded planning problems.

18.3 Planning Graphs

We now discuss how to extend planning-graph techniques to deal with conformant planning. In Chapter 6 we explained planning-graph techniques for classical

planning. Given a planning problem with a single initial state, the planner constructs a planning graph where the nodes at level 0 are the propositions denoting the initial state. The preconditions of deterministic actions are connected through action nodes to the effects, keeping track of mutual exclusion relations. Then a plan is extracted by searching the planning graph backward.

Uncertainty in the initial condition means that there is a set of possible states rather than just a single state. Uncertainty in actions' effects means that an action can lead to a set of states rather than a single state. The basic idea is to model uncertainty by using several different planning graphs, one for each possible initial state and one for each possible nondeterministic action outcome.

Given this idea, we can write algorithms for weak conformant planning. It is sufficient to take one graph at a time and work on it in the same way as in the deterministic case. In order to generate strong conformant plans instead, we cannot work on the different graphs independently of the others. It is necessary to consider possible interactions among the effects of the same action in different graphs: an action that has some desired effects in one graph may have some undesired effects in another graph. Therefore, the planning algorithm works as follows.

1. It constructs a separate planning graph for each possible initial state and each possible nondeterministic evolution.
2. It checks mutual exclusion relations not only within one graph but also among different graphs.
3. It extracts a plan with a backward search by checking at each level the effects of an action in all the graphs.

We illustrate the algorithm through a simple example in the DWR domain. For simplicity, in the example we have uncertainty only in the initial state, while actions are deterministic. The generalization of the algorithm to the case of nondeterministic actions is not trivial. However, the simple example should give at least an idea of how a planning-graph algorithm can deal with uncertainty in general.

Example 18.4 Figure 18.2 shows a DWR domain, in which there are a robot r1, a container c1, *n* locations l1, ..., ln, and a location dock. Proposition at(r1,l1) is true if the robot is in l1, unloaded is true if the robot is not loaded, and in(c1,l1) is true if c1 is in l1. Action unload unloads the robot, and move moves the robot to the dock.

unload	move
preconditions :	preconditions :
Effects : unloaded	effects :
	¬at(r1, l1) when at(r1, l1), unloaded;
	¬at(r1, l1), in(c1, dock)when at(r1, l1), ¬unloaded;
	in(c1, dock) when ¬at(r1, l1), ¬unloaded

Initially, the robot is either loaded in l1 or it is unloaded somewhere else. The goal is to move the robot out of l1 without delivering c1 to the dock,

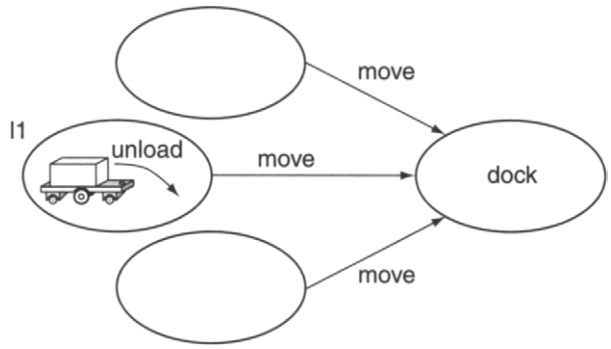

Figure 18.2 A simple DWR domain for conformant planning.

i.e., ¬at(r1,l1) ∧ ¬in(c1,dock). Notice that unload is applicable even if the robot is not loaded. The strong conformant plan is to unload the robot first, and then move it to the dock.

Figure 18.3 shows the two graphs corresponding to the two initial states expanded up to level 4.

■

We now illustrate how the strong conformant algorithm works on this example. The algorithm expands both graphs first, then it searches for a solution separately in each graph, and finally it checks whether a candidate solution for one graph is acceptable for the other ones. The first time, the algorithm expands both planning graphs up to to level 2 because, in both graphs, this level contains the two propositions of the goal ¬at(r1,l1) and ¬in(c1,dock).

The algorithm analyzes the two graphs separately. In G1, the two goal propositions are both initially true. In G2, ¬at(r1,l1) at level 2 is achieved by move, while ¬in(c1,dock) is initially true. There are no mutex relations between actions. A potential solution is therefore action move of the second graph, i.e., move applied to the robot in l1. An alternative potential solution extracted from the first graph is doing nothing.

The algorithm then checks the effects of the potential solution of a graph in all the other graphs. In the example, move in graph G1 has an undesired effect: in(c1,dock). This means that move is not a strong conformant solution. Similarly, doing nothing in G2 leads to at(r1,l1). No backtracking alternative remains; therefore, the algorithm has to further expand the graph. This expansion leads to level 4.

The two graphs are again analyzed separately. In G1, the two goal propositions are both already true at level 2, so no action is needed, while in G2, the algorithm selects the occurrence of move that has at(r1,l1) and unloaded as preconditions because it leads to the goal proposition ¬at(r1,l1).

The algorithm then checks the effects of move in the other graph, G1. This time we have two possibilities: we have one occurrence of move that has the undesired effect

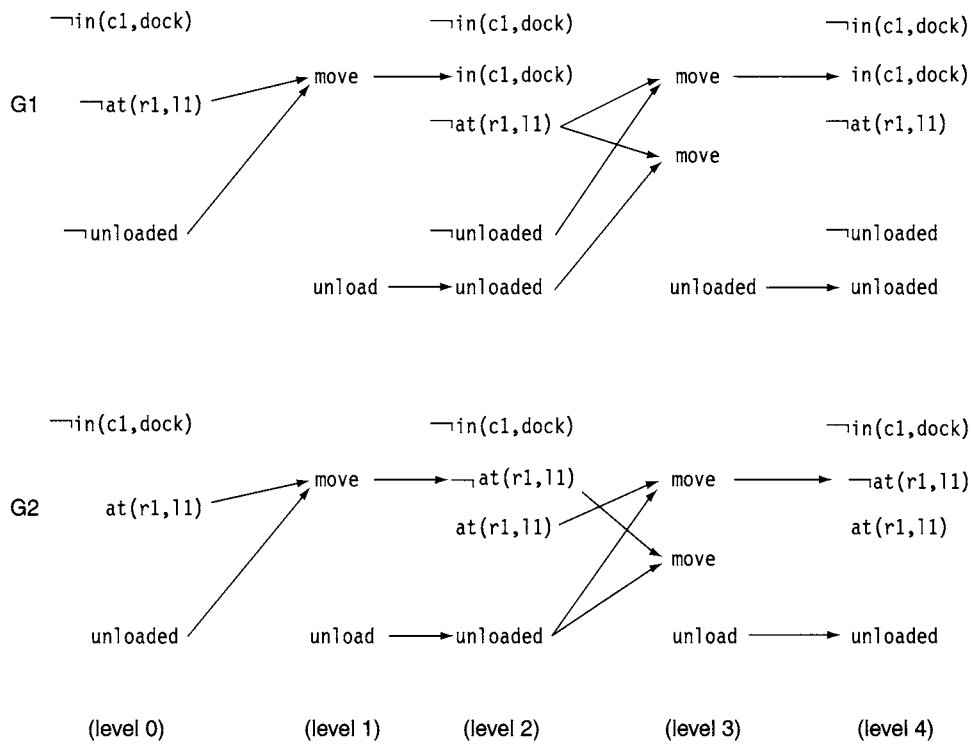

Figure 18.3 Examples of planning graphs for conformant planning.

in(c1,dock) and one occurrence that has no effects. The algorithm can therefore detect that the former should be avoided. The algorithm at this point checks the preconditions of the two occurrences of move: ¬at(r1,l1) is common to both, while ¬unloaded is a precondition of the dangerous occurrence. This precondition should be negated; unloaded is therefore chosen as a subgoal at level 2.

Extraction continues at level 2, where action unload is chosen and analyzed in both graphs. The final conformant plan is the sequence ⟨unload, move⟩.

18.4 Discussion and Historical Remarks

In this chapter, we focused on two neoclassical approaches, satisfiability and planning-graph techniques, as representative cases of approaches originally devised to deal with classical planning that were later extended to deal with uncertainty. So far they can deal with a rather limited portion of the problems in planning under uncertainty: planning as satisfiability can deal only with conformant planning

[109, 110, 188, 231], while planning-graph techniques can deal with conformant planning [490] and some limited form of partial observability [545]. Other than satisfiability and planning-graph techniques, different approaches have addressed the problem of conformant planning. Among them is extension to situation calculus (see Chapter 12), where hand-coded control strategies (in the style of those described in Chapter 10) have been used to significantly prune the search.

Conformant planning is a very particular case of planning under uncertainty. It does not seem to be of relevance to most planning problems, such as those in robotics, where the robot eventually has to do some sensing, otherwise it is lost. Claims that conformant planning can be applied to real-world applications are unsupported except for very peculiar cases, for instance, in the so-called "home sequencing" problem (i.e., the problem of executing a sequence of instructions that makes a microprocessor move to a desired set of states). In this case no observation is possible because memory variables are not readable.

A question is whether satisfiability and planning-graph techniques can scale up to problems of interesting size. While the question remains open for techniques based on satisfiability, planning-graph techniques have been significantly outperformed in several experimental evaluations by techniques based on model checking (Chapter 17), greedy search applied to the MDP framework, and satisfiability itself. On the other hand, we have to recall that planning-graph techniques outperformed earlier classical approaches to uncertainty, such as plan-space planning.

Indeed, the first attempts to deal with uncertainty were the pioneering extensions to plan-space planning [438, 448, 543]. This work has been referred to as *conditional planning*. The basic idea is to extend plan-space algorithms to take into account different alternatives and to generate strong solutions. In general, this approach has been considered a limited form of nondeterminism through the use of an extension of classical STRIPS operators called *conditional actions* (i.e., actions with different, mutually exclusive sets of outcomes). A major problem with this approach is the inability to deal in practice even with simple toy problems.

18.5 Exercises

18.1 Can a policy (as defined in Chapters 16 and 17) be translated into a conformant plan? Vice versa?

18.2 Does a weak or strong conformant solution exist for the planning problem in Example 17.1?

18.3 Describe the SAT encoding of Example 17.1.

18.4 Describe the CGP graph for the planning problem of Example 17.1.

18.5 How would the graphs in Figure 18.3 change in the case of partial observability?