list. That is, \( \text{Paint}(x, \text{can}) \) does not mention the variable \( c \), representing the color of the paint in the can. In the fully observable case, this is not allowed—we would have to name the action \( \text{Paint}(x, \text{can}, c) \). But in the partially observable case, we might or might not know what color is in the can. The variable \( c \) is universally quantified, just like all the other variables in an action schema.

\[
\text{Action (RemoveLid (can),}
\begin{align*}
\text{PRECOND: } & \text{Can(can)} \\
\text{EFFECT: } & \text{Open(can)}
\end{align*}
\]

\[
\text{Action (Paint(x, can),}
\begin{align*}
\text{PRECOND: } & \text{Object(x)} & \text{A} & \text{Can(can)} & \text{A} & \text{Color(can, c)} & \text{A} & \text{Open(can)} \\
\text{EFFECT: } & \text{Color(x, e)}
\end{align*}
\]

To solve a partially observable problem, the agent will have to reason about the percepts it will obtain when it is executing the plan. The percept will be supplied by the agent's sensors when it is actually acting, but when it is planning it will need a model of its sensors. In Chapter 4, this model was given by a function, \( \text{PERCEPT}(s) \). For planning, we augment PDDL with a new type of schema, the percept schema:

\[
\begin{align*}
\text{Percept (Color (x, c),}
\text{PRECOND: } & \text{Object(x)} & \text{A} & \text{InView(x)} \\
\text{Percept (Color (con, c),}
\text{PRECOND: } & \text{Can(can)} & \text{A} & \text{InView(can)} & \text{A} & \text{Open(can)}
\end{align*}
\]

The first schema says that whenever an object is in view, the agent will perceive the color of the object (that is, for the object \( x \), the agent will learn the truth value of \( \text{Color}(x, c) \) for all \( c \)). The second schema says that if an open can is in view, then the agent perceives the color of the paint in the can. Because there are no exogenous events in this world, the color of an object will remain the same, even if it is not being perceived, until the agent performs an action to change the object's color. Of course, the agent will need an action that causes objects (one at a time) to come into view:

\[
\begin{align*}
\text{Action (LookAt (x),}
\text{PRECOND: } & \text{InView(y)} & \text{A} & \text{(x y)} \\
\text{EFFECT: } & \text{InView(x)} & \text{A} & \text{—InView(y)}
\end{align*}
\]

For a fully observable environment, we would have a \( \text{Percept} \) axiom with no preconditions for each fluent. A sensorless agent, on the other hand, has no \( \text{Percept} \) axioms at all. Note that even a sensorless agent can solve the painting problem. One solution is to open any can of paint and apply it to both chair and table, thus coercing them to be the same color (even though the agent doesn't know what the color is).

A contingent planning agent with sensors can generate a better plan. First, look at the table and chair to obtain their colors; if they are already the same then the plan is done. If not look at the paint cans; if the paint in a can is the same color as one piece of furniture, then apply that paint to the other piece. Otherwise, paint both pieces with any color.

Finally, an online planning agent might generate a contingent plan with fewer branches at first—perhaps ignoring the possibility that no cans match any of the furniture—and deal
with problems when they arise by replanning. It could also deal with incorrectness of its action schemas. Whereas a contingent planner simply assumes that the effects of an action always succeed—that painting the chair does the job—a replanning agent would check the result and make an additional plan to fix any unexpected failure, such as an unpainted area or the original color showing through.

In the real world, agents use a combination of approaches. Car manufacturers sell spare tires and air bags, which are physical embodiments of contingent plan branches designed to handle punctures or crashes. On the other hand, most car drivers never consider these possibilities; when a problem arises they respond as replanning agents. In general, agents plan only for contingencies that have important consequences and a nonnegligible chance of happening. Thus, a car driver contemplating a trip across the Sahara desert should make explicit contingency plans for breakdowns, whereas a trip to the supermarket requires less advance planning. We next look at each of the three approaches in more detail.

### 11.3.1 Sensorless planning

Section 4.4.1 (page 138) introduced the basic idea of searching in belief-state space to find a solution for sensorless problems. Conversion of a sensorless planning problem to a belief-state planning problem works much the same way as it did in Section 4.4.1; the main differences are that the underlying physical transition model is represented by a collection of action schemas and the belief state can be represented by a logical formula instead of an explicitly enumerated set of states. For simplicity, we assume that the underlying planning problem is deterministic.

The initial belief state for the sensorless painting problem can ignore InView fluents because the agent has no sensors. Furthermore, we take as given the unchanging facts Object(Table) A Object(Chair) A Can(Cn) A Can(C2) because these hold in every belief state. The agent doesn't know the colors of the cans or the objects, or whether the cans are open or closed, but it does know that objects and cans have colors: ∀x, c: Color(x, c). After Skolemizing, we obtain the initial belief state:

\[ b_0 = Color(x, C(s)) \]

In classical planning, where the closed-world assumption is made, we would assume that any fluent not mentioned in a state is false, but in sensorless (and partially observable) planning we have to switch to an open-world assumption in which states contain both positive and negative fluents, and if a fluent does not appear, its value is unknown. Thus, the belief state corresponds exactly to the set of possible worlds that satisfy the formula. Given this initial belief state, the following action sequence is a solution:

\[ \text{[RemoveLid(Can_1), Paint(Chair, Can_1), Paint(Table, Can_1)]} \]

We now show how to progress the belief state through the action sequence to show that the final belief state satisfies the goal.

First, note that in a given belief state, the agent can consider any action whose preconditions are satisfied by \( b \). (The other actions cannot be used because the transition model doesn't define the effects of actions whose preconditions might be unsatisfied.) According