The action schemas are straightforward, with one exception: preconditions and effects now may contain variables that are not part of the action's variable list. That is, $Paint(x, can)$ 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 $Paint(x, can, c)$. But in the partially observable case, we might or might not know what color is in the can.

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, PERCEPT(s). For planning, we augment PDDL with a new type of schema, the **percept schema**:

Percept schema

$Percept(Color(x, c),$
    PRECOND:$Object(x) \wedge InView(x))$
$Percept(Color(can, c),$
    PRECOND:$Can(can) \wedge InView(can) \wedge Open(can))$

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 $Color(x, c)$ 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:

$Action(LookAt(x),$
    PRECOND:$InView(y) \wedge (x \neq y)$
    EFFECT:$InView(x) \wedge \neg InView(y))$

For a fully observable environment, we would have a *Percept* schema with no preconditions for each fluent. A sensorless agent, on the other hand, has no *Percept* schemas 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