Now let's return to the example problem of achieving a chair and table of matching color. Suppose the agent comes up with this plan:

\[
\text{[LookAt(Table), LookAt(Chair),}
\begin{array}{l}
\text{if Color(Table, a) A Color(Chair, c) then NoOp} \\
\text{else [RemoveLid(Can1), LookAt(Can1)} \\
\text{if Color(Table, c) t Color(Can1, c) then Paint(Chair, Cant)} \\
\text{else REPLAN] .}
\end{array}
\]

Now the agent is ready to execute the plan. Suppose the agent observes that the table and can of paint are white and the chair is black. It then executes \text{Paint(Chair, Cant)}. At this point a classical planner would declare victory; the plan has been executed. But an online execution monitoring agent needs to check the preconditions of the remaining empty plan— that the table and chair are the same color. Suppose the agent perceives that they do not have the same color—in fact, the chair is now a mottled gray because the black paint is showing through. The agent then needs to figure out a position in whole plan to aim for and a repair action sequence to get there. The agent notices that the current state is identical to the precondition before the \text{Paint(Chair, Cant)} action, so the agent chooses the empty sequence for \text{repair} and makes its plan be the same \text{[Paint]} sequence that it just attempted. With this new plan in place, execution monitoring resumes, and the \text{Paint} action is retried. This behavior will Mop until the chair is perceived to be completely painted. But notice that the loop is created by a process of plan–execute–replan, rather than by an explicit loop in the plan. Note also that the original plan need not cover every contingency. If the agent reaches the step marked \text{REPLAN}, it can then generate a new plan (perhaps involving \text{Can2}).

Action monitoring is a simple method of execution monitoring, but it can sometimes lead to less than intelligent behavior. For example, suppose there is no black or white paint, and the agent constructs a plan to solve the painting problem by painting both the chair and table red. Suppose that there is only enough red paint for the chair. With action monitoring, the agent would go ahead and paint the chair red, then notice that it is out of paint and cannot paint the table, at which point it would replan a repair—perhaps painting both chair and table green. A plan-monitoring agent can detect failure whenever the current state is such that the remaining plan no longer works. Thus, it would not waste time painting the chair red. Plan monitoring achieves this by checking the preconditions for success of the entire remaining plan—that is, the preconditions of each step in the plan, except those preconditions that are achieved by another step in the remaining plan. Plan monitoring cuts off execution of a doomed plan as soon as possible, rather than continuing until the failure actually occurs. Plan monitoring also allows for serendipity—accidental success. If someone comes along and paints the table red at the same time that the agent is painting the chair red, then the final plan preconditions are satisfied (the goal has been achieved), and the agent can go home early.

It is straightforward to modify a planning algorithm so that each action in the plan is annotated with the action's preconditions, thus enabling action monitoring. It is slightly

*Han monitoring means that finally, after 424 pages, we have an agent that is smarter than a dung beetle (see page 39). A plan-monitoring agent would notice that the dung ball was missing from its grasp and would replan to get another ball and plug its hole.
more complex to enable plan monitoring. Partial-order and planning-graph planners have the advantage that they have already built up structures that contain the relations necessary for plan monitoring. Augmenting state-space planners with the necessary annotations can be done by careful bookkeeping as the goal fluents are regressed through the plan.

Now that we have described a method for monitoring and replanning, we need to ask, "Does it work?" This is a surprisingly tricky question. If we mean, "Can we guarantee that the agent will always achieve the goal?" then the answer is no, because the agent could inadvertently arrive at a dead end from which there is no repair. For example, the vacuum agent might have a faulty model of itself and not know that its batteries can sun out. Once they do, it cannot repair any plans. If we rule out dead ends—assume that there exists a plan to reach the goal from any state in the environment—and assume that the environment is really nondeterministic, in the sense that such a plan always has some chance of success on any given execution attempt, then the agent will eventually reach the goal.

Trouble occurs when an action is actually not nondeterministic, but rather depends on some precondition that the agent does not know about. For example, sometimes a paint can may be empty, so painting from that can has no effect. No amount of retrying is going to change this. One solution is to choose randomly from among the set of possible repair plans, rather than to try the same one each time. In this case, the repair plan of opening another can might work. A better approach is to learn a better model. Every prediction failure is an opportunity for learning: an agent should be able to modify its model of the world to accord with its percepts. From then on, the replanner will be able to come up with a repair that gets at the root problem, rather than relying on luck to choose a good repair. This kind of learning is described in Chapters 18 and 19.

11.4 MULTIAGENT PLANNING

So far, we have assumed that only one agent is doing the sensing, planning, and acting. When there are multiple agents in the environment, each agent faces a multiagent planning problem in which it tries to achieve its own goals with the help or hindrance of others.

Between the purely single-agent and truly multiagent cases is a wide spectrum of problems that exhibit various degrees of decomposition of the monolithic agent. An agent with multiple effectors that can operate concurrently—for example, a human who can type and speak at the same time—needs to do multieffector planning to manage each effector while handling positive and negative interactions among the effectors. When the effectors are physically decoupled into detached units—as in a fleet of delivery robots in a factory—multieffector planning becomes multibody planning. A multibody problem is still a "standard" single-agent problem as long as the relevant sensor information collected by each body can be pooled—either centrally or within each body—to form a common estimate of the world state that then informs the execution of the overall plan; in this case, the multiple bodies act as a single body. When communication constraints make this impossible, we have

Futile repetition of a plan repair is exactly the behavior exhibited by the sphex wasp (page 39).