the effect $Own(i')$, yielding the substitution $\theta = \{i'/9780134610993\}$. Then we would regress over the action $Subst(\theta, A)$ to yield the predecessor state description $ISBN(9780134610993)$. This is part of the initial state, so we have a solution and we are done, having considered just one action, not a trillion.

More formally, assume a goal description g that contains a goal literal g_i and an action schema A. If A has an effect literal e'_j where $Unify(g_i, e'_j) = \theta$ and where we define $A' = \textsc{Subst}(\theta, A)$ and if there is no effect in A' that is the negation of a literal in g, then A' is a relevant action towards g.

For most problem domains backward search keeps the branching factor lower than forward search. However, the fact that backward search uses states with variables rather than ground states makes it harder to come up with good heuristics. That is the main reason why the majority of current systems favor forward search.

11.2.3 Planning as Boolean satisfiability

In Section 7.7.4 we showed how some clever axiom-rewriting could turn a wumpus world problem into a propositional logic satisfiability problem that could be handed to an efficient satisfiability solver. SAT-based planners such as SATPLAN operate by translating a PDDL problem description into propositional form. The translation involves a series of steps:

- Propositionalize the actions: for each action schema, form ground propositions by substituting constants for each of the variables. So instead of a single $Unload(c, p, a)$ schema, we would have separate action propositions for each combination of cargo, plane, and airport (here written with subscripts), and for each time step (here written as a superscript).
- Add action exclusion axioms saying that no two actions can occur at the same time, e.g. $\neg(FlyP_1SFOJFK^1 \wedge FlyP_1SFOBUH^1)$.
- Add precondition axioms: For each ground action A^t, add the axiom $A^t \Rightarrow \textsc{Pre}(A)^t$, that is, if an action is taken at time t, then the preconditions must have been true. For example, $FlyP_1SFOJFK^1 \Rightarrow At(P_1, SFO) \wedge Plane(P_1) \wedge Airport(SFO) \wedge Airport(JFK)$.
- Define the initial state: assert F^0 for every fluent F in the problem's initial state, and $\neg F^0$ for every fluent not mentioned in the initial state.
- Propositionalize the goal: the goal becomes a disjunction over all of its ground instances, where variables are replaced by constants. For example, the goal of having block A on another block, $On(A, x) \wedge Block(x)$ in a world with objects A, B and C, would be replaced by the goal

  $$(On(A, A) \wedge Block(A)) \vee (On(A, B) \wedge Block(B)) \vee (On(A, C) \wedge Block(C)).$$

- Add successor-state axioms: For each fluent F, add an axiom of the form

  $$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t),$$

  where $ActionCausesF$ stands for a disjunction of all the ground actions that add F, and $ActionCausesNotF$ stands for a disjunction of all the ground actions that delete F.

The resulting translation is typically much larger than the original PDDL, but the efficiency of modern SAT solvers often more than makes up for this.