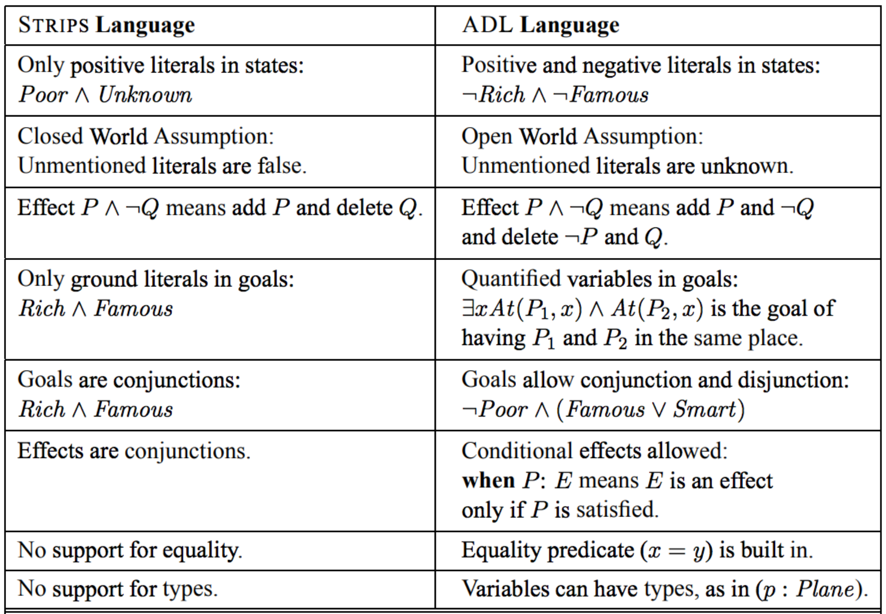
Historical Development in AI : Planning & Search

http://aima.cs.berkeley.edu/2nd-ed/newchap11.pdf

The most difficulty part is the problem-solving agent can be overwhelmed by irrelevant actions. The next difficulty is finding a good heuristic function. Suppose the agent's goal is to buy four different books online. Then there will be 1040 plans of just four steps, so searching without an accurate heuristic is out of the question. It is obvious that a good heuristic estimate for the cost of a state is the number of books that remain to be bought; unfortunately, this insight is not obvious to a problem-solving agent, because it sees the goal test only as a black box that returns true or false for each state. The problem solver might be inefficient because it cannot take advantage of problem decomposition. Consider the problem of delivering a set of overnight packages to problem decomposition their respective destinations, which are scattered across Australia. *It makes sense to find out the nearest airport for each destination and divide the overall problem into several subproblems, one for each airport. The various planning formalisms used in AI have been systematized within a standard syntax called the the Planning Domain Definition Language, or PDDL. This language allows researchers to exchange benchmark problems and compare results. PDDL includes sublanguages for STRIPS, ADL(as left figure)*:



STRIPS **Language**	ADL **Language**
Only positive literals in states: *Poor* $\wedge$ *Unknown*	Positive and negative literals in states: $\neg$*Rich* $\wedge$ $\neg$*Famous*
Closed World Assumption: Unmentioned literals are false.	Open World Assumption: Unmentioned literals are unknown.
Effect $P \wedge \neg Q$ means add P and delete Q.	Effect $P \wedge \neg Q$ means add P and $\neg Q$ and delete $\neg P$ and Q.
Only ground literals in goals: *Rich* $\wedge$ *Famous*	Quantified variables in goals: $\exists x At(P_1, x) \wedge At(P_2, x)$ is the goal of having P_1 and P_2 in the same place.
Goals are conjunctions: *Rich* $\wedge$ *Famous*	Goals allow conjunction and disjunction: $\neg$*Poor* $\wedge$ (*Famous* $\vee$ *Smart*)
Effects are conjunctions.	Conditional effects allowed: **when** P: E means E is an effect only if P is satisfied.
No support for equality.	Equality predicate $(x = y)$ is built in.
No support for types.	Variables can have types, as in $(p : Plane)$.

Figure 11.1 Comparison of STRIPS and ADL languages for representing planning problems. In both cases, goals behave as the preconditions of an action with no parameters.

PARTIAL-ORDER PLANNING

Forward and backward state-space search are particular forms of totally ordered plan search. They explore only strictly linear sequences of actions directly connected to the start or goal. This means that they cannot take advantage of problem decomposition. Rather than work on each subproblem separately, always make decisions about how to sequence actions from all the subproblems. Such an approach also has the advantage of flexibility in the order in which it constructs the plan.