planning

goal oriented action planning (GOAP)

search for a plan that achieves the goal in the current state

STRIPS planning

- typical description of a planning problem

  initial state, goal, available actions
- typical solution: a sequence pf actions
- or, a method for every initial state & goal /

  for every application domain

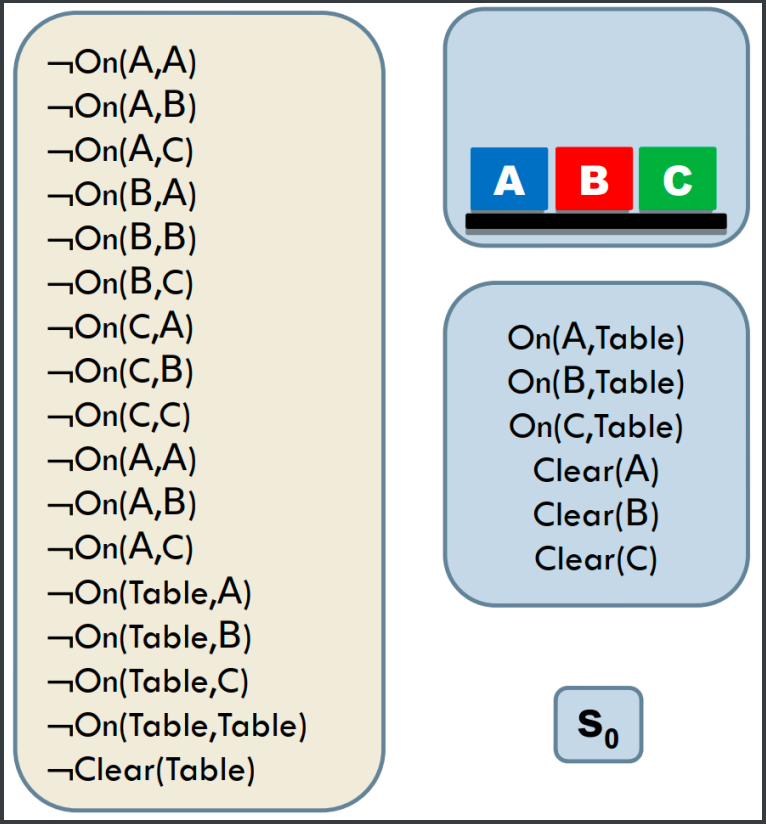
initial state

- representation - atomic formulas (predicates)

  using **FOL**, with
  - ground
  - function-free
  - positive

  the list of terms typically forms a logical conjunction
- completely specified: closed-world assumption

  atomic sentence not mentioned - assumed to be false



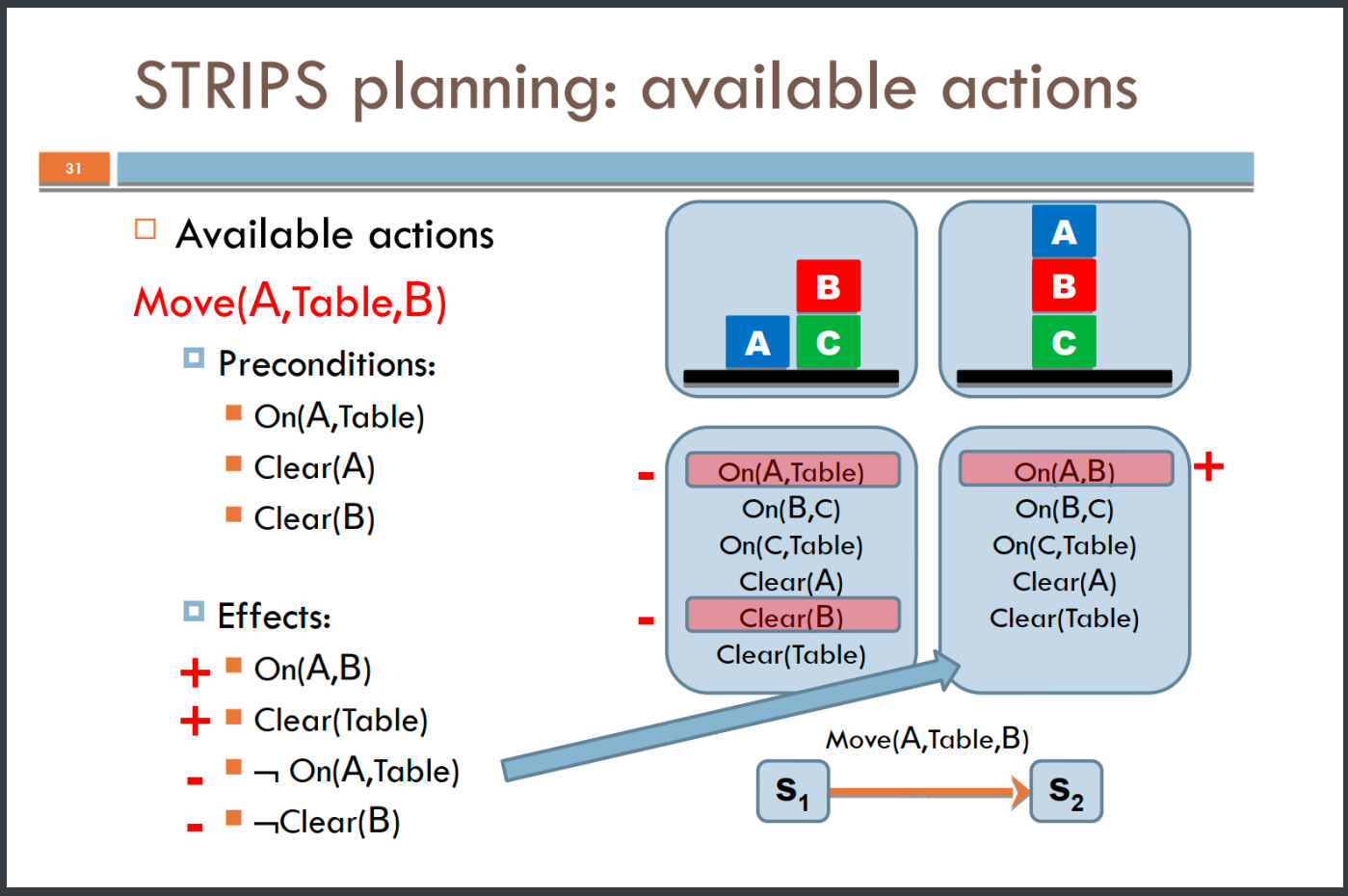
goal

- also using FOLs, with ground, function-free, positive (conjunctive)
- partially specified: no closed-world assumption

  a state *s* satisfies goal *g* if it contains all literals in *g*, more literals may be in *s*

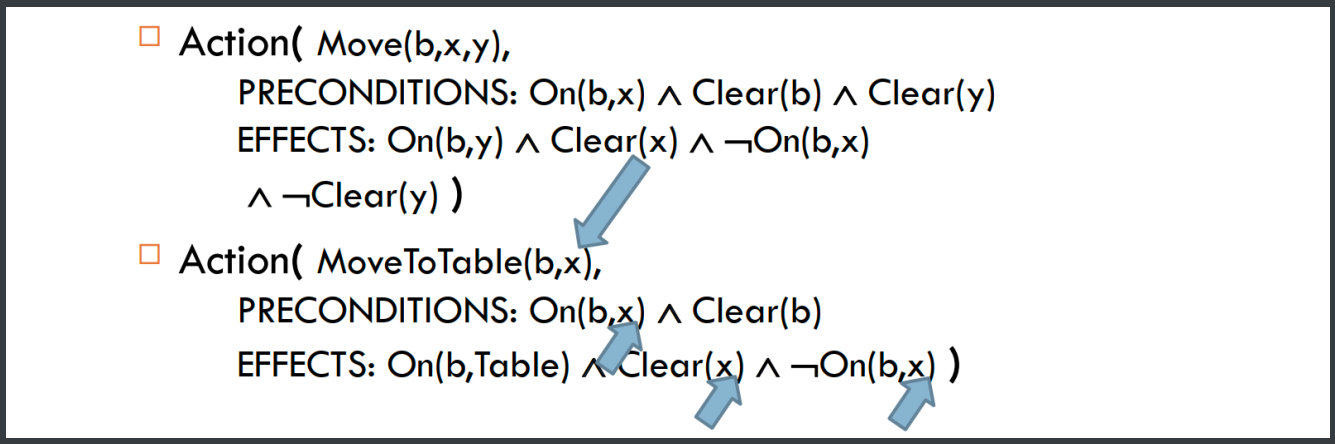
available actions

preconditions & effects



- problem here: Clear(Table) needs to be treated differently

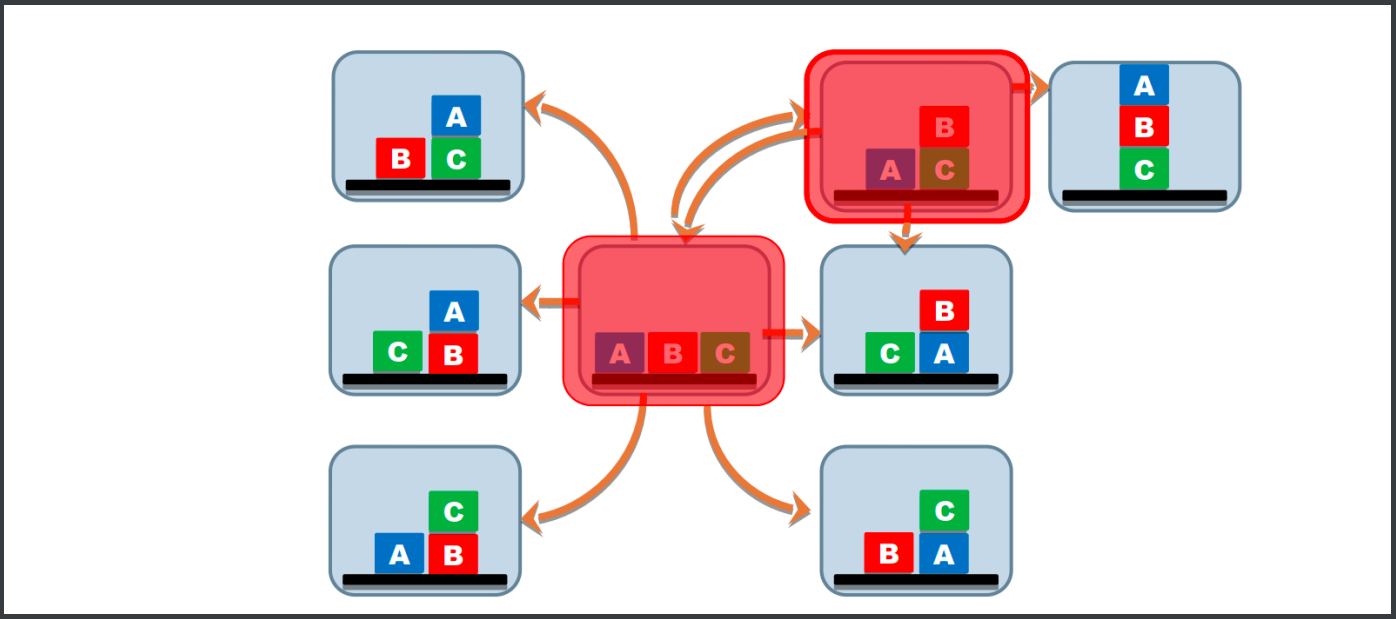
  solution: separate "normal move" and "move to table"



  ! variables that appear in preconditions and effects need to be parameters of the action schema

STRIPS planning: state-based, progression search

find successors -> pick one of the not-visited successors



- guaranteed to find a solution if one exists?

  yes! if state-space if finite. as long as we visit each state only once

heuristics for progression planning

- f(s) = g(s) + h(s) what's different than grid-based problems then?

  action schemas provide **domain independent** heuristic functions

- examples

  - Simple example: h(s) = number of literals in the goal that are missing from s
  - Empty list of preconditions
    - h(s) = number of actions needed to achieve the goal if we assume that all actions are always applicable
  - Empty list of negative effects (delete-relaxation)
    - h(s) = number of actions needed to achieve the goal if we disregard the negative effects of actions

- planning: pick the most promising of the successor states

regression planning (backward, not covered)

planning domain definition language (PDDL)