Planning Graphs and Graphplan

Section 11.4

Outline

- The planning graph
- Planning graph example
- The graphplan algorithm
- Using planning graphs for heuristics

Additional reference used for the slides:
Weld, D.S. (1999). Recent advances in AI planning. *AI Magazine*, 20(2), 93-122.

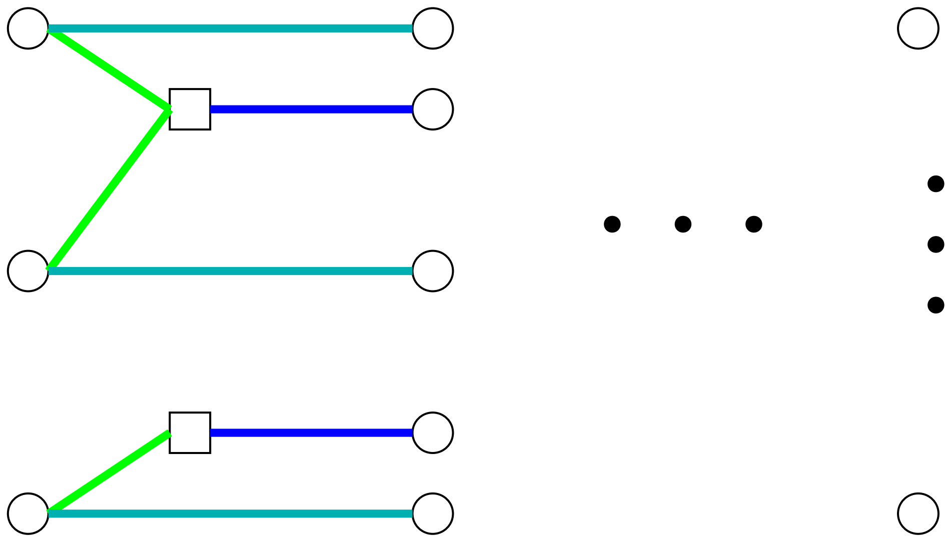
A planning graph

- A layered graph
- Two kinds of layers alternate
  - literal (proposition) (shown with circles)
  - action (shown with squares)
- Every two layers corresponds to a discrete time
- No variables as in action schemas

A planning graph

- The first layer is a literal layer which shows all the literals that are true in the initial layer
- Every action has a link from each of its preconditions and a link to each of its effects.
- Straight lines between to literals at consecutive literal levels denote *NoOp*

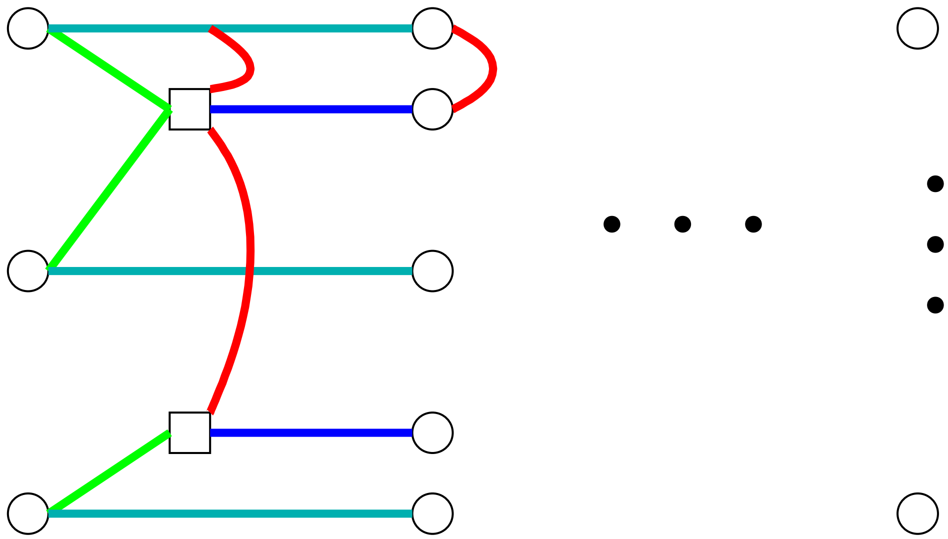
A planning graph



A planning graph

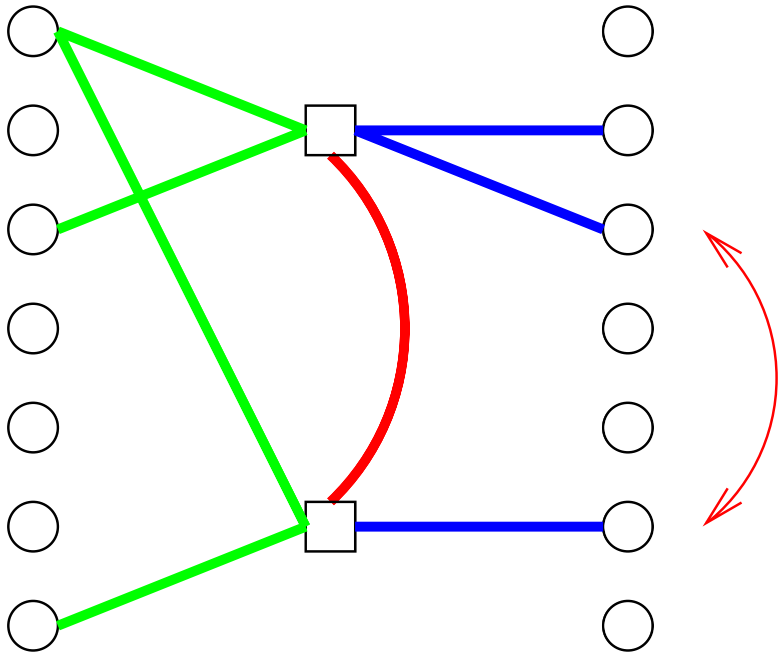
- The red lines show *mutex relationships*
- Those are the literals and actions that are mutually exclusive, i.e., cannot appear at the same time

A planning graph



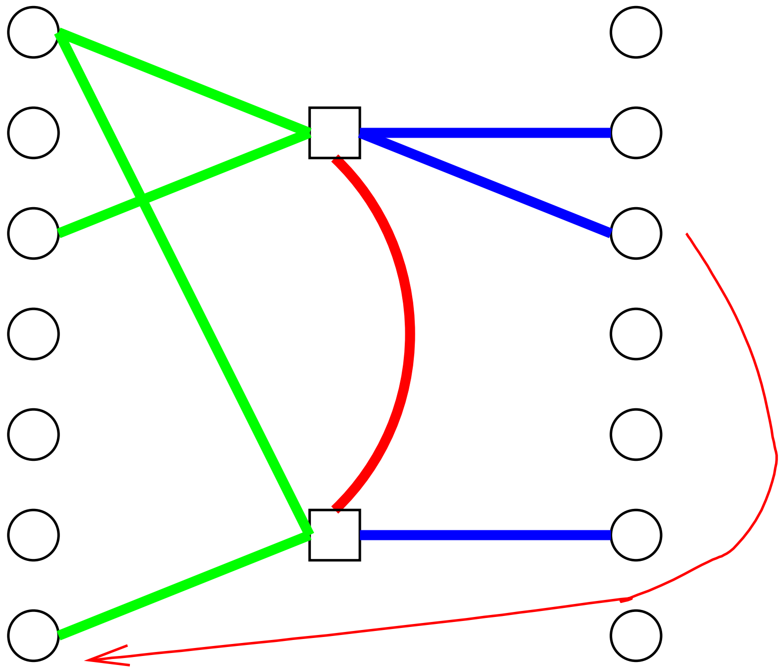
Mutex relationships for actions

- *Inconsistent effects*: one action negates the effect of the other



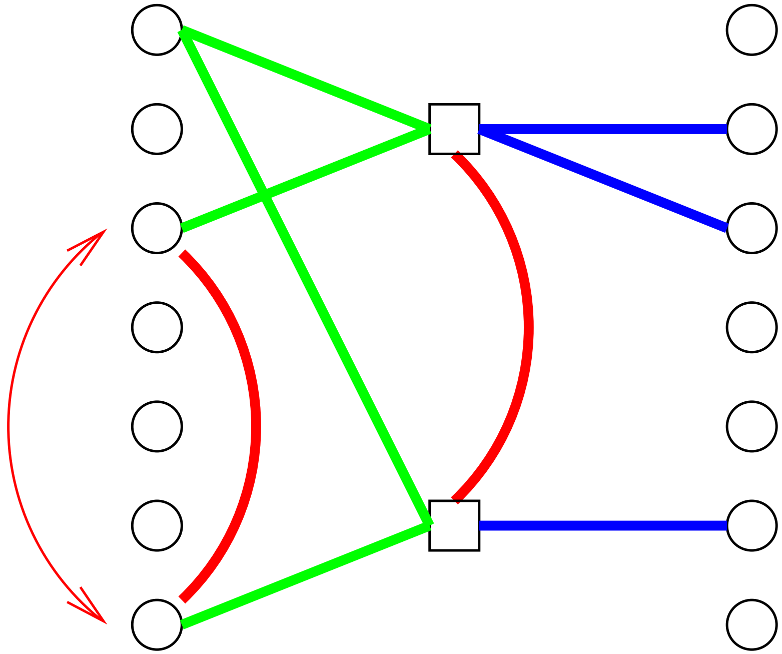
Mutex relationships for actions

- *Interference*: one of the effects of one action is the negation of a precondition of the other



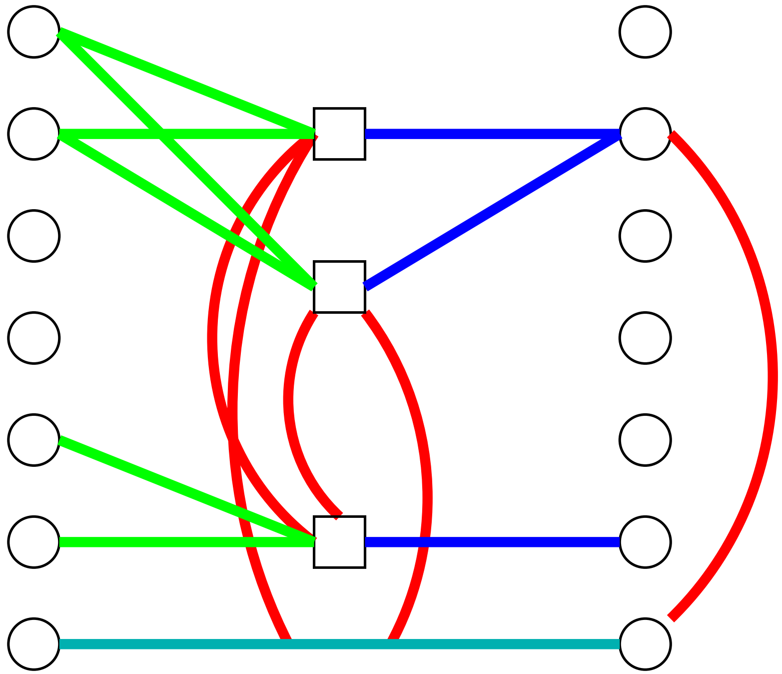
Mutex relationships for actions

- *Competing needs*: one of the preconditions of one action is mutually exclusive with a precondition of another



Mutex relationships for literals

- One is the negation of the other
- *Inconsistent support*: all ways of achieving two literals is mutually exclusive

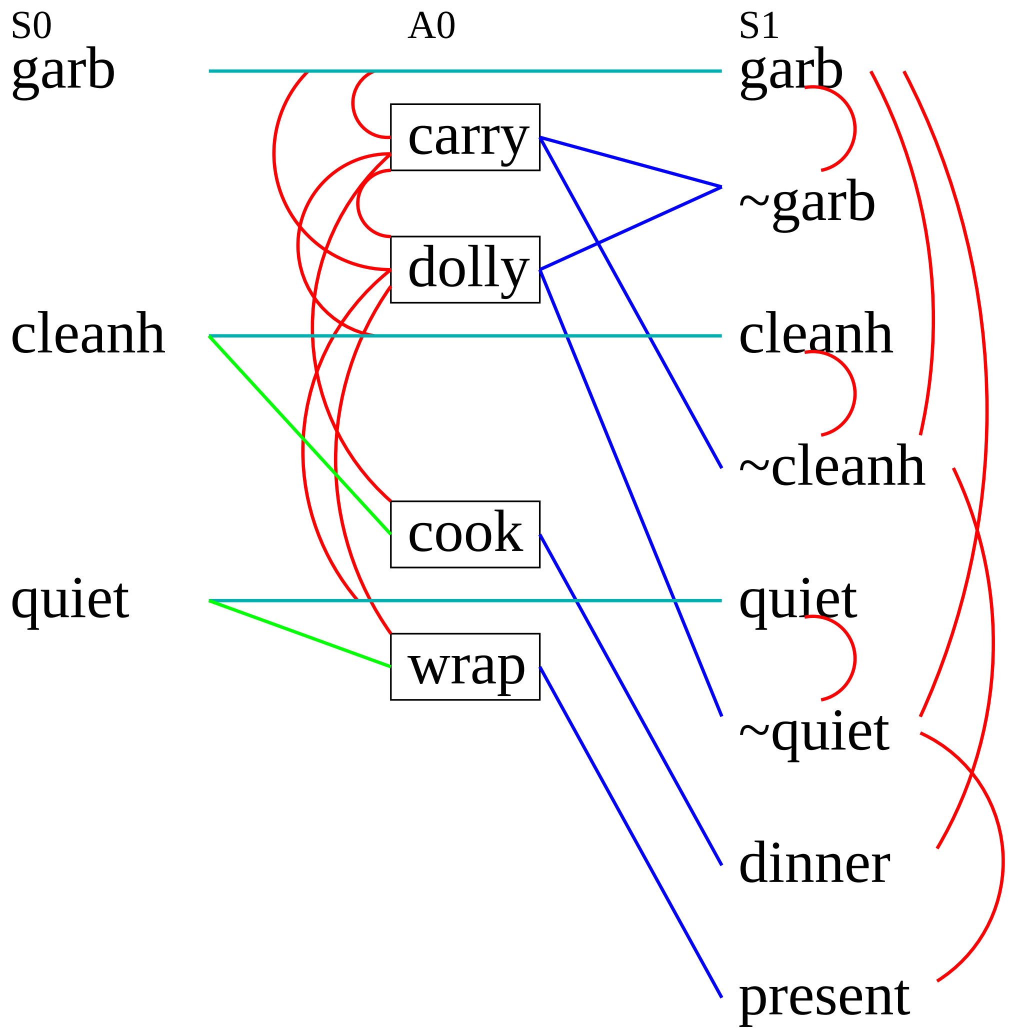


Example

Initial conditions: garbage, cleanhands, quiet
Goal: dinner, present, ~garbage
Operators:

- Cook:
  - pre: cleanhands
  - eff: dinner
- Wrap:
  - pre: quiet
  - eff: present
- Carry:
  - pre:
  - eff: ~garbage, ~cleanhands
- Dolly:
  - pre:
  - eff: ~garbage, ~quiet

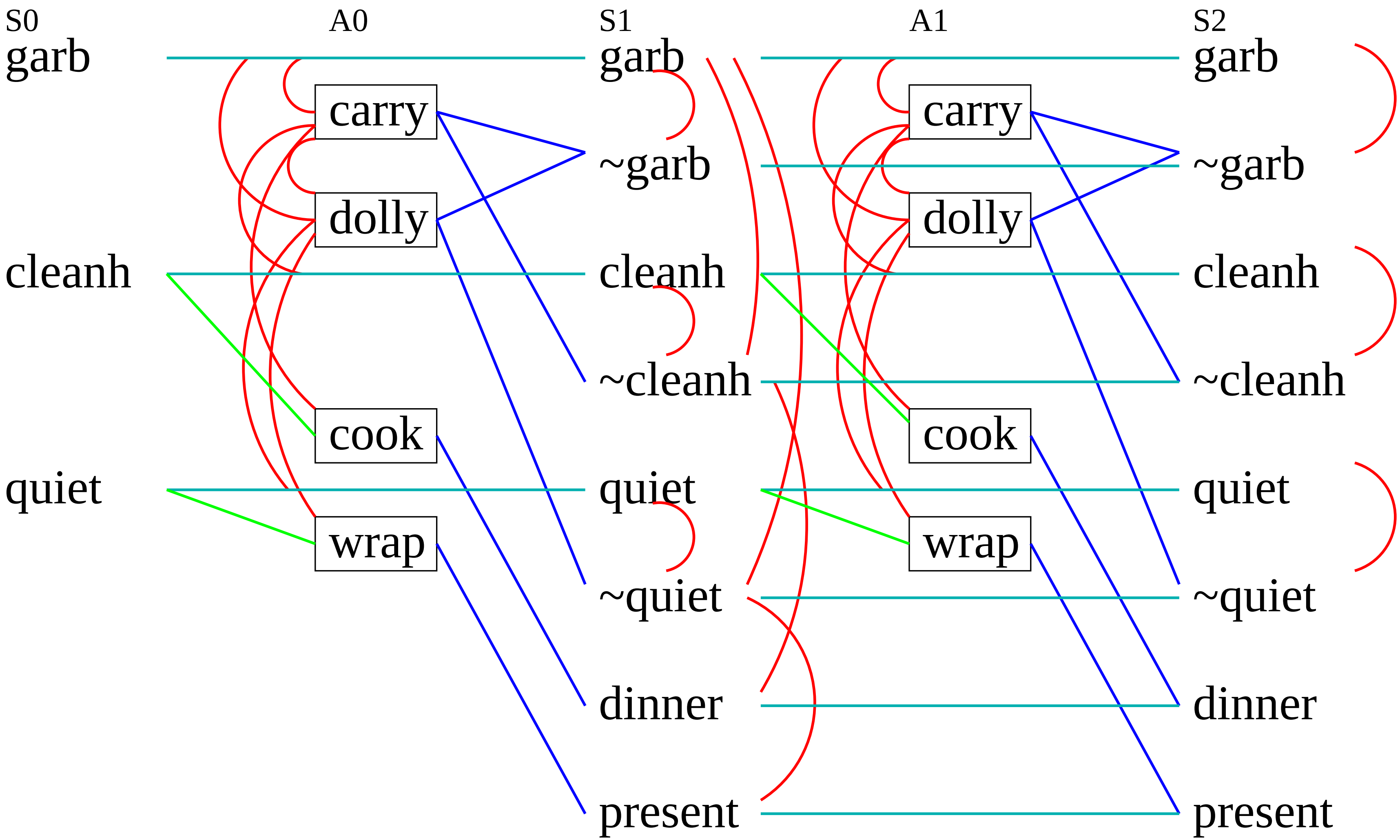
Example: graph expanded to level S_1



Solution extraction (first attempt)

- At level S_1, all the goal conditions are present, can look for a solution
- There are two ways to satisfy dinner, present, ˜ garbage:
  {carry; cook; wrap}
  {dolly; cook; wrap}
- carry is mutex with cook
  dolly is mutex with wrap
- Solution extraction fails, need to expand the graph

Example: graph expanded to level S_2



Solution extraction (second attempt)

- At level S_2, all the goal conditions are still present. In fact, they do not go away once they appear.
- Notice that there are fewer mutex relationships at level S_2
- There are three ways to satisfy ˜ garbage:
  carry, dolly, noop
  There are two ways to satisfy present:
  wrap, noop
  There are two ways to satisfy dinner:
  cook, noop
  So, look for at all 3 $\times$ 2 $\times$ 2 = 12 combinations

Solution extraction (second attempt)

- Support ~ garbage with carry, dinner with noop, and present with wrap
  This is a *consistent* set because none are mutually exclusive
- The subgoals from level A_1 are dinner (precondition of noop), quiet (precondition of wrap)
  There are only two subgoals because dolly and carry do not have preconditions
- Choose cook to support dinner, and noop for quiet. These two actions are not mutex.
- If there are multiple actions at a level, they can be executed in parallel.

The GRAPHPLAN algorithm

function GRAPHPLAN (*problem*)
returns a solution, or failure

```
graph ← INITIAL-PLANNING-GRAPH (problem)
goals ← GOALS[problem]
loop do
  if goals all non-mutex in last level of graph then do
  solution ← EXTRACT-SOLUTION ( graph, goals, LENGTH (graph))
  if solution ≠ failure then return solution
  else if NO-SOLUTION-POSSIBLE(graph) then return failure
graph ← EXPAND-GRAPH (graph, problem)
```

Properties of planning graphs

- A literal that does not appear in the final level of the graph cannot be achieved by any plan.
- The *level cost* of a goal literal is the first level it appears,
  e.g., 0 for cleanhands and 1 for dinner.
- Level cost is an admissible heuristic but might undercount: it counts the number of levels, whereas there might be several actions at each level
  $\rightarrow$ use a *serial planning graph*

Heuristics derived from planning graphs

- The *max level heuristic* takes the maximum level cost of any of the goals (admissible, not very accurate)
- The *level cost heuristic* returns the sum of the level costs of the goals (inadmissible, works well in practice)
- The *set level heuristic* finds the level at which all the literals in the conjunctive goal appear in the planning graph without any pair of them being mutually exclusive (dominates max level, works well when the subplans interact a lot)