Planning 1

Chapter 11.1-11.3

Some material adopted from notes by
Andreas Geyer-Schulz and Chuck Dyer

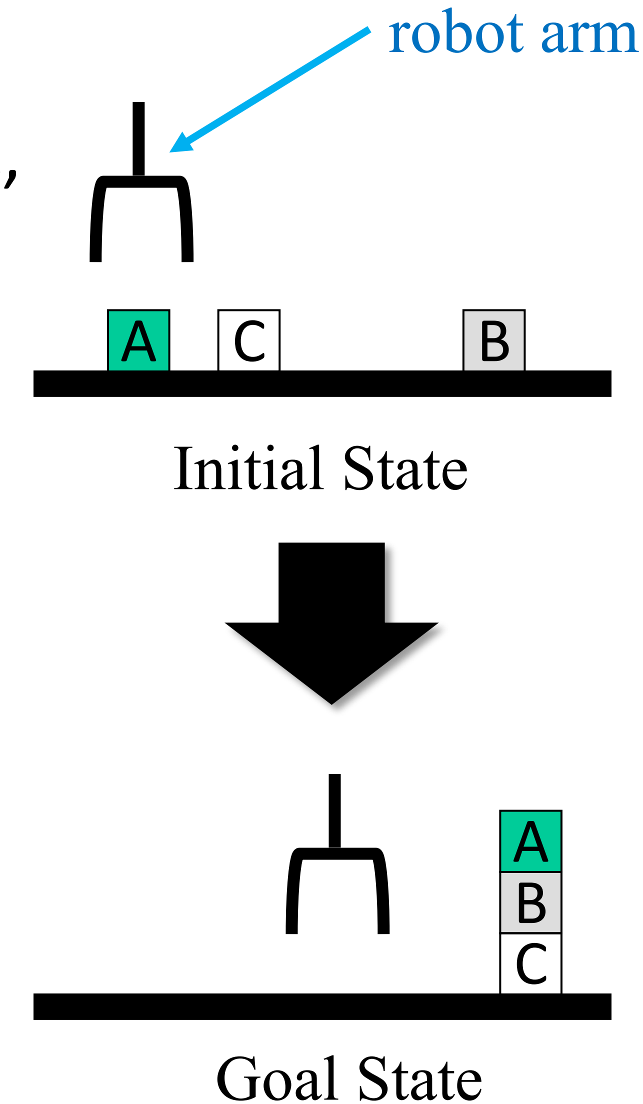
Planning is the art and practice of thinking before acting

— Patrik Haslum

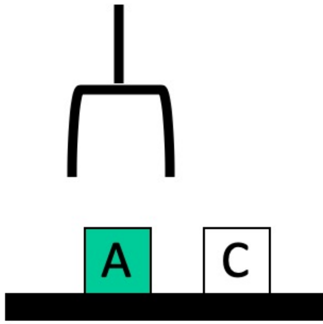
Classic Planning

Find **sequence of actions** to reach a **goal** in a discrete, deterministic, static, fully-observable environment

- State space search and logical reasoning could be used
- But classic planning developed custom representations & algorithms to do it more effectively
- The approach uses a **knowledge base** and reasoning about the state of the world and possible actions
- We'll look first at doing this in the simple **blocks world**



Blocks world

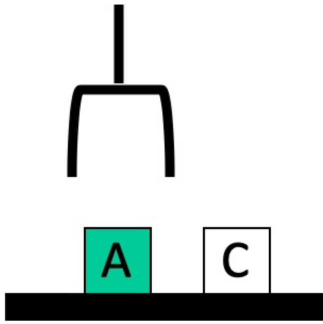


The blocks world is a “micro-world” with a **table**, a set of **blocks**, and a **robot hand**

Some constraints for a simple model:

- Only one block can be on another block
- Any number of blocks can be on the table
- The hand can only hold one block

Blocks world



Typical representation uses a logic notation to represent the state of the world:

ontable(a) ontable(c)

clear(a) clear(c)

handempty

And possible actions with their preconditions and effects:

Pickup Putdown

Stack Unstack

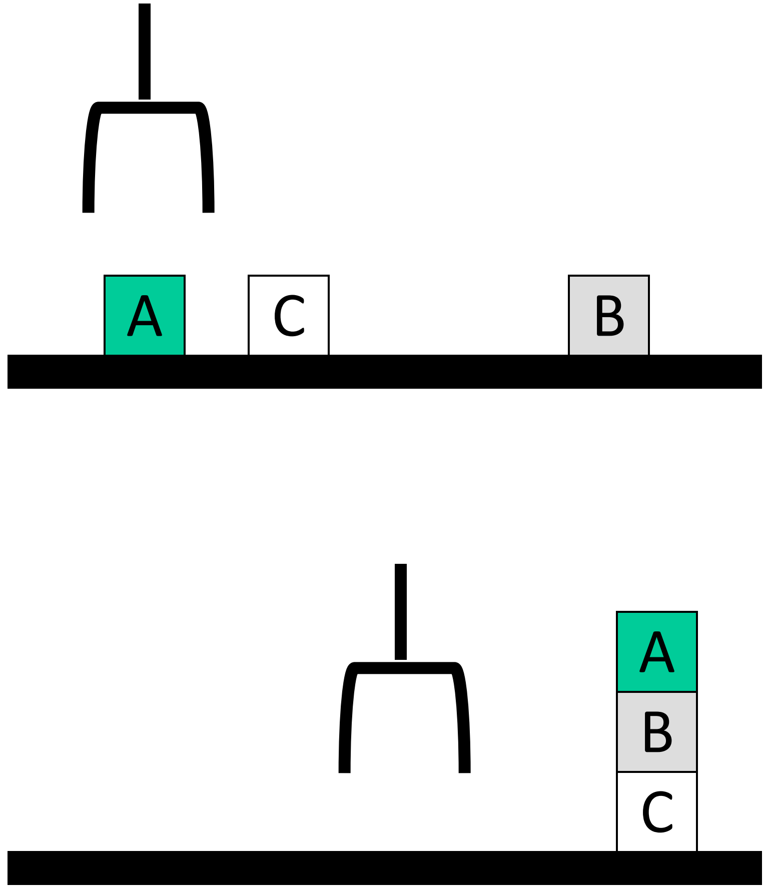
Typical BW planning problem

Initial state:

- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

Goal:

- on(b,c)
- on(a,b)
- ontable(c)



Typical BW planning problem

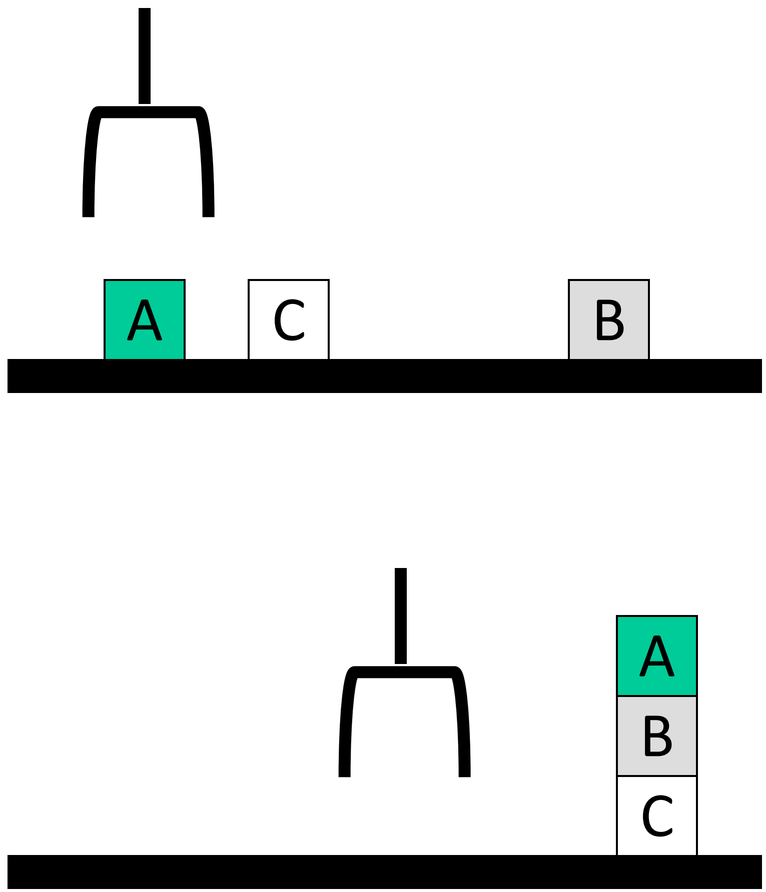
Logical assertions describing initial & final states

Initial state:

clear(a)
clear(b)
clear(c)
ontable(a)
ontable(b)
ontable(c)
handempty

Goal state:

on(b,c)
on(a,b)
ontable(c)



Sequence of robot actions

Plan:

pickup(b)
stack(b,c)
pickup(a)
stack(a,b)

Planning problem

- Find sequence of actions to achieve goal state when executed from initial state given
  - set of possible **primitive actions**, including their *preconditions* and *effects*
  - **initial state** description
  - **goal state** description
- Compute plan as a **sequence of actions** that, when executed in initial state, achieves goal state
- States specified as a KB , i.e. conjunction of conditions
  - e.g., $ontable(a) \land on(b, a)$

Planning vs. problem solving

- Problem solving methods solve similar problems
- Planning is more powerful and efficient because of the representations and methods used
- States, goals, and actions are decomposed into sets of sentences (usually in first-order logic)
- Search often proceeds through *plan space* rather than *state space* (though there are also state-space planners)
- Sub-goals can be planned independently, reducing the complexity of the planning problem

Typical simplifying assumptions

- **Atomic time**: Each action is indivisible
- **No concurrent actions**: but actions need not be ordered w.r.t. each other in the plan
- **Deterministic actions**: action results completely determined — no uncertainty in their effects
- Agent is the **sole cause** of change in the world
- Agent is **omniscient** with complete knowledge of the state of the world
- **Closed world assumption**: everything known to be true included in state description; anything not listed is false

Real AI planning systems can relax many of these

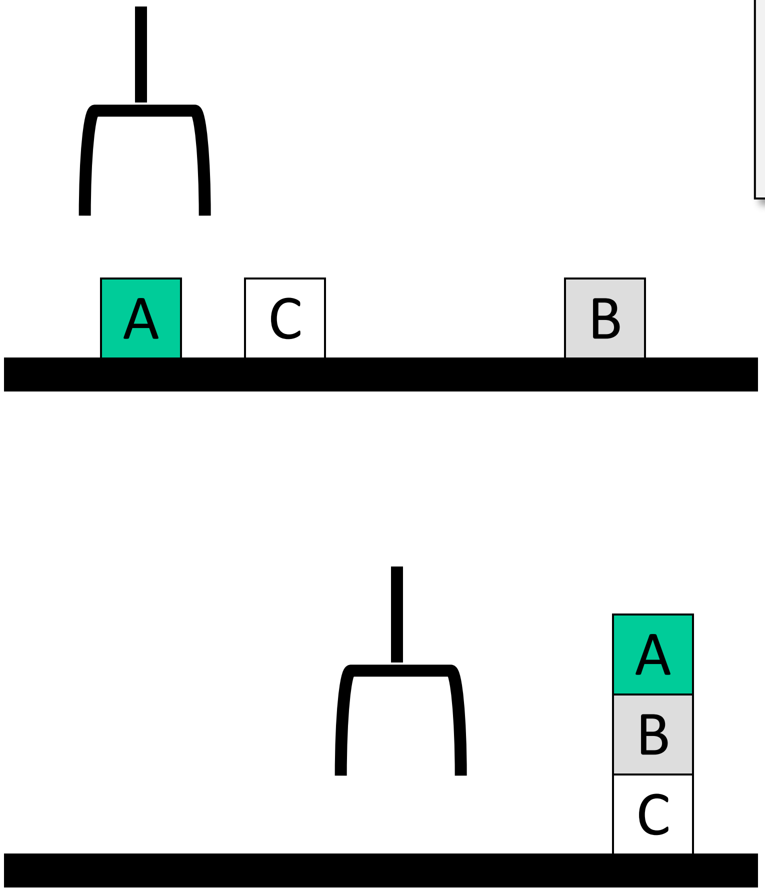
Typical BW planning problem

Initial state:

clear(a)
clear(b)
clear(c)
ontable(a)
ontable(b)
ontable(c)
handempty

Goal:

on(b,c)
on(a,b)
ontable(c)



Simple approach:

- find a way to achieve each goal in order

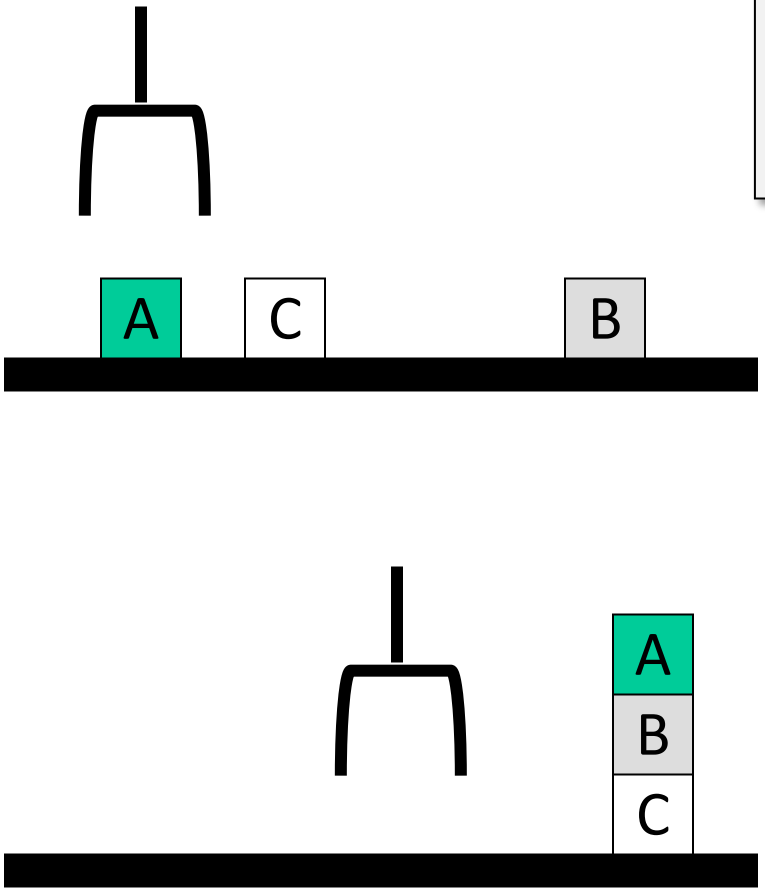
Typical BW planning problem

Initial state:

clear(a)
clear(b)
clear(c)
ontable(a)
ontable(b)
ontable(c)
handempty

Goal:

on(b,c)
on(a,b)
ontable(c)



Simple approach:

- find a way to achieve each goal in order

A plan:

pickup(b)
stack(b,c)
pickup(a)
stack(a,b)

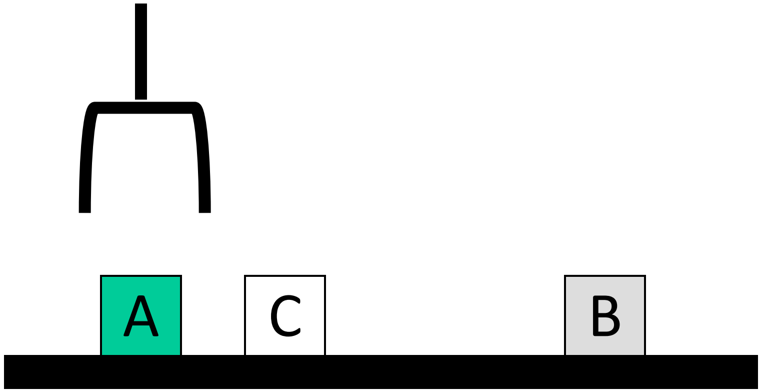
Another BW planning problem

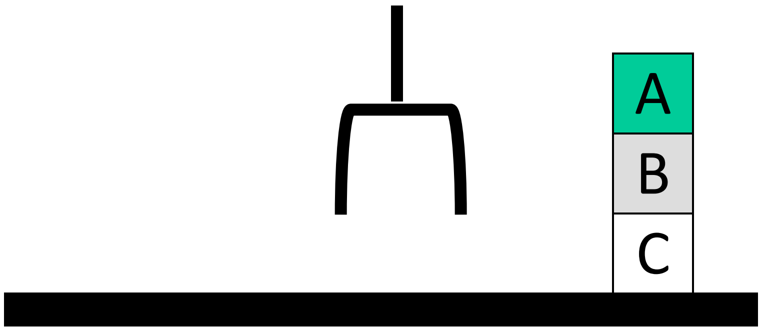
Initial state:

- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

Goal:

- on(a,b)
- on(b,c)
- ontable(c)





Simple approach:

- find a way to achieve each goal in order

Note: Goals in a different order!

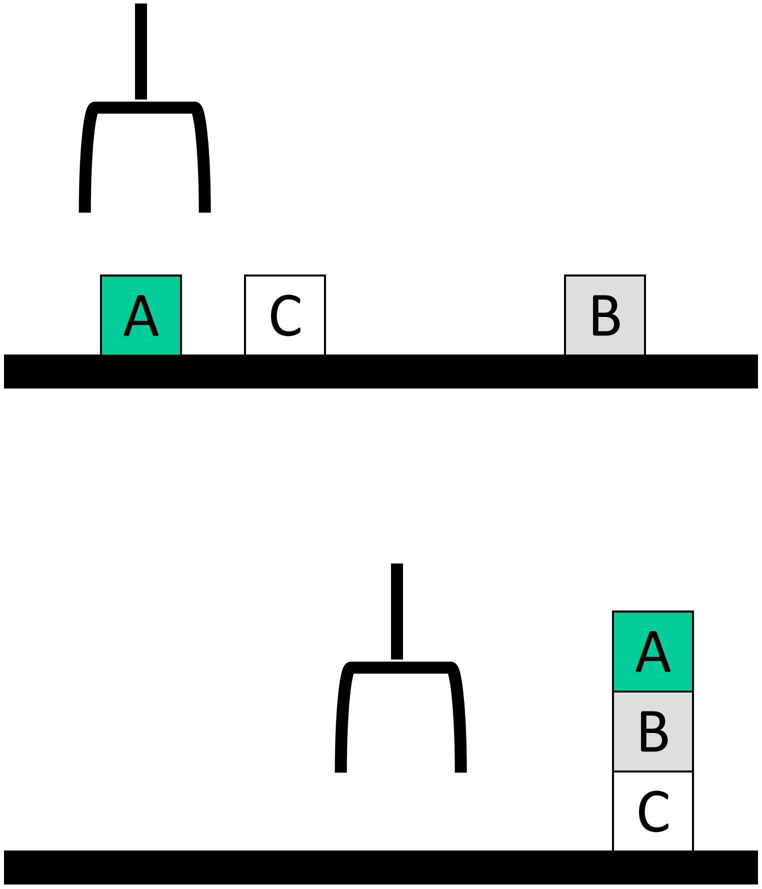
Another BW planning problem

Initial state:

- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

Goal:

- on(a,b)
- on(b,c)
- ontable(c)



A plan:

- pickup(a)
- stack(a,b)
- unstack(a,b)
- putdown(a)
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

Note: Goals in a different order!

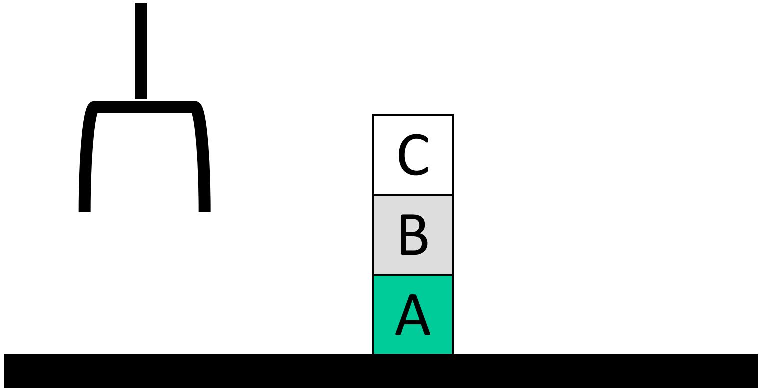
Yet Another BW planning problem

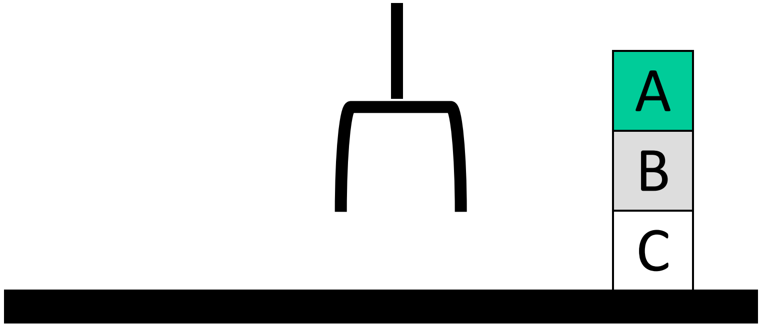
Initial state:

- clear(c)
- ontable(a)
- on(b,a)
- on(c,b)
- handempty

Goal:

- on(a,b)
- on(b,c)
- ontable(c)





Plan:

- unstack(c,b)
- putdown(c)
- unstack(b,a)
- putdown(b)
- pickup(a)
- stack(a,b)
- unstack(a,b)
- putdown(a)
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

Note: not very efficient!

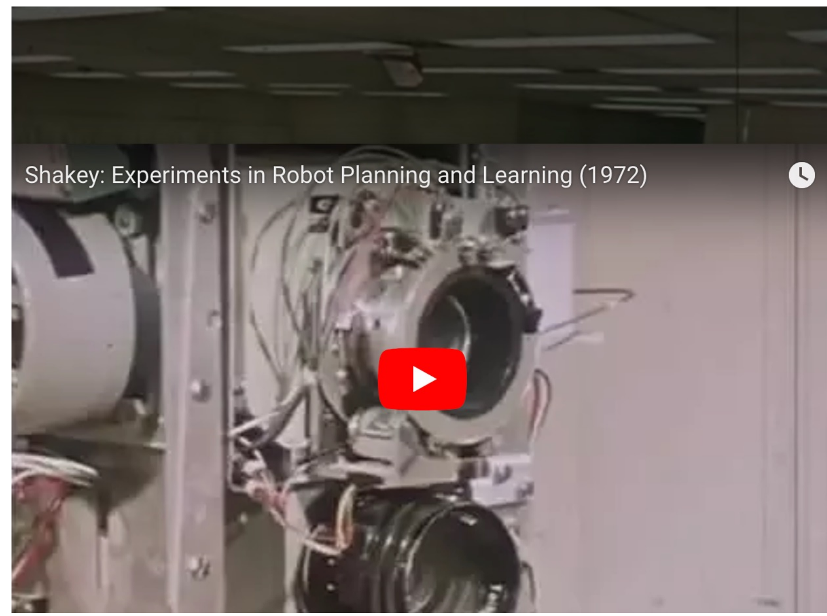
Major approaches

- Planning as search
- GPS / STRIPS
- Situation calculus
- Partial order planning
- Hierarchical decomposition (HTN planning)
- Forward planning with heuristics
- Planning with constraints (SATplan, Graphplan)
- Reactive planning

History: Shakey the robot

First general-purpose mobile robot to be able to reason about its own actions

Shakey the Robot: 1st Robot to Embody Artificial Intelli-gence (2017, 6 min.)



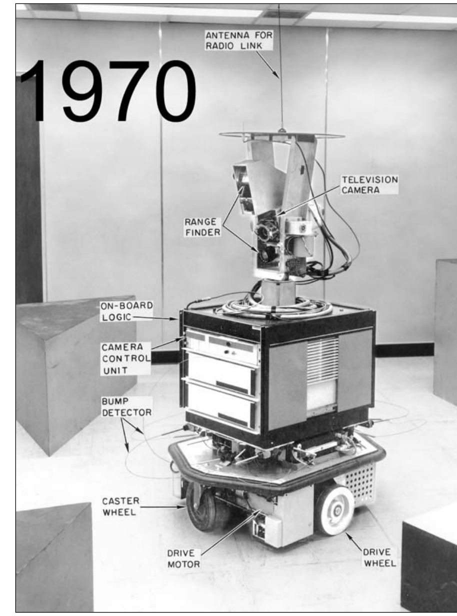
Shakey: Experiments in Robot Planning and Learning (1972, 24 min)

Strips planning representation

- Classic approach first used in the STRIPS (Stanford Research Institute Problem Solver) planner
- A State is a conjunction of ground literals

  at(Home) $\wedge$ $\neg$have(Milk) $\wedge$ $\neg$have(bananas) ...
- Goals are conjunctions of literals, but may have variables, assumed to be existentially quantified

  at(?x) $\wedge$ have(Milk) $\wedge$ have(bananas) ...
- Need not fully specify state
  - Non-specified conditions either don't-care or assumed false
  - Represent many cases in small storage
  - May only represent changes in state rather than entire situation
- Unlike theorem prover, not seeking whether goal is true, but is there a sequence of actions to attain it



Shakey the robot

Blocks World Operators

- Classic basic operations for the Blocks World
  - **stack(X,Y):** put block X on block Y
  - **unstack(X,Y):** remove block X from block Y
  - **pickup(X):** pickup block X
  - **putdown(X):** put block X on the table
- Each represented by
  - list of **preconditions**
  - list of new facts to be added (**add-effects**)
  - list of facts to be removed (**delete-effects**)
  - optionally, set of (simple) variable **constraints**

Blocks World Stack Action

stack(X,Y):

- **preconditions**(stack(X,Y), [holding(X), clear(Y)])
- **deletes**(stack(X,Y), [holding(X), clear(Y)]).
- **adds**(stack(X,Y), [handempty, on(X,Y), clear(X)])
- **constraints**(stack(X,Y), [X≠Y, Y≠table, X≠table])

STRIPS planning

- STRIPS maintains two additional data structures:
  - State List - all currently true predicates
  - Goal Stack - push down stack of goals to be solved, with current goal on top
- If current goal not satisfied by present state, find action that adds it and push action and its preconditions (subgoals) on stack
- When a current goal is satisfied, POP from stack
- When an action is on top stack, record its application on plan sequence and use its add and delete lists to update current state

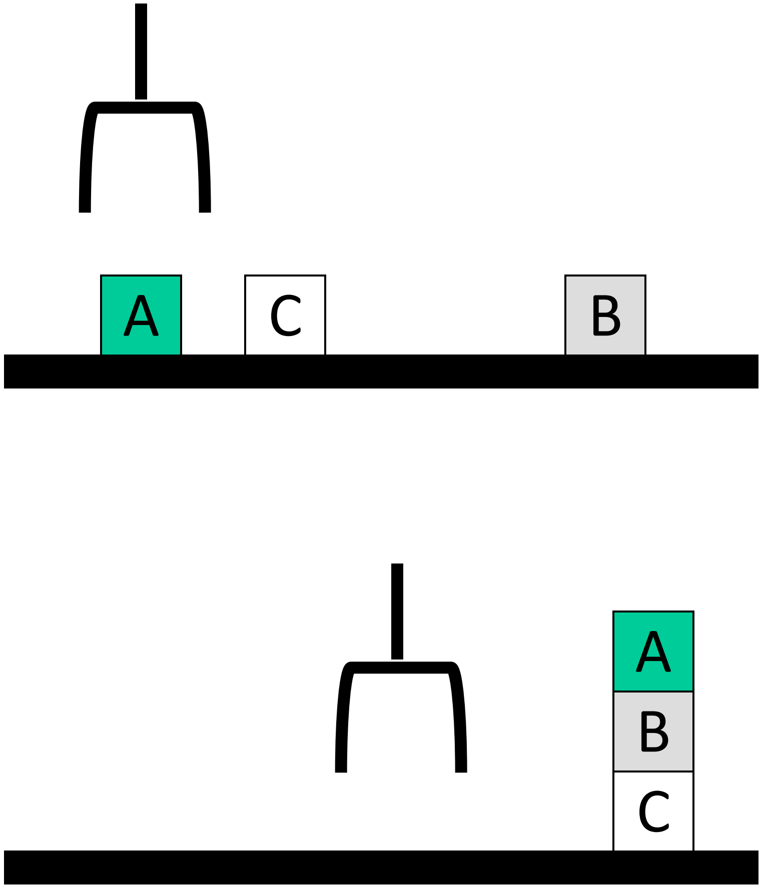
Typical BW planning problem

Initial state:

- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

Goal:

- on(b,c)
- on(a,b)
- ontable(c)



A plan:

- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

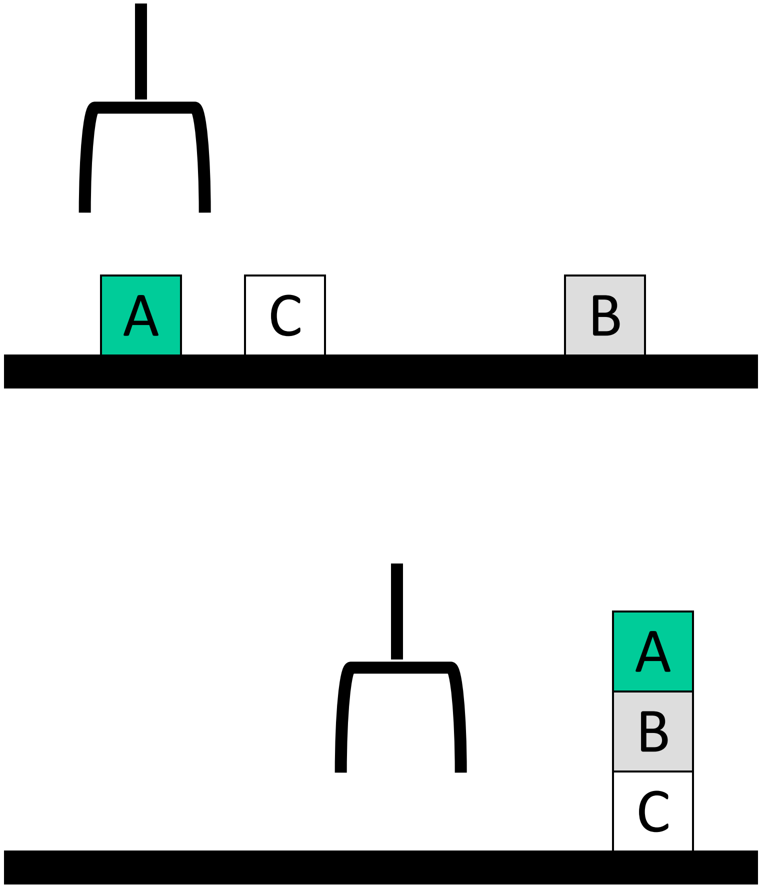
Another BW planning problem

Initial state:

- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

Goal:

- on(a,b)
- on(b,c)
- ontable(c)



A plan:

- pickup(a)
- stack(a,b)
- unstack(a,b)
- putdown(a)
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

😐

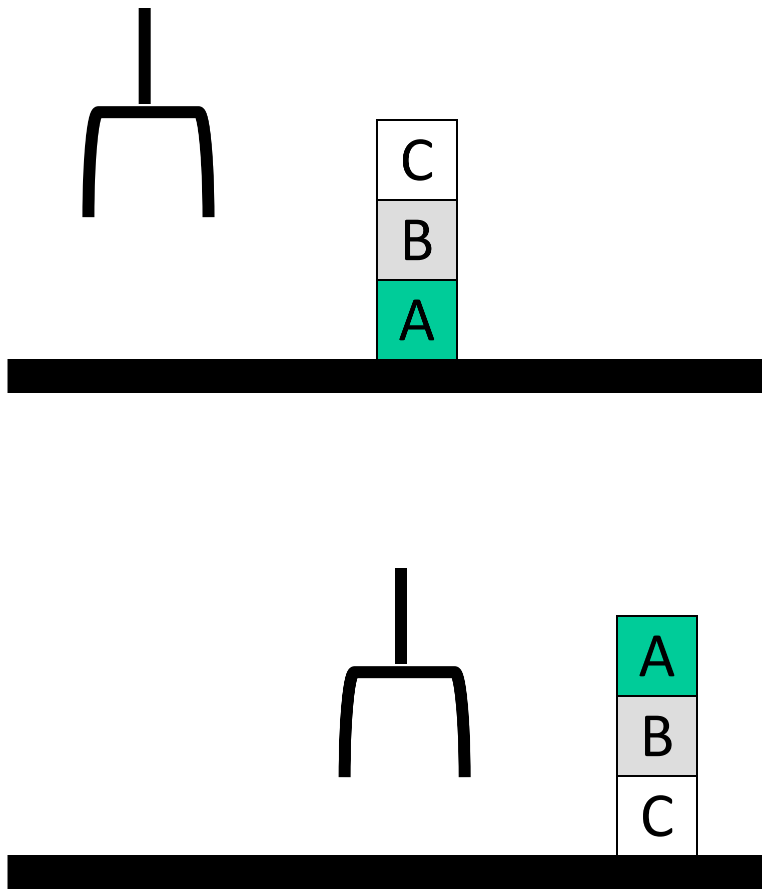
Yet Another BW planning problem

Initial state:

- clear(c)
- ontable(a)
- on(b,a)
- on(c,b)
- handempty

Goal:

- on(a,b)
- on(b,c)
- ontable(c)



Plan:

- unstack(c,b)
- putdown(c)
- unstack(b,a)
- putdown(b)
- pickup(b)
- stack(b,a)
- unstack(b,a)
- putdown(b)
- pickup(a)
- stack(a,b)
- unstack(a,b)
- putdown(a)
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

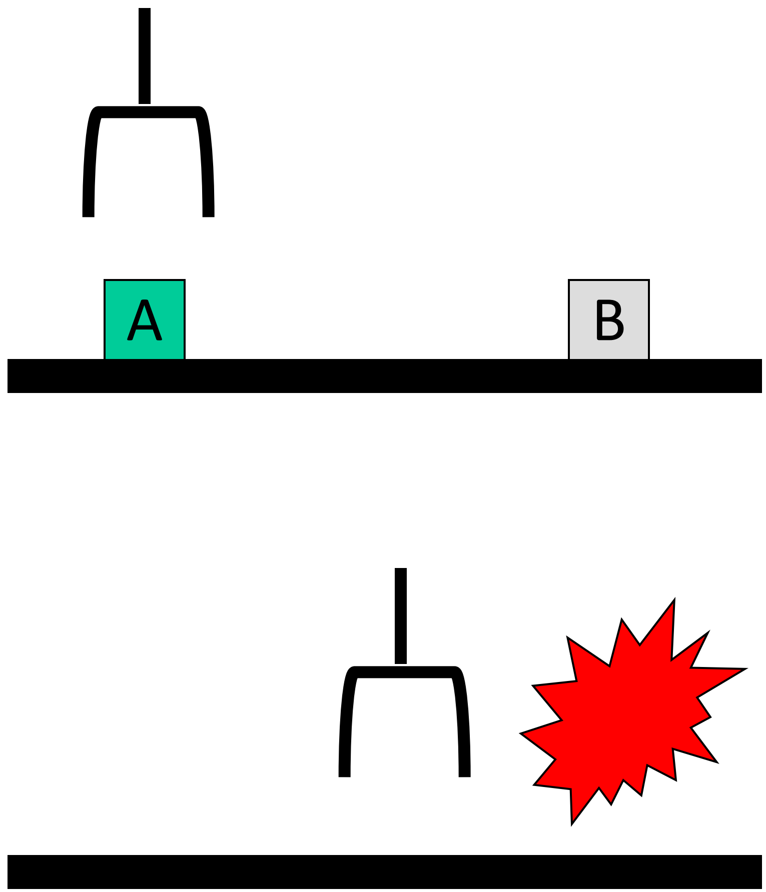
Yet Another BW planning problem

Initial state:

- ontable(a)
- ontable(b)
- clear(a)
- clear(b)
- handempty

Goal:

- on(a,b)
- on(b,a)



Plan:

- ??

Goal interaction

- Simple planning algorithms assume independent sub-goals
  - Solve each separately and concatenate the solutions
- Sussman Anomaly: an example of goal interaction problem:
  - Solving on(A,B) first (via unstack(C,A),stack(A,B)) is undone when solving 2nd goal on(B,C) (via unstack(A,B), stack(B,C))
  - Solving on(B,C) first will be undone when solving on(A,B)
- Classic STRIPS couldn't handle this, although minor modifications can get it to do simple cases

Initial state

Goal state

Fin