Planning

Planning

- The task of coming up with a sequence of actions that will achieve a goal is called planning. For classical planning we will consider only environments that are fully observable, deterministic, finite, static and discrete. These are called classical planning environments.
- Planning problems can be represented as states, actions and goals.
- Representation of states:- planners decompose the world into logical conditions and represent a state as a conjunction of positive literals.

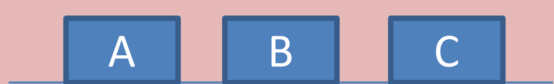
Planning

- Representation of goals:- A goal is a partially specified state, represented as a conjunction of positive literals.
- Representation of actions:- An action is specified in terms of the preconditions that must hold before it can be executed and the effects that ensure when it is executed.
- A solution for a planning problem is an action sequence that when executed in the initial state, result in a state that satisfies the goal.

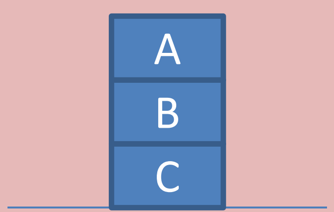
Planning

<u>Example: The blocks world</u>

- The blocks world consists of a set of cube shaped blocks sitting on a table.
- A robot arm can pick up only one block at a time and move it to another position , either on the table or on top of another block.
- Suppose the goal is to get block A on B and block B on C .



Initial state



Final/goal state

Example: The blocks world

- On(b,x) → Block b is on x, where x is either another block or the table.
- Clear(x) :- nothing on top of x.
- Move(b,x,y) → Moving block b from the top of x to the top of y.
  - The precondition for this action is no other block be on top of b or y. In STRIPS we can write :
  - Action(Move(b,x,y)

    PRECOND: On(b,x) ∩clear(b)∩clear(y)

    EFFECT: On(b,y) ∩clear(x) ∩~On(b,x) ∩~ Clear(y))

Example: The blocks world

So to build a three-block tower one solution is:-

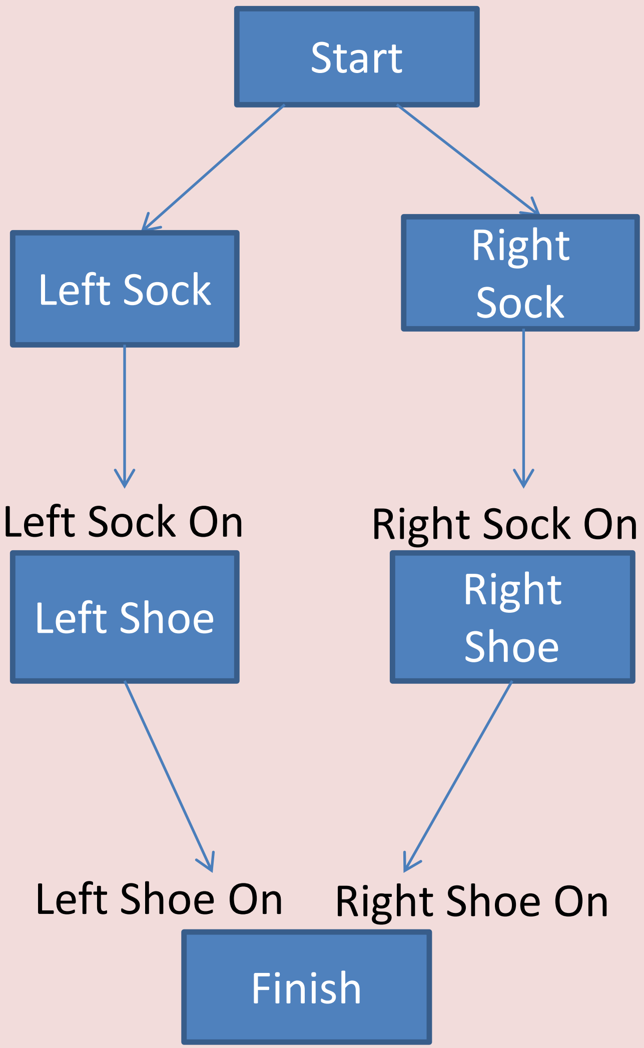
- Init (On(A,Table) ∩ On(B,Table) ∩ On(C,Table) ∩Block(A) ∩Block(B) ∩Block(C) ∩ clear(A) ∩clear(B) ∩clear(C))
- Goal(On(A, B) ∩ On(B,C))
- Action(Move(b, x, y))
- Then we can formulate the solution as the sequence

  [Move (B, Table, C), Move(A, Table, B)]
- The planning of the above type are called ***totally ordered planning***. It has only strict linear sequences of actions connecting the start state to the goal state. We can't decompose the problem into sub-problems.

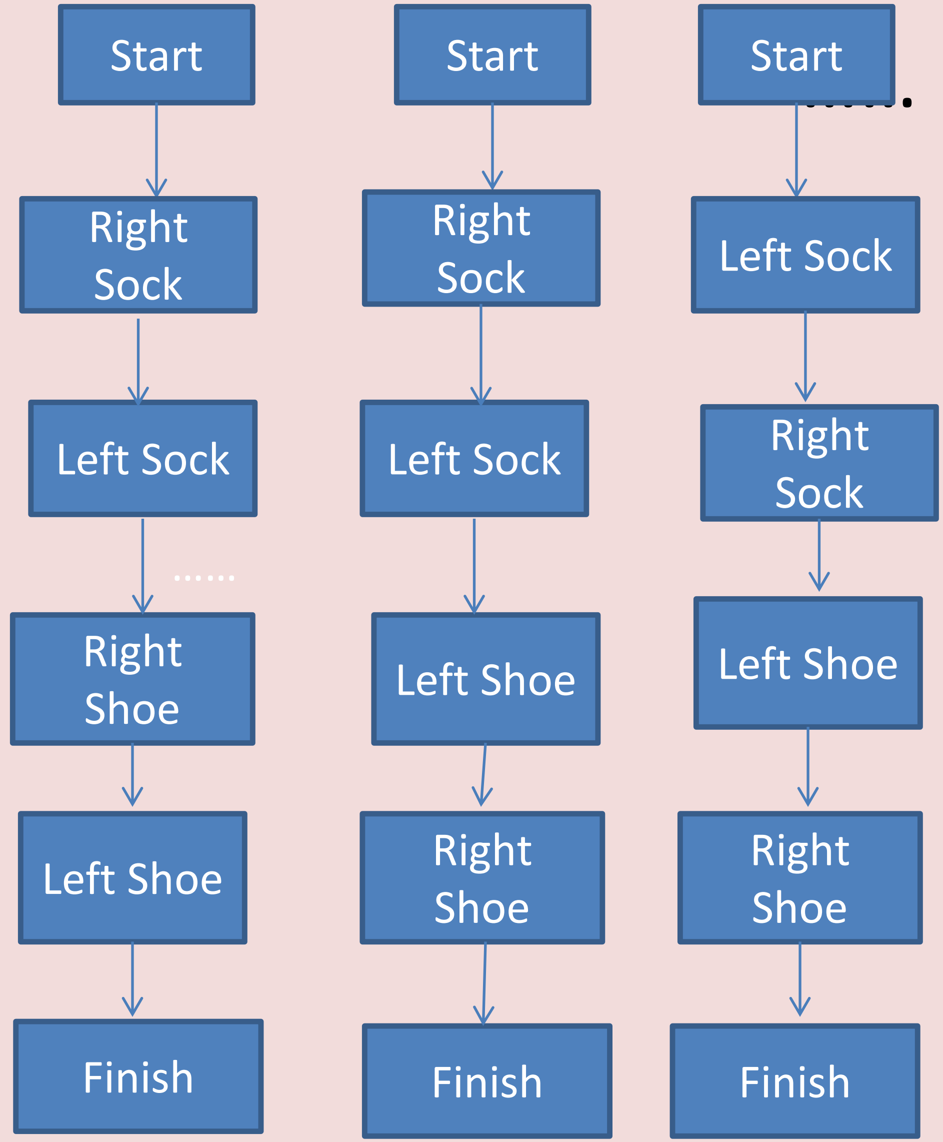
Partially ordered planning

- Consider the simple problem of putting on a pair of shoes.
- Goal(RightShoesOn ∩ LeftshoeOn) ; Init()
- Action(RighSock, EFFECT: RightSockOn)
- Action(LeftSock, EFFECT: LeftSock On)
- Action(RightShoe, PRECOND: RightSockOn, EFFECT: RightShoeOn)
- Action(LeftShoe, PRECOND: LeftSockOn, EFFECT: LeftShoeOn)
- A planner can plan 2 action sequences : [Rightsock, RightShoe] and [LeftSock, LeftShoe]. Then the two sequences can be combined to yield the final plan.
- Any planning algorithm that can place two actions into a plan without specifying which comes first is called a partial order planning.

Partial order plan



Total order plans



- In this example, the partial order solution corresponds to six possible total order plans; each of these is called a linearization of the partial order plan.