Planning

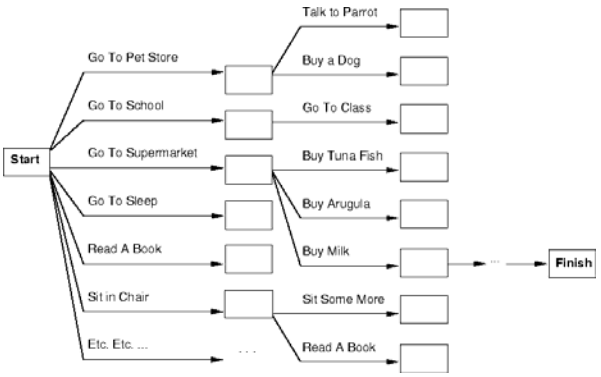
- What is planning?
- Strategies required to achieve a goal or to solve a particular problem

Need for Planning

- If our goal is to create an autonomous, intelligent entity, we need to study planning.
- Logic, natural language, learning, vision, are all useful, and they would make nice human-driven tools.
- But what is the entity going to DO???
- It has to decide WHAT it wants to do, and HOW to do it.

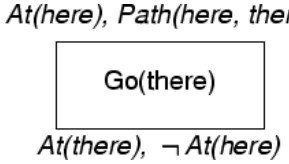
Planning Search Space

- Suppose the problem is “Get a quart of milk and two eggs (to make pancakes) and a variable-speed cordless drill (to fix house when done)”.
- We search through a space of states
- We try each possible action
- Thousands of actions, millions of states
- The heuristic function can direct search, but doesn't prune states



Planning Problem Representation

- STRIPS Operators
- **Precondition:**
  - Conjunction of atoms (positive literals) that must be true to apply operator
- **Effect:**
  - Conjunction of literals that describes how the situation changes when operator is applied
- **Example**
  - OP Go(*there*)
    Precondition: At(*here*) ∧ Path(*here*, *there*)
    Effect: At(*there*) ∧ ¬At(*here*)



State Space: Plan Representation

- Two ways to represent the planning problem:
  - State Space
  - Plan Space
- States are represented as a conjunction of instantiated literals (no functions)

  At(Home) ∧ ¬Have(Milk) ∧ ¬Have(Bananas) ∧ ¬Have(Drill) ∧ …

  - States can be incomplete – if not specified, then negation is assumed
    If Have(Unicorn) is not included, assume ¬Have(Unicorn)
  - This is the Closed-World Assumption
- Goals are described as conjunction of literals (possibly with variables)

  At(Home) ∧ Have(Milk) ∧ Have(Bananas) ∧ Have(Drill)

  or

  At(x) ∧ Sells(x, Milk) if goal is to be at store that sells milk
- Variables must be existential (like theorem prover goal)
- Here, we are not seeing if database entails goal
- We are trying to transform state to one that includes goal

State Space Planner

- Operators can have variables - then we unify goals with facts.
- We call this type of planner a “situation space” planner, or “state space” planner, because nodes in search space represent states or situations.
- This type of planner completely solves one goal, then tacks on plan for next goal.
- “Progression planner” if search forward (A*) This prunes options if high fan-in
- “Regression planner” if search backward (GPS) This prunes options if high fan-out

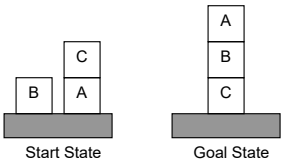
State Space Planner

- General Problem Solver, GPS
- Uses Means-Ends Analysis
  - If *goals* $\subseteq$ *initial-state* then return *True*
  - Choose a difference *d* $\in$ *goals* between *initial-state* and *goals*
  - Choose an operator *o* to reduce the difference *d*
  - If no more operators, then return *False*
  - *State*=GPS*(initial-state*, preconditions(*o*))
  - If *State*, then return GPS(apply(*o,initial-state*),*goals*)

Limitations

- Notice what happens when we apply STRIPS planning to the following problem:

Initial State: On(C,A) ∧ On(A,TABLE) ∧ On(B, TABLE) ∧ CLEAR(C) ∧ CLEAR(B)
Goal State: ON(A,B) ∧ ON(B,C)
Operator: PUTON(x, y)
Precondition: Clear(x) ∧ (y=TABLE ∨ Clear(y))
Add: On(x, y)
Delete: On(x, ~), y!=TABLE -> Clear(y)



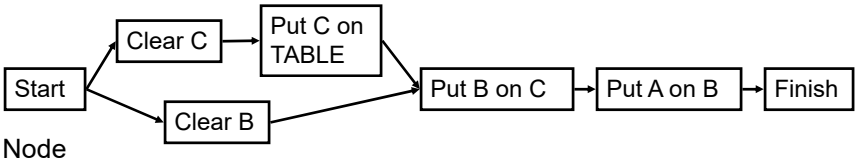
- In which order do we try to achieve the goals?
  If 1) ON(A,B)
  Try: PUTON(C,TABLE) to clear A
  PUTON(A,B) to achieve first goal
  - To achieve this goal, B will be re-cleared, undoing the first goal.

  If 2) ON(B,C)
  Try: PUTON(B,C) to achieve first goal
  - To achieve this goal, B and C would be removed, undoing this goal.

Sussman's Anomaly

- This problem can be solved, but it cannot be attacked by first applying all the operators to achieve one goal, and then applying operators to achieve another goal.
- The problem is that we have forced an ORDERING on the operators. Sometimes steps for multiple goals need to be interleaved.
- Nonlinear planning is a type of plan generation in which ordering is imposed on operators ONLY when it has to be imposed in order to achieve the goals.

Plan Space Planning



- Node
  - state space: world state
  - plan space: partial plan
- Operator
  - state space: next step in plan sequence
  - plan space
    - Add a link from existing action to open condition (precondition that is not yet fulfilled)
    - Add a plan step to fulfill open condition
    - Order a step in the sequence
- Gradually move from incomplete/vague plans to complete plans
- We call planners that use this type of approach partial-order planners (POP)

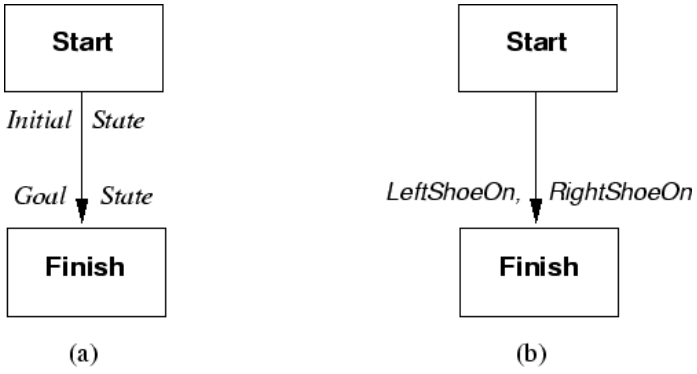
Pan Space Planning

- Instead of search through states from initial state to goal state, make “obvious” or “important” decisions first

  If goal includes Have(Milk)
  and Buy(*x*) achieves Have(*x*)
  then agent should consider plan that includes Buy(Milk)

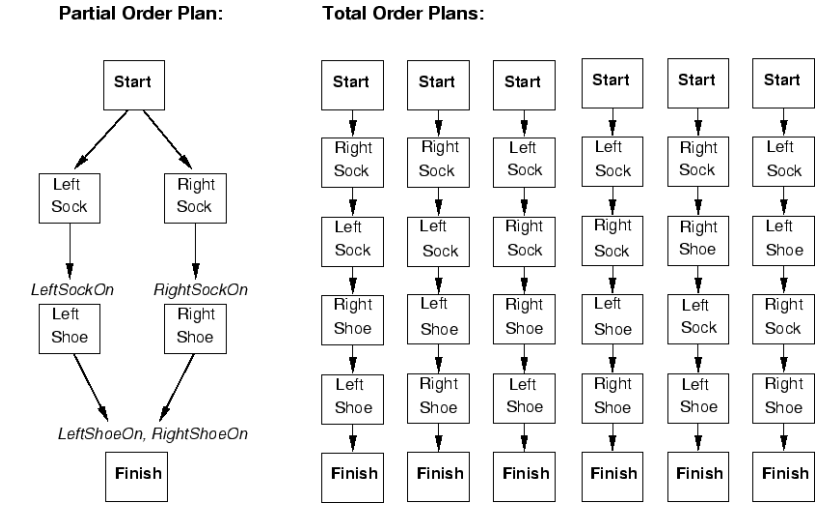
- The planner can add actions to the plan wherever they are needed, instead of always at the end (or beginning) of the plan
  - We can solve one part of conjunctive goal with one plan, another part of the goal with another plan

Partially-Ordered Plans



- The initial plan has only two steps (Start and Finish), 1 ordering (Start before Finish).
- A sock must be put on before a shoe.

Partially-Ordered Plans



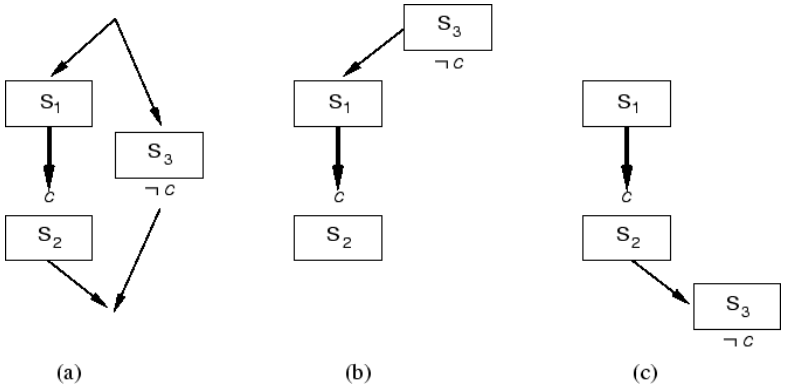
- One partial-order plan can have many linearizations (total orderings), as seen in this example.

Partially-Ordered Plans

- A plan in this scenario consists of the following components:
  1. Set of plan steps
  2. Set of ordering constraints (Step *i* must occur some time before Step *j*)
  3. Variable binding constraints (*v*=*x*)
  4. Causal links ($S_i \rightarrow^c S_j$) which reads "Step *i* achieves condition *c* for Step *j*".
     This is also referred to as a protection interval. If Step *i* achieves condition *c* for Step *j*, no operator better remove *c* during this interval.

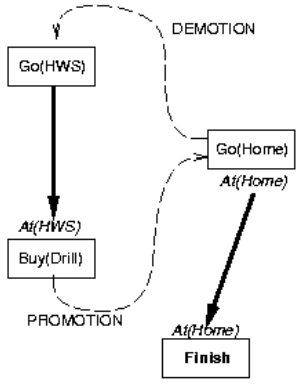
Clobbering and Promotion/Demotion

- A clobberer is a potentially intervening step that destroys the condition achieved by a causal link.
- We can prevent clobbering by ordering the clobberer *before* the start of the link or *after* the end of the link.



Example

- In this example, Go(Home) clobbers At(HWS).



Demotion: put before $Go(HWS)$

Promotion: put after $Buy(Drill)$

UCPOP

- For every open goal
  - Find step that add goal – This forms a link (step $\rightarrow^{goal}$?)
    - This step needs ordering - step < ?
    - Look through delete lists through each step of plan
      - If clobbers protection interval
        - Order clobberer before or after interval.

Sussman's Anomaly Partial Plan

Plan 0:
  Steps: Start(0), Finish(1: precond=(On A B) & (On B C))
  Order: 0 before 1
  Bind: none (Ignore bindings for now)
  Links: none

What is wrong with this plan? Preconds of 1 are not met. Try adding step.

Plan 1: (other sibling would add Puton(B,C))
  Steps: Start{0}, Finish{1: pre=(On A B)-2 & (On B C)}
    Puton(A, B){2: pre=(Clear A) & (Clear B)-0}
  Order: 0 before 1, 2 before 1
  Links: 2 establishes (On A B) for 1 - No step removes (On A B), ok
    0 establishes (Clear B) for 2

The other pre of 1 is not met.

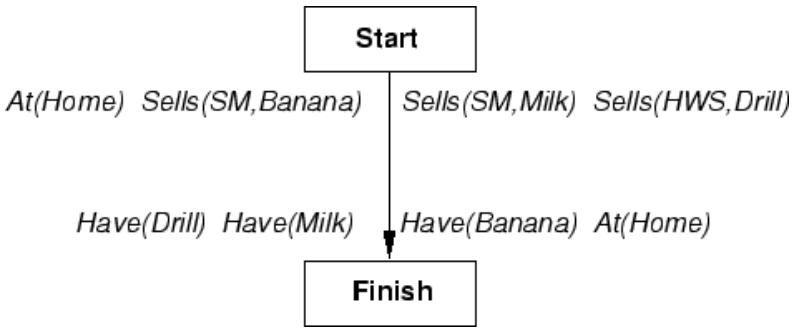
Plan 2: (siblings would address pre of 2)
  Steps: Start{0}, Finish{1: pre=(On A B)-2 & (On B C)-3}
    Puton(A, B){2: pre=(Clear A) & (Clear B)-0}
    Puton(B, C){3: pre=(Clear B)-0 & (Clear C)}
  Order: 0 before 1, 2 before 1, 0 before 2, 0 before 3,
    3 before 2, 3 before 1
  Links: 2 establishes (On A B) for 1, order 2 before 1 -
    No step removes (On A B), ok
    0 establishes (Clear B) for 2, order 0 before 2
    0 establishes (Clear B) for 3, order 0 before 3
    2 clobbers (Clear B), order 3 before 2
    3 establishes (On B C) for 1, order 3 before 1
    No step removes (On B C), ok
0->4->3->2->1

One pre of 2 is not met.

Plan 3:
  Steps: Start{0}, Finish{1: pre=(On A B)-2 & (On B C)-3}
    Puton(A, B){2: pre=(Clear A)-4 & (Clear B)-0}
    Puton(B, C){3: pre=(Clear B)-0 & (Clear C)-0}
    Puton(C, Table){4: pre=(Clear C)-0}
  Order: 0 before 1, 2 before 1, 0 before 2, 0 before 3,
    3 before 2, 3 before 1, 4 before 2, 4 before 3
  Links: 2 establishes (On A B) for 1, order 2 before 1
    No step removes (On A B), ok
    0 establishes (Clear B) for 2, order 0 before 2
    0 establishes (Clear B) for 3, order 0 before 3
    2 clobbers (Clear B), order 3 before 2
    3 establishes (On B C) for 1, order 3 before 1
    No step removes (On B C), ok
    4 establishes (Clear A) for 2, order 4 before 2
    No step removes (Clear A), ok
    3 clobbers (Clear C), order 4 before 3
    0 establishes (Clear C) for 4, order 0 before 4

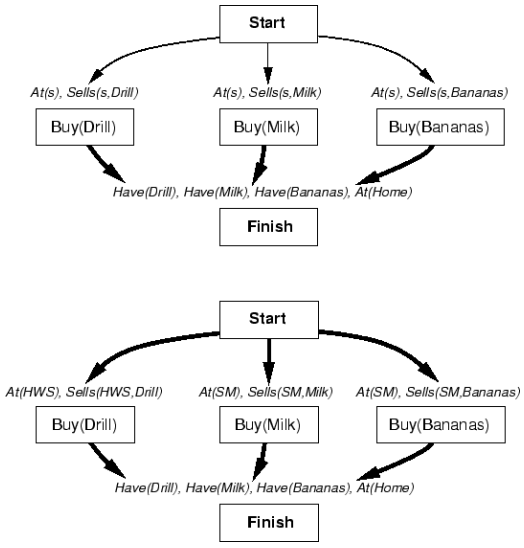
Example

- Here is the shopping problem initial incomplete plan.



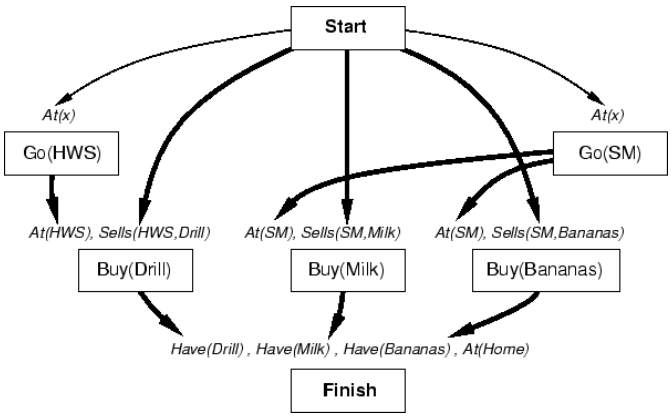
Example

- The top plan achieves three of the four Finish preconditions, the heavy arrows show causal links. The bottom plan refines the top plan by adding causal links to achieve the Sells preconditions of the Buy steps.



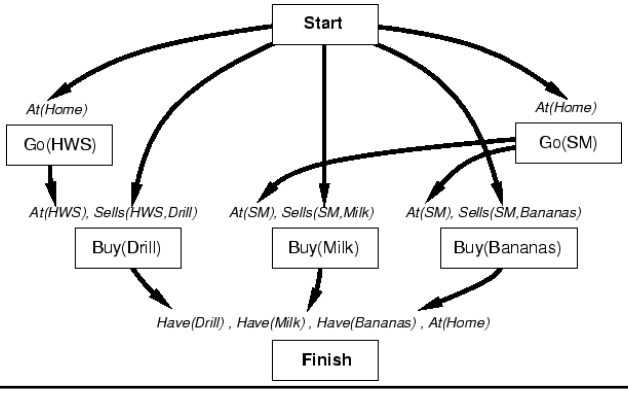
Example

- This partial plan achieves the At preconditions of the three Buy actions.



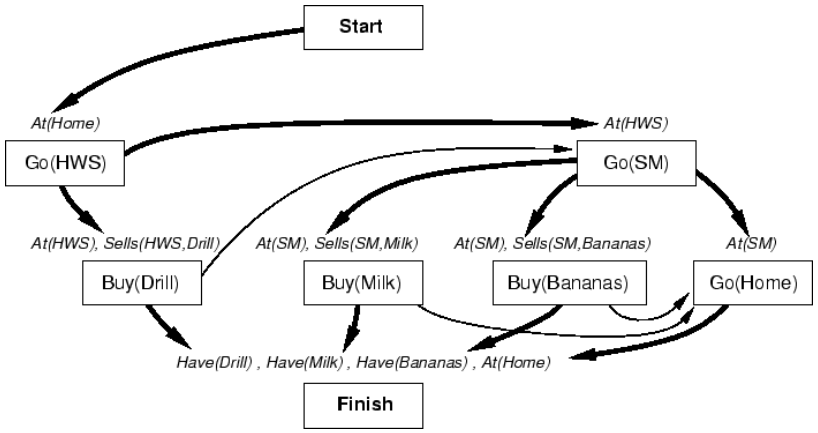
Example

- This is a flawed plan for directing the agent to the hardware store and the supermarket.
- There is no way to resolve the threat that each Go step poses to the other.
- We now backtrack in the search process.



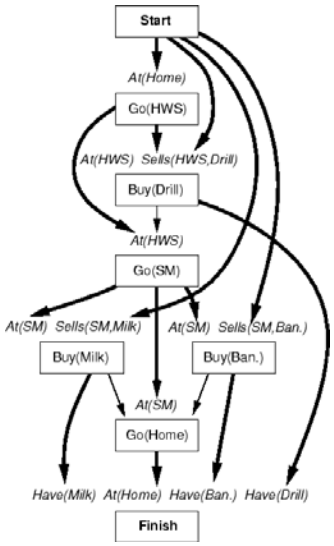
Example

- The next choice is to achieve the At(*x*) precondition of the Go(SM) step by adding a causal link from Go(HWS) to Go(SM).
- The Go(SM) step now threatens the At(HWS) precondition of the Buy(Drill) step, and is resolved by promotion.



Example

- The At(Home) precondition of the Finish step is still unachieved.
- If the planner links At(Home) in the initial state to Finish, there will be no way to resolve the threats raised by Go(HWS) and Go(SM).
- If the planner tries to link At(x) to Go(HWS), there will be no way to resolve the threat posed by Go(SM), which is ordered after Go(HWS).
- The planner links Go(SM) to At(x), so x is bound to SM and Go(Home) deletes the At(SM) condition, resulting in threats to the At(SM) precondition for Buy(Milk) and Buy(Bananas). These threats are resolved by promoting Go(Home).
- Here is the final plan.



Hierarchical Planning

- Reduce complexity of planning by planning in multiple levels
- Example: Attending IJCAI in Acapulco
- Classical Planner:

  Plan to call taxi
  Plan to get to front door
  Plan to get to taxi
  Plan route to airport
  Plan route to ticket counter
  …
  Plan airplane route from Dayton to Acapulco
  …

ABSTRIPS Methodology

- Plan at most abstract level
  Completely ignore predicates labeled as “less critical”
- Using abstract plan, fill in details for next level of abstraction
  Complete preconditions, adding predicates for next lower level of criticality
- Repeat until reached ground (know exactly how to execute it) level
- Issues
  - How form hierarchies?
  - How do levels interact?

Example

- Operator: PushThruDoor(bx, dx, rx)
- Preconditions:
  - (6) Pushable(bx)
  - (6) Isa(dx, Door)
  - (6) Isa(rx, Room)
  - (2) Status(dx, Open)
  - (1) NextTo(bx, dx)
  - (1) NextTo(Robot, bx)
  - (5) Inroom(bx, ry)
  - (6) Inroom(Robot, ry)
  - (6) Connects(dx, ry, rx)
- Similar example with lightbulbs
- STRIPS: 119 nodes, 23 on successful path, 30 minutes CPU
- ABSTRIPS: 60 nodes, 54 on successful path, 5:28 CPU (1/5 time)

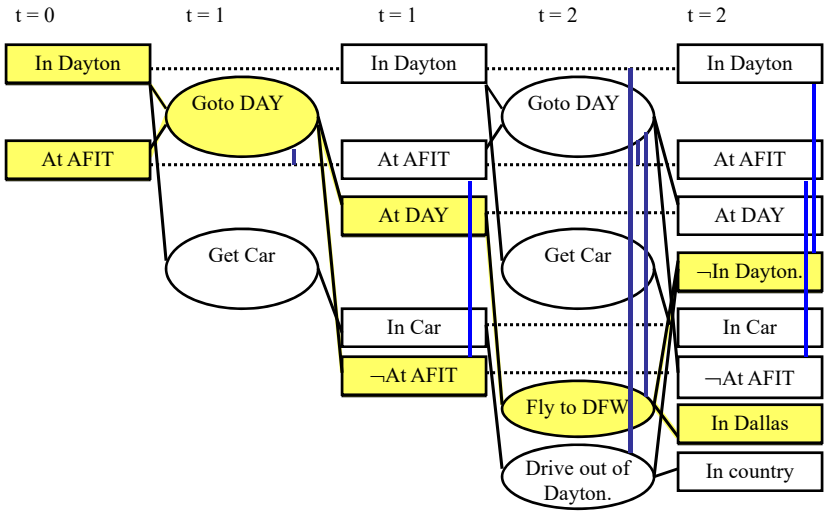
Graphplan Algorithm

- A single plan graph is constructed on which backward chaining search is conducted
- Generating a single shot plan under the closed world assumption
- Forward pass
  - Graphplan iteratively generates a graph layer and if the goals are present:
- Backward pass
  - Searches for a plan
- The plan graph alternates between proposition layer and action layer, representing all reachable propositions from the initial condition
- Termination occurs when all goal propositions are non-mutex in the final layer and reachable from the initial condition.

Graphplan Mutex Constraints

- Mutual exclusions (mutex constraints) assist Graphplan in maintaining state information with the propositional representation.
  - Inconsistent effects: one action negates an effect of the other.
  - Interference: one of the effect of one action is the negation of a precondition of the other.
  - Competing needs: one of the preconditions of one action is mutually exclusive with a precondition of the other

Graphplan Algorithm



HSP, FF

- State-space planning is restricted by the number of searchable plans
- HSP and FF both improve on this by using Graphplan to generate search heuristics
  - The heuristics are general rather than domain specific
- Graphplan is run limiting the action expansion to one action, giving an estimate of a states distance from the initial condition

Planning in Practice

- NASA New Millenium Program
  Programs on board spacecraft perform science planning and scheduling
  Execute plans without human intervention
- Hitachi's O-PLAN
  Process planning and scheduling of 30-day assembly schedules
- Desert Storm Planning
  Plan for transportation of troops, supplies, weapons

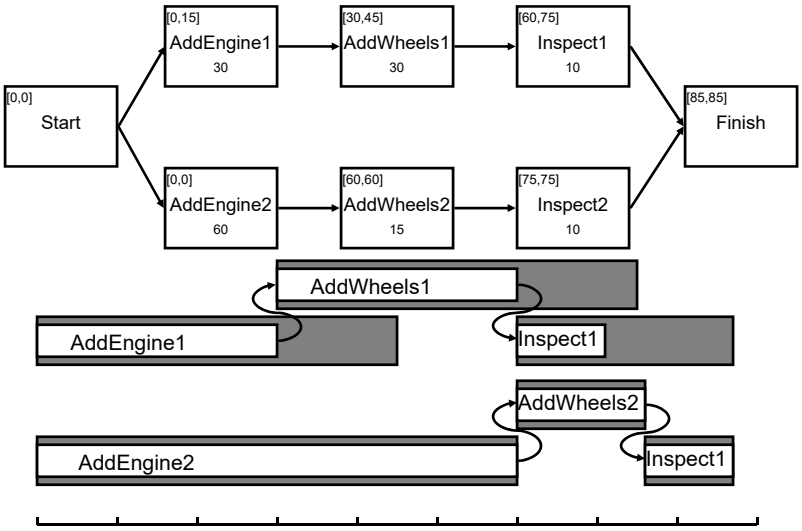
Additional Planning Considerations

- Resource Constraints
- Conditional / Contingent Planning
- Monitoring / Replanning / Sensing actions
- Scalability
- Heuristic Planning
- Decision-theoretic Planning

Scheduling

- Represent planning problem with a duration
- Each partial order plan path has a duration
- Solution must minimize the duration of the critical path
- The other measurement is to minimize the slack time of all plan paths
  - Slack time is the sum of the differences between the latest start time and previous end time.

Scheduling



Scheduling

- In addition to time, resources are also constrained. Resources can be
  - Occupied
  - Reusable
- Resources add additional interactions and complications

Scheduling

Non-Deterministic Planning

- Conformant plan a sequential plan with no sensing actions that ensures that the plan achieves the goal under all possible circumstances.
- Conditional plan/Contingency plan bounds indeterminacy by constructing a conditional plan with different branches for the different contingencies that arise
- Execution monitoring and replanning/Anytime planning the process of the agent maintaining current state information and verifying the plan state matching the current state information
- Continuous plan/policy is a plan that handles any contingency over the lifetime of the system, not just until reaching the goal condition

Conditional Planning

- Action representation
  Op: Suck
  Pre:
  Effect:(When AtL: CleanL) ∧ (When:AtR: CleanR)
- Search is then like a game-tree with the opponent as nature, solving for each branch to the goal condition
- If the domain is partially observable, the state set is then based on the agent's belief state

Multi-Agent Planning

- Cooperative – team based planning
  - Coordination via communication
  - Synchronization for joint actions
  - Search for a joint plan – actions for each agent
- Competitive
  - Game playing agents
  - Any communication of intentions must be questioned – eg. 3 generals

Joint Action Plans

- Joint plans require an agreed upon convention
  - Domain – actions describe concurrency
  - Social laws – domain specific constraints, or execution agreement (first plan found, etc)
  - Swarm – reactive emergent behavior agents
    - Separation – don't get too close to neighbors
    - Cohesion – move toward the center of your group
    - Alignment – steer in the direction of neighbors

Review

- Planning
  - POP
  - Graphplan
  - Resource Constraints
  - Non-deterministic domains
- Next Class:
  - Uncertainty Reasoning