State-Space Planning (Recap)

- Planning as state-space search
  - Forward (progression) from initial state
  - Backward (regression) from goal

- Action is fully specified at time of introduction to plan
  - All variables bound to specific objects
  - Ordering in sequence with existing actions
GraphPlan

- Blum & Furst, 1995.
- Radical reformulation of plan search space
  - Applies to propositional encodings
    - Must propositionalize knowledge
  - Compacted state space with plan graph rep’n
  - Demonstrated dramatic performance improvements

Planning Graphs

- Interleave *levels* of literals and actions, related by support, precondition, and mutex

![Planning Graph Diagram]

- Propositions true at time 0
- Propositions true at time t-1
- Actions
- Propositions true at time t
GraphPlan Algorithm

- zeroth level ← initial state
- Repeat until solution or “level off”:
  - Extend planning graph by considering application of actions
  - Attempt to extract solution

Example: “Dinner Date”

- Object: Take out garbage, fix dinner, wrap present.
- Goal: \( \neg \text{garb} \land \text{dinner} \land \text{present} \)
- Actions: \( (\text{act [preconds] [effects]}) \)
  - cook [cleanH] [dinner]
  - wrap [quiet] [present]
  - carry [] [\( \neg \text{garb} \neg \text{cleanH} \)]
  - dolly [] [\( \neg \text{garb} \neg \text{quiet} \)]
- Initial: \( \text{garb} \land \text{cleanH} \land \text{quiet} \)

from Weld, “Recent advances in AI planning”, AI Magazine, 1999.
Constructing the Plan Graph

Mutex
- Actions
  - Inconsistent effects
    - Effects conflict
  - Interference
    - Effects conflict with Preconditions
  - Competing needs
    - Preconditions conflict
- Propositions
  - Complements
  - Inconsistent support
    - Actions that generate propositions are mutex

Solution?
- Can achieve ~garb, dinner, present separately
- Two “solns”:
  - {carry, cook, wrap}
  - {dolly, cook, wrap}
- But:
  - mutex(carry, cook)
  - mutex(dolly, wrap)
Extend the Plan Graph

Solution!
Your turn

Scoreboard

X

O

1 2 3

4 5 6

7 8 9
The Grand Plan

- In principle, could anticipate all possible contingencies, and produce one conditional plan, once and for all
  - The GRAND PLAN
  - Universal planning
- Why not?

Planning under Uncertainty

- Whenever agent cannot *perfectly* predict result of executing plans
- Uncertainty may be about:
  - Initial state
  - Effects of actions
  - Exogenous events
Approaches

- Sensorless (Conformant) Planning
- Conditional Planning
- Execution Monitoring and Replanning
- Continuous Planning
- Explicit Uncertainty Modeling [next time]

Sensorless (Conformant) Planning

Coerce world to known state
Limitations of Belief State

- Suppose we have N propositions
- How many possible states?
- How many possible belief states?
  - Is there a better way?
- Suppose we only record value of propositions that are constant within belief state (the rest are “unknown”)
  - How many possible states?

Uncertainty Resolution

- Uncertainty at plan construction time may be resolved by execution time
  - Example: shopping plan, don’t know how much milk costs
  - Problem: cannot specify complete guaranteed plan without specifying amount of money for Pay action
  - Solution: find out the price when we get to Busch’s
Conditional Planning

- Identify conditions that will be observable at execution time
- Include construct in plan that selects course of action based on condition:
  
  ```
  If (Has(Mary, Keys))
  Unlock(Door)
  else
  Break(Window)
  ```

- May need to introduce information-gathering actions
  - Door might be unlocked already, but don’t know that in advance!

Information-Gathering Action

- **CheckOnSale** (x, y)
  
  ```
  Precond: InStore(x)
  Effect: KnowsWhether(“OnSale(y)”) 
  ```

- KnowsWhether: eligible for conditioning
- A legal (conditional) plan:

  ```
  Move(John, Store) CheckOnSale(John, Milk)
  If ( OnSale(Milk) )
      Pay(John, Store, $0.99)
  else
      Pay(John, Store, $1.49)
  ```
Putting off Contingencies

- **Alternative:**
  - Construct plan for expected case
  - If it doesn’t work out, *then* figure out what to do

- **Assumes**
  - We can detect when things go wrong (*execution monitoring*)
  - We will have time (and knowledge) to deal with it (*replanning*)

---

Online replanning
Execution Monitoring

- Identify in advance what conditions are assumed by baseline plan
  - Constructs like causal links can help
- Monitor to determine whether they hold at execution time
  - Plan explicit sensing (information-gathering) actions if necessary
- On detection, invoke replanning

Action Monitoring

- Action monitoring: before attempting an action, check to make sure its preconditions really are true.

- Spare Tire Example:
  - After I take off the flat tire and I go to get the spare, my friend sees the (bad, but she thinks good) tire and decides to help by putting it back on.
  - Before I try to jam the spare onto the (now-occupied) hub, check that the hub is really clear
Plan Monitoring

- Plan monitoring: before attempting next action, check to make sure all causal links are still established.

- Spare Tire Example: When do I rethink things if it turns out after the first action that the spare has become deflated?

Real-time Replanning
Replanning

- Find new plan from situation agent finds itself in
- Try to use (some of) existing plan if possible
- Interesting issues of serendipity....

Continual Planning

- In reality, we are always part of the way through executing our “grand plan” (living our life)
- Continual planning agent
  - Interleaves planning and execution
    - Never really has a complete plan
  - Updates current state based on percepts
  - May receive new goals, experience positive and negative world changes
Continual Replanning

Real Time

- It matters *when* things happen, not just *what* happens
- All realistic problems exist in real time, with varying degrees of time stress
- In many domains, time constraints are what makes the problem hard.
Multi-Agent Planning

- Now, changes to environment are not strictly caused by the agent
- Also, possible that effects of actions change depending on concurrent actions of other agents
- Some possibilities:
  - Compete: Plan to achieve goals despite others’ actions, where “chance” nodes really aren’t all chance (we saw this in game search)
  - Coexist: Coordinate plans with other agent’s to avoid conflicting or redundant activities
  - Cooperate: Plan as a team to achieve goals that no agent can achieve alone

Next Time

- Explicitly modeling uncertainty
  - Probability!