Overview

Why Plan

Modeling the World

Strips

Reasoning about Sequences of Actions

Planning

Actions and Planning

We will assume that Agent is the only one changing the world.

How does the world change over time?

So, need to reason about actions and how they affect the world.

But, often times it will take a sequence of actions to achieve goals.

After all, we want to build agents that can act proactively.

Agents might also want to change the world.

We can reason about it.

So far, we have considered how to represent the world.
### Static Relations

- **robot(Ag):** true if Ag is a robot
- **opens(Key,Door):** true if Key opens Door
- **nodoorbetween(Pos1,Pos2):** true if position Pos1 is adjacent to position Pos2 and there is no door between them
- **between(Door,Pos1,Pos2):** true if Door is between position Pos1 and position Pos2

This is a slightly different conceptualization of the world than in the textbook. Textbook version has adjacent as being a static & derived relation.

### Modeling the Delivery Robot World

- **Individuals:** rooms, doors, keys, parcels, and the robot
- **Relations:**
  - Robot's position
  - Position of packages, keys, locked doors
  - What robot is holding

### Actions:

- Move around
- Pick up and put down keys and packages
- Unlock doors (with the appropriate keys)
- + Push and pull doors
- + Door opening

### Why is it useful to distinguish?

- Static relations are always the same: no need to recompute them at each time point.
- Derived relations need to be re-derived at each time point (usually from primitives).
- Only need to reason about how the primitive relations change over time.
- Derived need to be re-computed at each time point (usually from primitives).
- Static always the same: no need to recompute them at each time point.
- Why is it useful to distinguish?

### Types of Relations

- **Static relations:**
  - Truth value does not change.
- **Dynamic relations:**
  - Action can change truth values.
  - Dynamic relations can be further classified as:
    - **Primitive relations:** truth value can be determined by considering its value in the past and what actions have been performed.
    - **Derived relations:** truth value can be derived from other relations.
  - Dynamic relations are useful for reasoning about changes in the world.
  - Two basic types of relations: static and dynamic.

The above diagram illustrates the concept of robot positions and actions in a delivery world.
Actions

- **move(Ag,From,To)**: agent Ag moves from location From to adjacent location To. The agent must be sitting at location From.
- **pickup(Ag,Obj)**: agent Ag picks up Obj. The agent must be at the location that Obj is sitting.
- **putdown(Ag,Obj)**: the agent Ag puts down Obj. It must be holding Obj.
- **unlock(Ag,Door)**: agent Ag unlocks Door. It must be outside the door and carrying the key to the door.

Derived Relations

- **adjacent(Pos1,Pos2)**: true if the robot can move from Pos1 to Pos2 in one step. So, Pos1 and Pos2 must have no door between them or have an unlocked door between them.

- **at(Obj,Loc)**: true in a situation if Obj is at Loc. How at(Obj,Loc) is derived depends on if object is being carried or not.
  - at(Obj,Loc) ← sitting at(Obj,Loc)
  - at(Obj,Loc) ← carrying(Agt,Obj) ∧ sitting at(Agt,Loc)

- **primitive Relations**

  - **unlocked(Door)**: true in a situation if Door is unlocked.
  - **sitting at(Obj,Loc)**: true in a situation if Obj is sitting on the ground (not being carried) at location Loc in the situation. Also used to describe robot's location, where Obj is the robot.
  - **carrying(Ag,Obj)**: true in a situation if agent Ag is carrying Obj in that situation. Either carrying or sitting at will be true for any object (including the robot).
  - **Door (pos1,pos2)**: true if the robot can move from Pos1 to Pos2.
How do we reason about sequences of actions?

• Before determining how we plan, let's focus on just being able to verify a plan accomplished the goal.

Goal: $\text{carrying}(\text{rob}, \text{k1}) \land \text{at}(\text{rob}, \text{o103})$

• How do we know that this plan accomplished the goal?

move(\text{rob}, \text{o109}, \text{o103})

Overview

• Why Plan
• Modeling the World
  ⇒ Reasoning about Sequences of Actions
• Strips
• Planning

Initial World

• Static Facts
  
  $\text{between} (\text{door1}, \text{o103}, \text{lab2}).$
  $\text{opens} (\text{k1}, \text{door1}).$
  $\text{robot} (\text{rob}).$
  $\text{nodoorbetween} (\text{o109}, \text{o103}).$
  $\text{nodoorbetween} (\text{o103}, \text{o109}).$
  ...
  $\text{nodoorbetween} (\text{lab2}, \text{o109}).$

• Primative Relations of Initial World
  
  $\text{sitting} (\text{man}, \text{lab2}).$
  $\text{sitting} (\text{prof}, \text{lab2}).$
  $\text{sitting} (\text{prof}, \text{lab2}).$

• Why Plan

Planning:

Steps:

How do we reason about sequences of actions?

Modelling the World:

Initial World:
Planning
- Strips

Reasoning about Sequences of Actions

Modelling the World

Why Plan

Overview

• Why Plan
• Modeling the World
• Reasoning about Sequences of Actions
⇒ Strips

Continued
• Actions: move, pickup, unlock
- They are not predicates, that are true or false
- They are things that transform the current world
+ Need to specify if they can be applied to the current world
+ Need to specify how the current world is transformed

Strategy 1: Strips
- View actions are outside of the logic
+ View domain is having world objects in it that can be referred to

Strategy 2: Situation Calculus
- View actions are inside of the logic
+ There are things that transform the current world
+ They are not predicates; they are facts or rules

Key Insight

• Actions: move, pickup, unlock
- First one is the initial world
- A sequence of actions induces a sequence of worlds
- A sequence of actions is outside of the logic
+ There are things that transform the current world
+ They are not predicates; they are facts or rules
+ They are not predicates; they are facts or rules

Actions: move, pickup, unlock
STRIPS Representation of "pickup"

• The action pickup(Ag,Obj) can be defined by:
  - preconditions
    - robot(Ag), Ag ≠ Obj, at(Ag,Pos), sitting at(Obj,Pos)
  - delete list
    - sitting at(Obj,Pos)
  - add list
    - carrying(Ag,Obj)

• move(Ag,Pos1,Pos2)
  - preconditions
    - robot(Ag), adjacent(Pos1,Pos2), at(Ag,Pos1)
  - delete list
    - sitting at(Obj,Pos1)
  - add list
    - sitting at(Ag,Pos2)

• putdown(Ag,Obj)

• unlock(Ag,Door):

Atoms in preconditions might have variables in them.

STRIPS and Theorem Proving

• STRIPS Representation:
  - The actions are external to the logic
  - Applying an action to a world creates a new world
  - Each action has:
    - preconditions: a list of atoms that need to be true by the action
    - delete list: a list of primitive relations no longer true by the action
    - add list: a list of primitive relations made true by the action
  - The actions are executed in the order:
    - add list, delete list, preconditions
  - STRIPS assumption:Primitive relations not mentioned in the description of the action stay unchanged
Planning

- Given:
  - an initial world description
  - a description of available actions
  - a goal
  - in initial world description

A plan is a sequence of actions that will achieve the goal.

Overview

- Why Plan
- Modeling the World
- Reasoning about Sequences of Actions
- Strips

Example Transitions

Initial State

- between(door1,o103,lab2).
- opens(k1,door1).
- autonomous(rob).
- nodoorbetween(o109,o103).
- nodoorbetween(o103,o109).
- nodoorbetween(lab2,o109).
- sitting at(rob,o109).
- sitting at(parcel,storage).
- sitting at(k1,mail).

- After move(rob,o109,storage)?
- After pickup(rob,parcel)?
STRIPS Forward Planner

• Put pair of <initial plan, initial world> in frontier

  - Initial world could just include primitives (not static nor derived)
  - Initial plan is just an empty list

• Loop: while frontier is not empty

  - Take out top path/world from frontier

  - If goal true in top world then stop

  - Figure out all possible actions using precondition lists

  + This will tell you the neighboring worlds

  - Plot failures of all possible actions using precondition lists

  - Don't use in top world then stop

  - Take one step forward from frontier

  - If folder is not empty, is not empty

  - Insert pair of <initial plan, initial world> in frontier

What would A∗ use as g and h?

Forward Planner

- The goal doesn't uniquely specify a state.
- You usually can do backwards planning in the state space as
  - You usually must do backwards planning in the state space, as
  - the number of actions to achieve goal
  - Branching factor is the number of legal actions. Path length is
  - Guaranteed to find a solution
  - A complete search strategy (e.g., A∗ or iterative deepening) is
  - Search from initial state to a state that satisfies the goal
  - Search from initial state to a state that satisfies the goal

- Nodes will have a complete listing of what is true in world
- Search in the state-space graph, where the nodes represent
  - World and the arcs represent actions
  - Search in the state-space graph, where the nodes represent

Example Planning

• If you want a plan to achieve Rob holding the key and being

  - You usually must do backwards planning in the state space as
  - the number of actions to achieve goal
  - Branching factor is the number of legal actions. Path length is
  - Guaranteed to find a solution
  - A complete search strategy (e.g., A∗ or iterative deepening) is
  - Search from initial state to a state that satisfies the goal

  - You usually must do backwards planning in the state space as
  - the number of actions to achieve goal
  - Branching factor is the number of legal actions. Path length is
  - Guaranteed to find a solution
  - A complete search strategy (e.g., A∗ or iterative deepening) is
  - Search from initial state to a state that satisfies the goal

- Nodes will have a complete listing of what is true in world
- Search in the state-space graph, where the nodes represent

What strategy should you use to find a solution?