Planning

Informatics Engineering Study Program
School of Electrical Engineering and Informatics

Institute of Technology Bandung

Contents

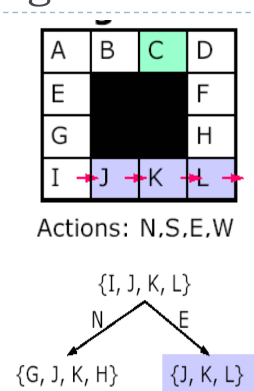
- Review
- Planning vs problem solving
- Situation calculus
- STRIPS representation
- Plan-space planning
- Partially Ordered Plan

Review

- Problem Solving agent
- Knowledge Based agent
- Learning agent
- Probabilistic Reasoning System
- Planning agent → TODAY

Planning as problem solving

- Planning:
  - Start state (S)
  - Goal state (G)
  - Set of actions
- Can be cast as "problem-solving" problem
- But, what if initial state is not known exactly? start in bottom row in 4x4 world, with goal being C.
- Do search over "sets" of underlying (atomic) states.



Planning as Logic

- The problem solving formulation in terms of sets of atomic states is incredibly inefficient because of the exponential blowup in the number of sets of atomic states.
- Logic provides us with a way of describing sets of states.
- Can we formulate the planning problem using logical descriptions of the relevant sets of states?
- This is a classic approach to planning: situation calculus, use the mechanism of FOL to do planning.
- Describe states and actions in FOL and use theorem proving to find a plan.

Situation Calculus

- Reify situations: [reify = name, treat them as objects] and use them as predicate arguments.
  - At(Robot, Room6, S9) where S9 refers to a particular situation
- Result function: a function that describes the new situation resulting from taking an action in another situation.
  - Result(MoveNorth, S1) = S6
- Effect Axioms: what is the effect of taking an action in the world
  - For All x.s. Present(x,s) ^ Portable(x) → Holding(x, Result(Grab, s))
  - For All x.s. ¬ Holding(x, Result(Drop, s))
- Frame Axioms: what doesn't change
  - For All x.s. color(x,s) = color(x, Result(Grab, s))
  - Can be included in Effect axioms

Planning in Situation Calculus

- Use theorem proving to find a plan
- Goal state: For Some s. At(Home, s) ∧ Holding(Gold, s)
- Initial state: At(Home, S_0) ∧ ¬ Holding(Gold, S_0) ∧ Holding(Rope, S_0) …
- Plan: Result(North, Result(Grab, Result(South, S_0)))
  - A situation that satisfies the requirements
  - First, move South, then Grab and then move North.

Special Properties of Planning

- Reducing specific planning problem to general problem of theorem proving is not efficient.
- We will build a more specialized approach that exploits special properties of planning problems.
  - Connect action descriptions and state descriptions [focus searching]
  - Add actions to a plan in any order
  - Sub-problem independence
  - Restrict language for describing goals, states and actions

STRIPS representations

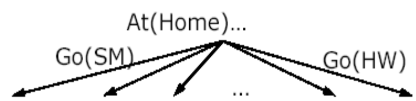
- States: conjunctions of ground literals
  - In(robot, r3) ∧ Closed(door6) ∧ …
- Goals: conjunctions of literals
  - (implicit For Some r) In(Robot, r) ∧ In(Charger, r)
- Actions (operators)
  - Name (implicit For All): Go(here, there)
  - Preconditions: conjunction of literals
    - At(here) ∧ path(here, there)
  - Effects: conjunctions of literals [also known as post-conditions, add-list, delete-list]
    - At(there) ∧ ¬ At(here)
  - Assumes no inference in relating predicates (only equality)

Strips Example

- Action
  - Buy(x, store)
    - Pre: At(store), Sells(store, x)
    - Eff: Have(x)
  - Go(x, y)
    - Pre: At(x)
    - Eff: At(y), ¬At(x)
  - Goal
    - Have(Milk) ∧ Have(Banana) ∧ Have(Drill)
  - Start
    - At(Home) ∧ Sells(SM, Milk) ∧ Sells(SM, Banana) ∧ Sells(HW, Drill)

Planning Algorithms

- Progression planners: consider the effect of all possible actions in a given state.



- Regression planners: to achieve a goal, what must have been true in previous state.

  Have(M) $\wedge$ Have(B) $\wedge$ Have(D)

  Buy(M,store)

  At(store) $\wedge$ Sells(store,M) $\wedge$ Have(B) $\wedge$ Have(D)

- Both have problem of lack of direction – what action or goal to pursue next.

Plan – Space Search

- Situation space – both progressive and regressive planners plan in space of situations
- Plan space – start with null plan and add steps to plan until it achieves the goal
  - Decouples planning order from execution order
  - Least-commitment
    - First think of what actions before thinking about what order to do the actions
  - Means-ends analysis
    - Try to match the available means to the current ends

Partially Ordered Plan (POP)

- Set of steps (instance of an operator)
- Set of ordering constraints Si < Sj
- Set of variable binding constraints v=x
  - v is a variable in a step; x is a constant or another variable
- Set of causal links Si c Sj
  - Step i achieves precondition c for step j

Initial Plan

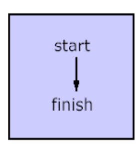
- Steps: {start, finish}
- Ordering: {start < finish}
- start
  - Pre: none
  - Effects: start conditions
- finish
  - Pre: goal conditions
  - Effects: none

Plan Completeness

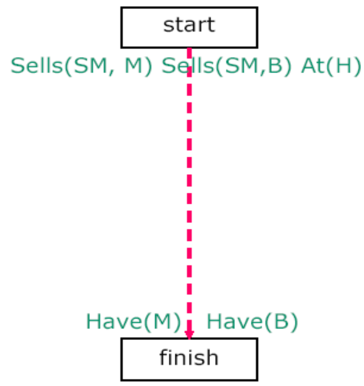
- A plan is complete iff every precondition of every step is achieved by some other step.
- Si $\rightarrow_c$ Sj (“step I achieves c for step j”) iff
  - Si < Sj
  - c Є effects(Si)
  - ¬ ∃ Sk. ¬c Є effects(Sk) and Si < Sk < Sj is consistent with the ordering constraints
- A plan is consistent iff the ordering constraints are consistent and the variable binding constraints are consistent.

Partial Order Planning Algorithms

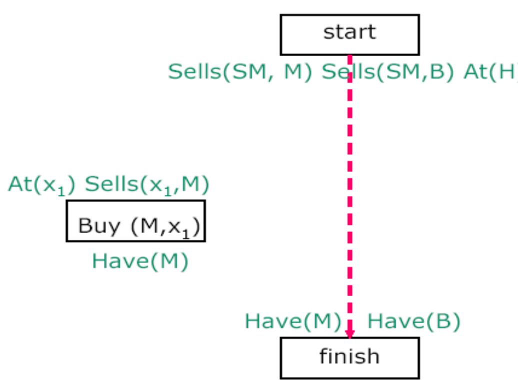
- Plan
  - Steps
  - Ordering Constraints
  - Variable binding constraints
  - Causal links
- POP Algorithm
  - Make Initial Plan
  - Loop until plan is a complete plan
    - Select a subgoal
    - Choose an operator
    - Resolve threats



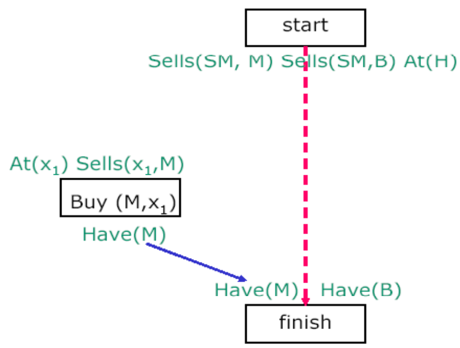
PO Plan Example



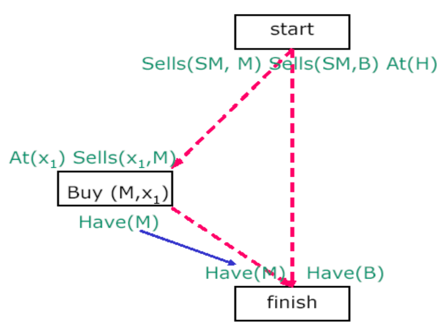
PO Plan Example



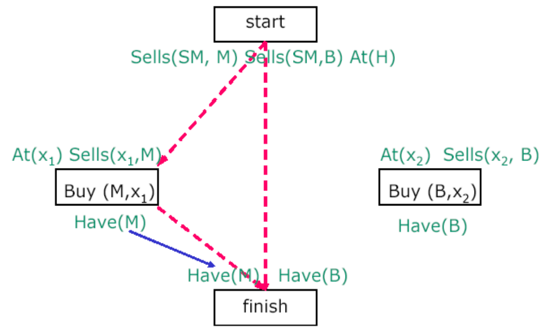
PO Plan Example



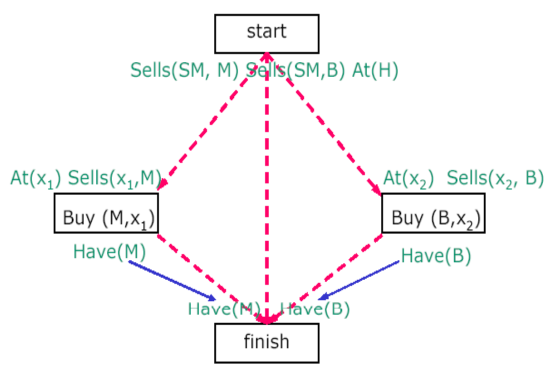
PO Plan Example



PO Plan Example

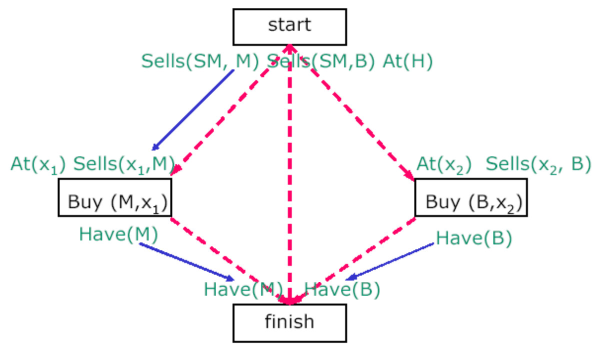


PO Plan Example



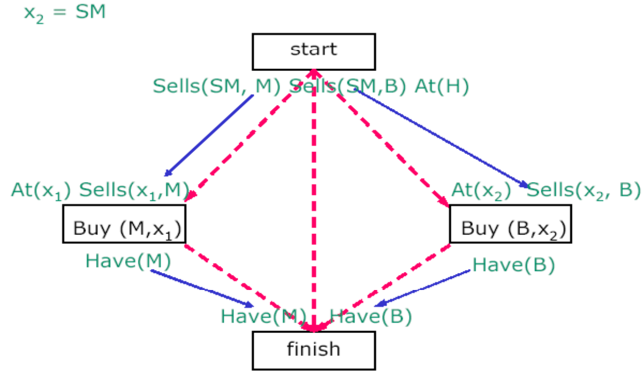
PO Plan Example

x_1 = SM



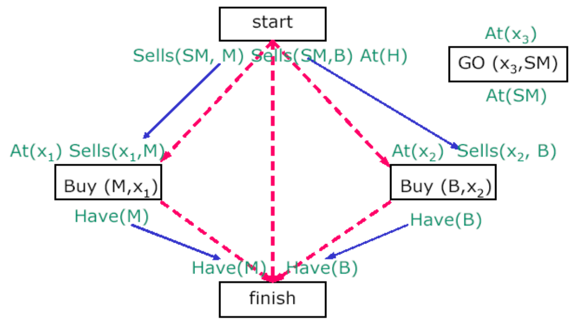
PO Plan Example

x_1 = SM
x_2 = SM



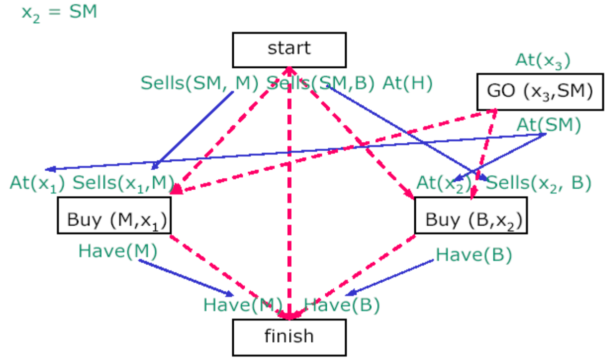
PO Plan Example

x_1 = SM
x_2 = SM

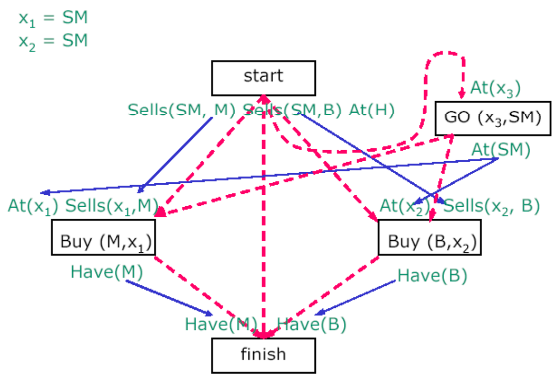


PO Plan Example

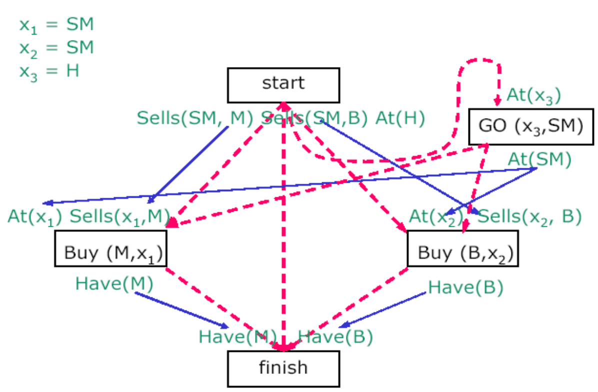
x_1 = SM
x_2 = SM



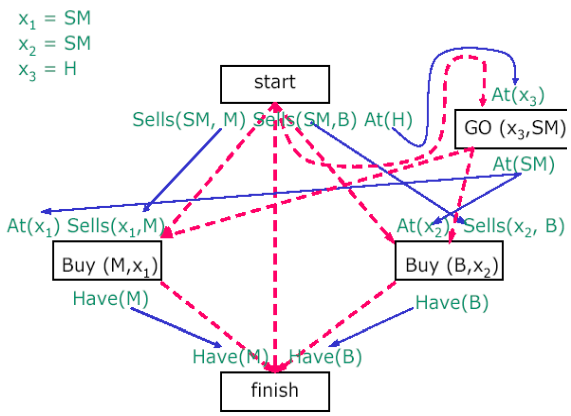
PO Plan Example



PO Plan Example



PO Plan Example



Resolve Threats

- step S threatens a causal link Si →^{c} Sj iff ¬ c Є effects(S) and it's possible that Si < S < Sj
- For each threat
  - Choose
    - Promote S : S < Si < Sj
    - Demote S : Si < Sj < S
  - If resulting plan is inconsistent, then Fail

Review

- Planning agent as Problem Solving Agent
- Special properties of Planning
- STRIPS Representation
- Plan Completeness
- Partial Ordered Plan Algorithm
- Resolve threats

THANK YOU