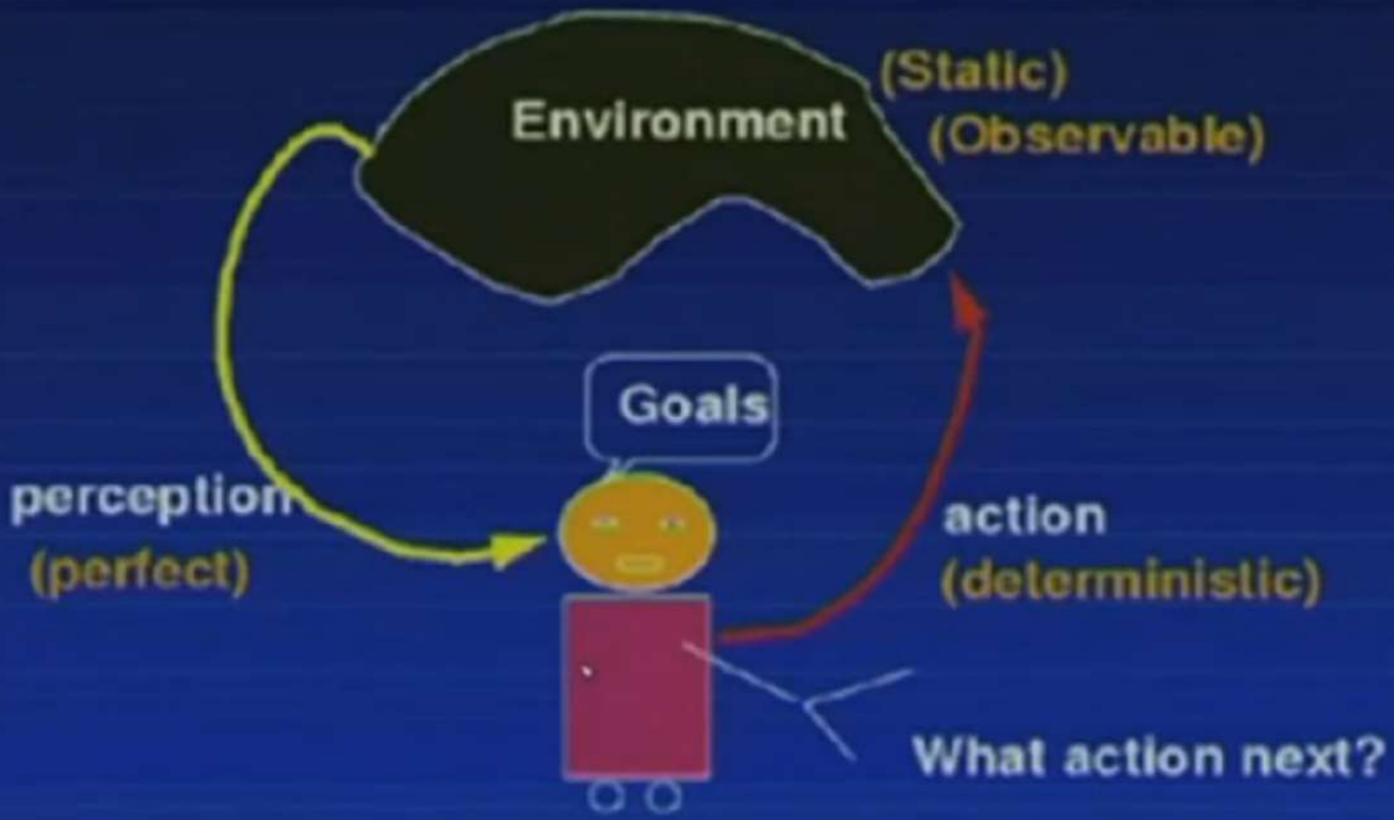
Indian Institute of Technology, Kharagpur



Planning

- Synthesize goal directed behaviour
- Select action sequences
  - Handle causal dependencies
- We will restrict ourselves to deterministic and fully observable situations.

Why Planning

- Intelligent agents must operate in the world.
  - Take intelligent actions
  - Compose actions together to achieve complex goals
- Change the world to suit the needs. Agents need to reason about what the world will be like after executing a sequence of actions.
  - Need to reason about dynamic environment

A planning problem

- Goal: Have a birthday party
- Current situation: Agent is at home, Has flour, Does not have butter, Does not have sugar.
- To Do:

Invite friends, Buy butter, Buy sugar, Buy balloons, Decorate house, Bake cake, …..

Applications

- Mobile robots
- Autonomous agents
  - NASA Deep Space planning agent
- Simulated environments
  - Goal-directed agents for training or games
- Web and grid environments
  - Composing queries or services

Applications

- Scheduling problems with action choices as well as resource handling
  - Hubble Space Telescope scheduler
  - Workflow management
- Software test case generation
- Plan based interfaces

Generating plans

- Given:
  - A way to describe the world
  - An initial state of the world
  - A goal description
  - A set of possible actions to change the world

Indian Institute of Technology, Kharagpur

Planning problems

- Planning algorithms should take advantage of the logical structure of the problem.
- The problem should be expressed in a suitable logical language.

Modeling States

- States are modeled in terms of propositions or state variables.
  - Complete initial state
  - Partial goal state

A state is a conjunction of positive literals

$lighted \wedge hot \wedge madeofbrass$

we can also use first order literals

$At(Robot1, Kitchen) \wedge At(Robot2, Garden)$

Modeling States

We can also use first order literals

At (Robot1, Kitchen) $\wedge$ At (Robot2, Garden)

Literals used must be ground and function-free.

<u>Closed World Assumption</u>:

Any condition not mentioned in a state are assumed false.

Indian Institute of Technology, Kharagpur

Representation of goals

- A goal is a partially specified state

  Ex: Rich $\wedge$ Famous $\wedge$ Stays (Mumbai)

- A state s satisfies goal g

  if s contains all the propositions of g

lighted $\wedge$ hot $\wedge$ brass

satisfies the goal

lighted $\wedge$ hot

Modeling Planning Problems

- States are modeled in terms of propositions.
  - Complete initial state
  - Partial goal state
- Actions modeled as state transformations. Two frameworks:
  - STRIPS
  - ADL

Indian Institute of Technology, Kharagpur

Action Representation

Specification of Actions:

1. Preconditions that must hold before the action can be executed
2. Effects of executing the action

Action: Fly (p, from, to)

Precondition: At (p, from) $\wedge$ Canfly(p) $\wedge$ Airport(from) $\wedge$ Airport(to)

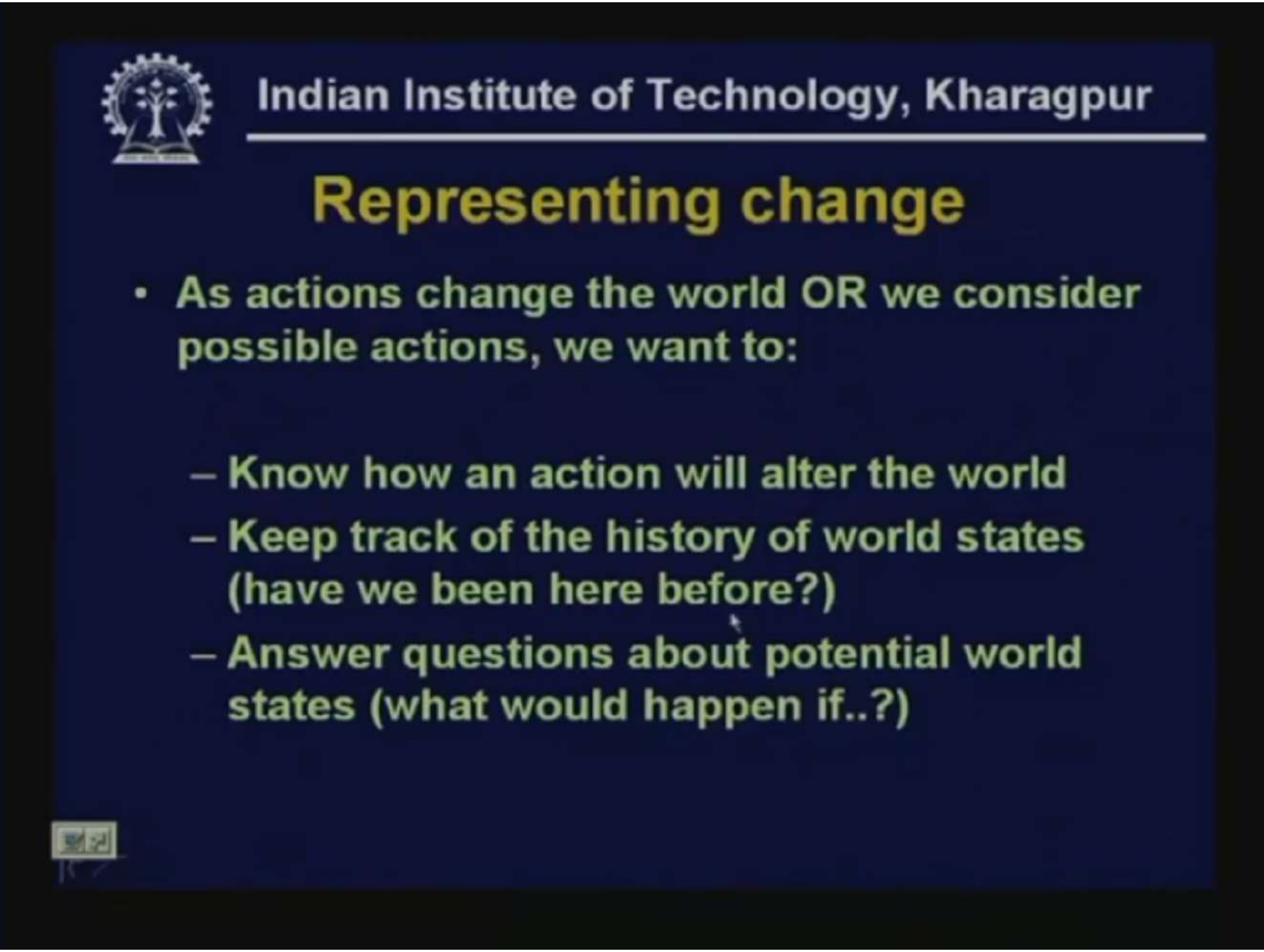
Effect: $\neg$At(p, from) $\wedge$ At(p, to)

Action: Fly (p, from, to)
Precondition: At (p, from) $\wedge$ Canfly(p) $\wedge$ Airport(from) $\wedge$ Airport(to)
Effect: $\neg$At(p, from) $\wedge$ At(p, to)

Action Schema: can represent a number of different actions

Precondition: A conjunction of function-free positive literals. What must be true in a state before an action can be executed.

Effect: A conjunction of function-free positive literals. How the state changes when the action is executed.


Indian Institute of Technology, Kharagpur
Representing change
• As actions change the world OR we consider possible actions, we want to:
– Know how an action will alter the world
– Keep track of the history of world states (have we been here before?)
– Answer questions about potential world states (what would happen if..?)

The situation calculus

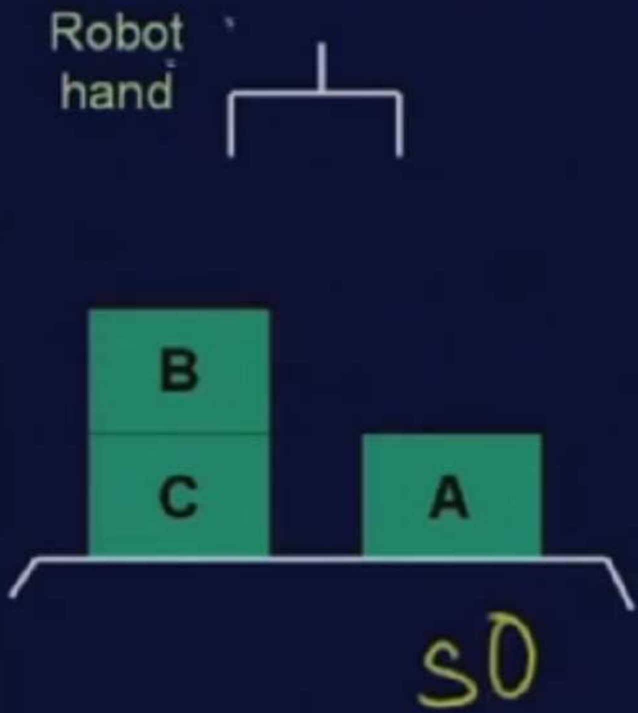
- Situation calculus (McCarthy) : a formalism to model dynamic worlds within first order logic.
- Key idea: represent a snapshot of the world, called a 'situation' explicitly.
- Situations are used to index states of the world.
- Actions map situations to situations.

Fluents

- Properties that change from situation to situation (called fluents) take an extra situation argument.
- Ex: On (B, C, s), Clear (A, s)

Indian Institute of Technology, Kharagpur

Blocks world example



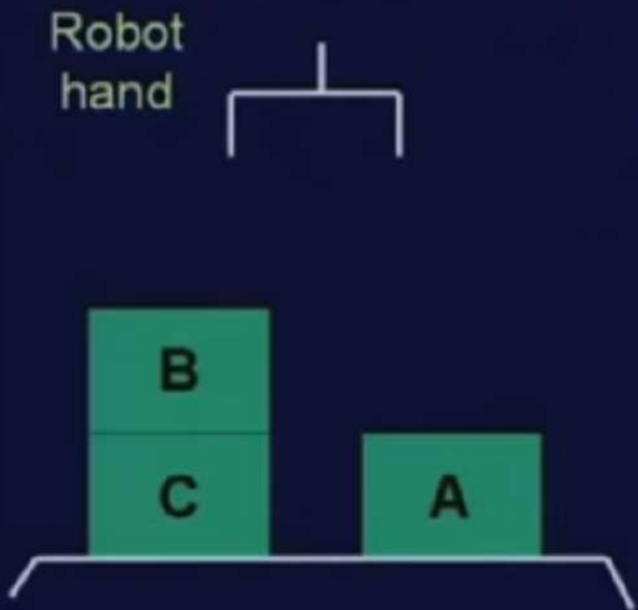
Clear (B, s0)

On (B, C, s0)

Clear (A, s0)

Handempty (s0)

Blocks world example: actions



pickup (B)

Stack (B, A)

do (pickup(B), s0)

the new situation that is the result of applying pickup (B) in the situation s0

S1 = do (*go(store)*, S0)

¬ At (home, S1)

At (store, S1)

At (home, S0)

Colour (door, red)

Frame problem

- I go from home to the store, creating a new situation S' where
  - My son is still at home
  - The store still sells bread
  - My age is still the same
- How can we efficiently represent everything that has not changed?

Indian Institute of Technology, Kharagpur

Successor state axioms

- Normally, things stay true from one state to the next --

  unless an action changes them:

At (X, do(A,S)) iff A = go(X)

or At (X, S) and A != go(Y)]

Successor state axioms

- We need one or more of these for every fluent.
- Now we can use theorem proving to deduce a plan.

STRIPS

- Representation for actions:
  - Preconditions (list of propositions to be true)
  - Delete list (list of propositions that will *become* false)
  - Add list (list of propositions that will *become* true)

Indian Institute of Technology, Kharagpur

Action Representation

<u>STRIPS formalism</u>

Actions must specify all the state variables whose values they change

No disjunction allowed in effect

Preconditions and effects are propositional

Indian Institute of Technology, Kharagpur

Example problem:

Initial state:

At (home), ¬ Have(banana), ¬ Have(noney)

Goal: Have(banana), Have(money), At(home)

Actions:

Buy (X):

Pre: At (store)

Add: Have (X)

Go (X, Y):

Pre: at(X)

Del: at(X)

Add: at(Y)

Indian Institute of Technology, Kharagpur

Frame problem (again)

I go from home to the store, creating a new situation S'. In S':

- My son is still at home
- The store still sells bread
- My age is still the same

- How can we efficiently represent everything that hasn't changed?
  - Strips provides a good solution for simple actions

Ramification problem

- I go from home to the store, creating a new situation S'. In S':
  - I am now in Golebazar
  - The number of people in the store went up by 1
  - The contents of my pockets are now in the store.
- Do we want to say all that in the action definition?

Solutions to the ramification problem

- In Strips, some facts are inferred within a world state,
  - e.g. the number of people in the store
- 'primitive' facts, e.g. At (home) persist between states unless changed. 'inferred' facts are not carried over and must be re-inferred.
  - Avoids making mistakes, perhaps inefficient.

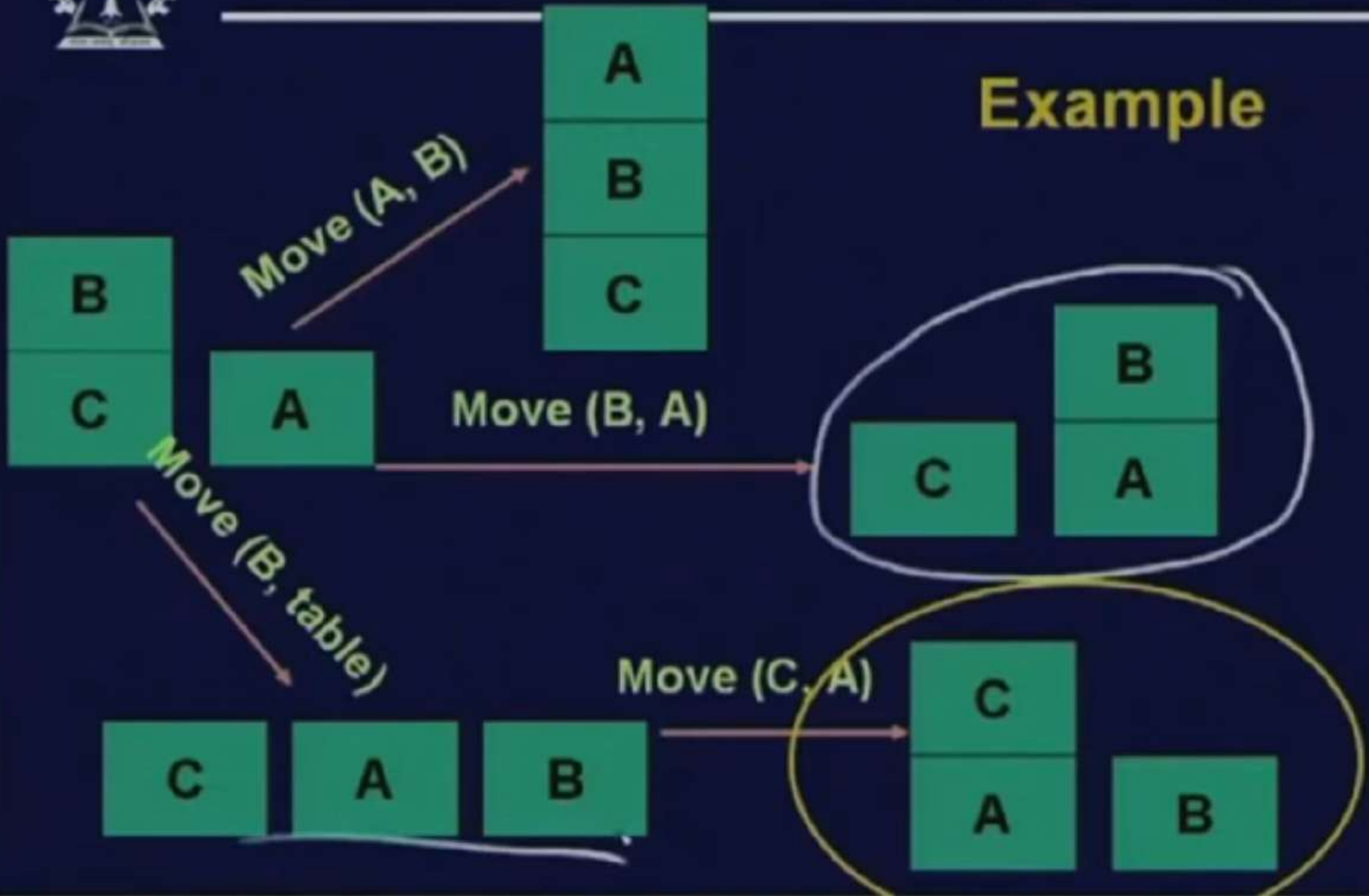
Planning as a search problem

- Given a representation of the initial state, a set of STRIPS operators, and a goal condition we want to achieve:
  - The planning problem is to determine a sequence of actions that when applied to the initial state yields a state which satisfied the goal.

Planning as search

- This can be treated as a search problem.
  - The initial state is given.
  - The actions are operators mapping a state to a new state
  - The goal is satisfied by any state that satisfies the goal

Example



Problems

- Search tree is generally quite large
- The representation suggests some structure. Each action only affects a small set of facts. Actions depend on each other via their preconditions.
- Planning algorithms are designed to take advantage of the special nature of the representation.

Planning with state space search

Forward state space search

- Search from what is known in the initial state and apply operators in the order they are applied.

Backward state space search

- Search from the description of the goal and identify actions that help to reach the goal.

Comparison

- Forward search:
  - branching factor can be extremely high.
  - All applicable actions considered from each state. Includes many irrelevant actions
  - Search is very inefficient without an accurate heuristic.

Comparison

- Backward search:
  - Allows us to consider relevant actions. However there can still be irrelevant actions.
  - Branching factor is usually smaller.

Questions

- Explain how planning systems differ from classical search techniques.
- Formulate the blocks world planning problem.

Questions

1. Explain how planning systems differ from classical search problems.

   - Decomposable sub-goals of a goal
   - States decomposable (conjunction of variables)
   - An action typically changes only a few of the variables.

Questions

2. Formulate the blocks world planning problem.
   - Initial state : Conjunction of propositions
     Ontable(A), On(B, A), Clear(B)
   - Actions: Pickup(X), Stack(X, B)
   - Goal state: On(B, C)

Pickup(X)

Pre: Clear(X), Handempty(), On(X,Y)

Effects: ¬Handempty(), ¬On(X,Y)
Holding(X)

Stack(X,Y)

Pre: Holding(X), ¬Handempty(), Clear(Y)

Effects: ¬Holding(X), ¬Clear(Y)
Handempty, On(X,Y)

Search in Planning

- Given an initial state and a goal state
- How do we generate a sequence of actions?

- Forward state space planning

Forward Planner

Search queue: {Initial state}

Loop

- Pick a state from the search queue
- If it is a goal state, terminate, and return the path from the initial state to this state
- Apply all the applicable actions in the state
  - "Progress" the state through the operator application
  - Put each of the states in the search queue

End

Indian Institute of Technology, Kharagpur

Progression: Forward search (I -> G)

```
Progress (state, goals, actions, path)
 If state satisfies goals, then return path
  else a = choose(actions), s.t.
       preconditions(a) satisfied in state
   if no such a, then return failure
    else return
      Progress ( apply(a, state), goals, actions,
                 concatenate(path, a))
```

First call: Progress (IS, G, Actions, ())

Problems with forward search

Branching factor is the number of legal actions. Path length is the number of actions required to achieve the goal.

In any real world situation there are just too many applicable actions.

- Use heuristics
- Search in the backward direction

Forward Search Example

- Initial state: At (Kharagpur)
- Goal state: At (Leh)
- Actions:

Forward Search Example

- Actions:
  - Train (Kharagpur, Delhi)
  - Train (Kharagpur, Chennai)
  - Train (Kharagpur, Mumbai)
  - Train (Kharagpur, Howrah)
  - Train (Howrah, Delhi)
  - Train (Delhi, Chandigarh)
  - Bus (Chandigarh, Manali)
  - Bus (Manali, Leh)
  - ....

Indian Institute of Technology, Kharagpur

Backward Search

- The goal does not uniquely specify a state, but is a partial description only.
- Given a goal, consider only actions that actually achieve it.

Backward Search

An action A is applicable in state S in the backward direction if:

- The effect of A is consistent with S
- There is at least one effect of A that is part of S

The state resulting from applying A in the reverse direction (the result of regressing S through A) ?

Indian Institute of Technology, Kharagpur

Backward Search

The state resulting from applying A in the reverse direction (the result of regressing S through A):

- Precondition of A +
- the variable value assignments of every state variable not in preconditions of A, but in S.

Backward Search

Termination criterion:

The current backward state's partial assignment is consistent with the variable assignment in the initial state.

Regression: Backward Search (I <- G)

Regress (init-state, current-goals, actions, path)

If init-state satisfies current-goals, then return path

else a =*choose* (actions), s.t. some effect of a satisfies one of current-goals

If no such a, then return failure [unachievable*]

If some effect of a contradicts some of current-goals, then return failure [inconsistent state]

CG' = current-goals – effects(a) + preconditions(a)

If current-goals $\subset$ CG', then return failure [useless*]

RegWS(init-state, CG', actions, concatenate(a,path))

First call: RegWS(IS, G, Actions, ())

STRIPS Planner

- Divide and conquer:

to create a plan to achieve a conjunction of goals,

1. create a plan to achieve one goal, and
2. then create a plan to achieve the rest of the goals.

STRIPS Planner

- To achieve a list of goals:
  - Choose one of them to achieve
  - If it is not already achieved
    - Choose an action that makes the goal true
    - Achieve the preconditions of the action
    - Carry out the action
  - Achieve the rest of the goals

Fixing STRIPS

- Two ideas to make STRIPS sound:
  - Protect sub-goals so that, once achieved, they cannot be undone.
    - Protecting sub-goals makes STRIPS incomplete.
  - Re-achieve sub-goals that have been undone.
    - Re-achieving sub-goals finds longer plans.

Review

- Frame problem:
  - How to specify what does not change
- Qualification problem
  - Hard to specify all preconditions
- Ramification problem
  - Hard to specify all effects

Review

- <u>Situation Calculus</u>
  - <u>Successor state axioms</u>:

$$\text{Broken}(x, do(s,a)) \leftrightarrow [a = drop(x) \wedge fragile(x, s)] \vee$$
$$\exists b[a=explode(b) \wedge nextTo(b, x, s)] \vee$$
$$broken(x,s) \wedge \neg \exists\, a = repair(x)$$

  - <u>Preconditions axioms</u>:

$$\text{Poss}(pickup(r,x), s) \leftrightarrow robot(r) \wedge$$
$$\forall z\, \neg holding(z, x, s) \wedge nextTo(r, x, s)$$

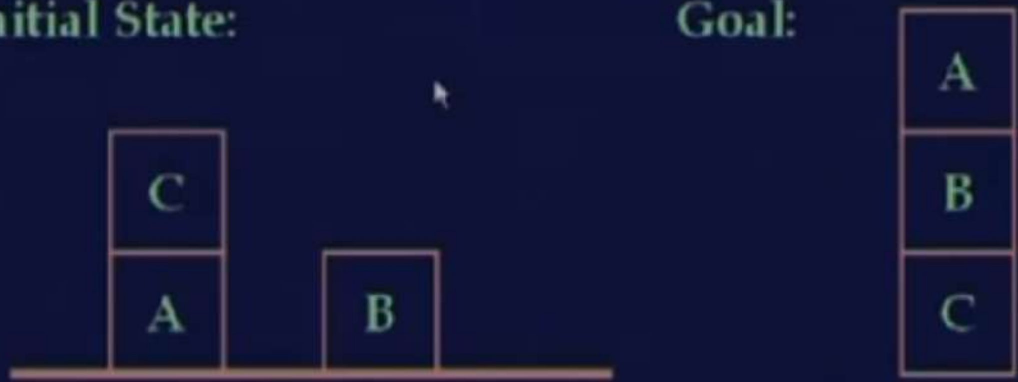
- Strips representation
- Means-ends analysis
- Networks of Actions (Noah)

"Classical" Planning Assumptions

- Deterministic Effects
- Omniscience
- Sole agent of change
- Goals of attainment

Example Problem Instance: "Sussman Anomaly"

Initial State: Goal:



Initial State: (and (on-table A) (on C A) (on-table B) (clear B) (clear C))

Action Representation: Propositional STRIPS

Move-C-from-A-to-Table:

- preconditions: (on C A) (clear C)
- effects:
  - add (on-table C)
  - delete (on C A)
  - add (clear A)

Solution to frame problem: explicit effects are the only changes to the state.

Action Representation: Propositional STRIPS

Move-C-from-A-to-Table:

preconditions: (and (on C A) (clear C))

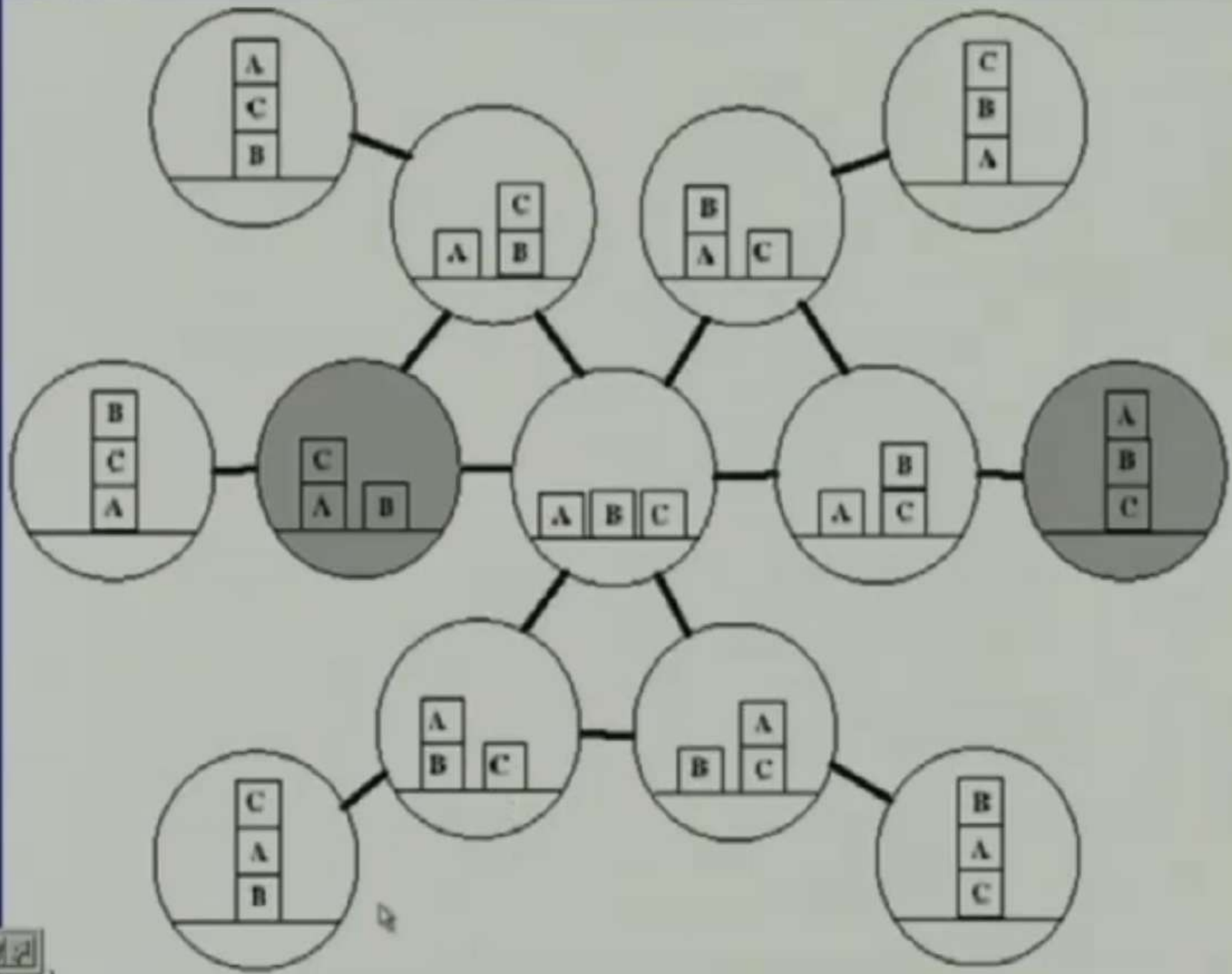
effects:

(and (on-table C)

(not (on C A))

(clear A))

Solution to frame problem: explicit effects are the only changes to the state.



Progression Example

I: (on-table A) (on C A) (on-table B) (clear B) (clear C)
G: (on A B) (on B C)

- P(I, G, BlocksWorldActions, ())
- P(S1, G, BWA, (move-C-from-A-to-table))
- P(S2, G, BWA, (move-C-from-A-to-table, move-B-from-table-to-C))
- P(S3, G, BWA, (move-C-from-A-to-table, move-B-from-table-to-C, move-A-from-table-to-B))

Non-Deterministic Choice!

$G \subseteq S3$ => Success!

Indian Institute of Technology, Kharagpur

Regression Example

I: (on-table A) (on C A) (on-table B) (clear B) (clear C)

G: (on A B) (on B C)

R(I, G, BlocksWorldActions, ())

R(I, ((clear A) (on-table A) (clear B) (on B C)), BWA,
(move-A-from-table-to-B))

R(I, ((clear A) (on-table A) (clear B) (clear C), (on-table B)),
BWA, (move-B-from-table-to-C, move-A-from-table-to-B))

R(I, ((on-table A) (clear B) (clear C) (on-table B) (on C A)),
BWA, (move-C-from-A-to-table, move-B-from-table-to-C,
move-A-from-table-to-B))

Non-Deterministic Choice!

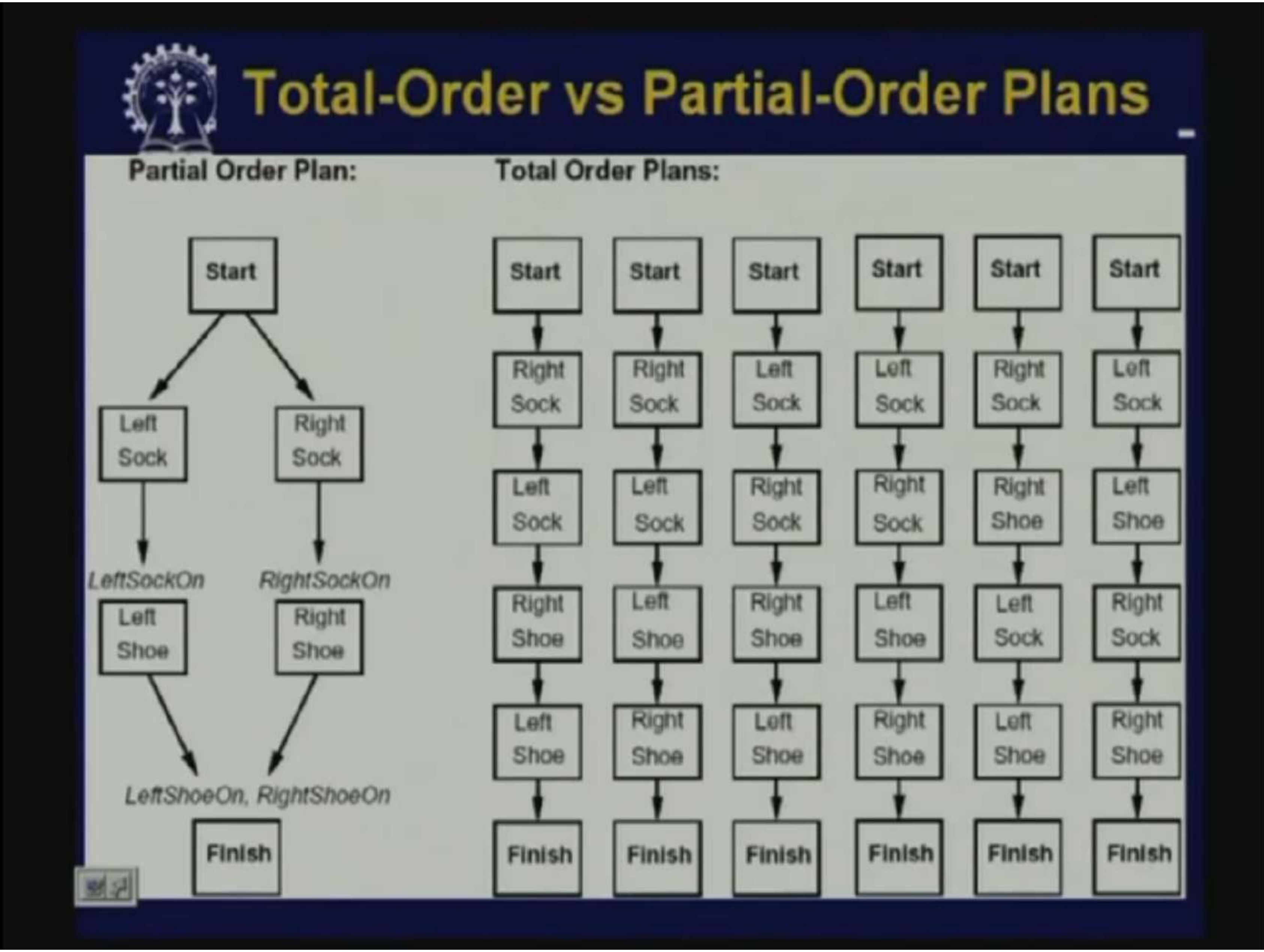
$\boxed{\text{current-goals} \subseteq I \Rightarrow \text{Success!}}$

Indian Institute of Technology, Kharagpur

Progression vs. Regression

- Both algorithms are:
  - Sound: the result plan is valid
  - Complete: if valid plan exists, they find one
- Non-deterministic choice => search!
  - Brute force: DFS, BFS, Iterative Deepening, ..,
  - Heuristic: A*, IDA*, …
- Complexity: $O(b^n)$ worst-case

  b = branching factor, n = |"choose"|
- Regression: often smaller b, focused by goals
- Progression: full state to compute heuristics


Total-Order vs Partial-Order Plans
Partial Order Plan:
Start
Left Sock
Right Sock
LeftSockOn
RightSockOn
Left Shoe
Right Shoe
LeftShoeOn, RightShoeOn
Finish
Total Order Plans:
Start
Right Sock
Left Sock
Right Shoe
Left Shoe
Finish
Start
Right Sock
Left Sock
Left Shoe
Right Shoe
Finish
Start
Left Sock
Right Sock
Right Shoe
Left Shoe
Finish
Start
Left Sock
Right Sock
Left Shoe
Right Shoe
Finish
Start
Right Sock
Right Shoe
Left Sock
Left Shoe
Finish
Start
Left Sock
Left Shoe
Right Sock
Right Shoe
Finish

Plan Generation: Search space of plans

Partial-Order Planning (POP)

- Nodes are partial plans
- Arcs/Transitions are plan refinements
- Solution is a node (not a path).

Principle of "Least commitment"

- e.g. do not commit to an order of actions until it is required

Indian Institute of Technology, Kharagpur

Partial Plan Representation

- Plan = (A, O, L), where
  - A: set of actions in the plan
  - O: *temporal orderings* between actions (a < b)
  - L: *causal links* linking actions via a literal

$$Ap \xrightarrow{Q} Ac$$

- Causal Link:

Action Ac (consumer) has precondition Q that is established in the plan by Ap (producer).

move-a-from-b-to-table $\xrightarrow{\text{(clear b)}}$ move-c-from-d-to-b

Indian Institute of Technology, Kharagpur

Threats to causal links

Step A_t threatens link (A_p, Q, A_c) if:

1. A_t has (not Q) as an effect, and
2. A_t could come between A_p and A_c, i.e.
   $O \cup (A_p < A_t < A_c)$ is consistent

What's an example of an action that threatens the link example from the last slide?

Partial Plan Representation

- Plan = (A, O, L), where
  - A: set of actions in the plan
  - O: *temporal orderings* between actions (a < b)
  - L: *causal links* linking actions via a literal

A1 → A3

$$Ap \xrightarrow{Q} Ac$$

- Causal Link:

Action Ac (consumer) has precondition Q that is established in the plan by Ap (producer).

$$\text{move-a-from-b-to-table} \xrightarrow{\text{(clear b)}} \text{move-c-from-d-to-b}$$

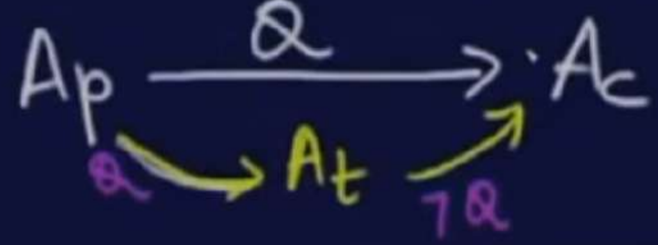
Threats to causal links

Step A_t <u>threatens</u> link (A_p, Q, A_c) if:

1. A_t has (not Q) as an effect, and
2. A_t could come between A_p and A_c, i.e.

   $O \cup (A_p < A_t < A_c)$ is consistent

What's an example of an action that threatens the link example from the last slide?



Threats to causal links

Step A_t <u>threatens</u> link (A_p, Q, A_c) if:

1. A_t has (not Q) as an effect, and
2. A_t could come between A_p and A_c, i.e.
   $O \cup (A_p < A_t < A_c)$ is consistent

What's an example of an action that threatens the link example from the last slide?

Initial Plan

For uniformity, represent initial state and goal with two special actions:

- A_0:
  - no preconditions,
  - initial state as effects,
  - must be the first step in the plan.
- A_∞:
  - no effects
  - goals as preconditions
  - must be the last step in the plan.

POP algorithm

POP((A, O, L), agenda, actions)

If agenda = () then return (A, O, L)

Pick (Q, a_{need}) from agenda

a_{add} = choose(actions) s.t. Q $\in$ effects(a_{add})

If no such action a_{add} exists, fail.

L' := L $\cup$ (a_{add}, Q, a_{need}) ; O' := O $\cup$ ($a_{add} < a_{need}$)

agenda' := agenda - (Q, a_{need})

If a_{add} is new, then A := A $\cup$ a_{add} and

$\forall$P $\in$ preconditions(a_{add}), add (P, a_{add}) to agenda'

Questions

- Describe some examples for which progression planning is appropriate
- Describe some examples where backward regression is more efficient.

Plan Generation: Search space of plans

Partial-Order Planning (POP)

- Nodes are partial plans
- Arcs/Transitions are plan refinements
- Solution is a node (not a path).

Principle of “Least commitment”

- e.g. do not commit to an order of actions until it is required

Terminology

- Step: a step in the partial plan—which is bound to a specific action
- Orderings: s1<s2 s1 must precede s2
- Open Conditions: preconditions of the steps (including goal step)
- Causal Link: $Ap \xrightarrow{Q} Ac$
  a commitment that the condition Q, needed at Ac will be made true by Ap
  - Requires Ap to "cause" Q
    Should have an effect Q

Terminology

- Causal Link
- Unsafe Link: (s1—p—s2; s3)
  if s3 can come between s1 and s2 and undo p (has an effect that deletes p).
- Empty Plan: { S:{I,G}; O:{I<G}, OC:{g1@G;g2@G..}, CL:{}; US:{}}

Partial plan representation

$P = (A, O, L, OC, UL)$

A: set of action steps in the plan $S_0, S_1, S_2 \ldots, S_{inf}$

O: set of action ordering $S_i < S_j, \ldots$

L: set of causal links

$$S_i \xrightarrow{p} S_j$$

OC: set of open conditions (subgoals remain to be satisfied)

UL: set of unsafe links where p is deleted by some action S_k

<u>Flaw:</u> Open condition OR unsafe link

<u>Solution plan:</u> A partial plan with no remaining flaw

- Every open condition must be satisfied by some action
- No unsafe links should exist (i.e. the plan is consistent)

Indian Institute of Technology, Kharagpur

Partial plan representation

$P = (A, O, L, OC, UL)$

A: set of action steps in the plan $S_0, S_1, S_2 \ldots, S_{Inf}$

O: set of action ordering $S_i < S_j, \ldots$

L: set of causal links

$$S_i \xrightarrow{p} S_j$$

OC: set of open conditions (subgoals remain to be satisfied)

UL: set of unsafe links where p is deleted by some action S_k

$$S_i \xrightarrow{p} S_j$$