Planning

Lecture 6

November 3, 2010

News

- Revised proposals due tonight (before morning)
- Discussions by Mark and Prabin

Peer Review Meeting

next week

writeup due November 10

How it is graded

- Report sent to me and the other group
- Grade is for the reviewer, not the project subject
- Of course, I hope this will encourage you to have some progress to talk about during the meeting

Structure

- Meeting notes
- What progress has been made on the project (relative to proposal)?
- What is going well?
- What is going poorly?
- What advice do you have?

Status Report

- Plan a short (informal) presentation on what you’ve done
- Demo any code
- Explain the next steps

Progress Made

- Have they met any milestones?
- Have they started the evaluation?
  - If not, how complete are the evaluation plans?

What is going well/ poorly

- What impressed you with what they’ve done?
- What was harder than they expected?
- Does it look like good science?

Advice

- If you were doing their project, what would you do next?
- What should they make sure to do before December 1?
- Other helpful, constructive comments

Logical Interpretation Review

- First order logic, sentences are true with regard to a model and an interpretation
- Model specifies the objects and relations between them
- Interpretation specifies exactly which objects, relations, functions are referred to by functional symbols

Isomorphic Interpretations

- Two interpretations are fundamentally the same if you can map one to the other just by renaming
- Consider "There exists one person who is liked by two people"
- Which are the same?

Domain: {Alice,Bob,Charlie}
Likes(Alice,Bob) Likes(Bob,Bob)

Domain: {Alice,Bob,Charlie}
Likes(Alice,Bob)
Likes(Charlie ,Bob)

Domain: {Alyssa, Bob, Chuck}
Likes(Chuck,Alyssa)
Likes(Alyssa,Alyssa)

Domain: {A,B,C}
Likes(C,A) Likes(B,A)

Subject to interpretation...

- Are each of the following formulas valid i.e., true for all interpretations? (Remember that the relation names are just names in the formula; don't assume the name has to have any bearing on their interpretation
- For arbitrary a and b and in the domain: atLeastAsWiseAs(a,b) V atLeastAsWiseAs(b,a)
- For arbitrary a in the domain: prime(a) => {odd(a) => prime(a)}
- For arbitrary a and b and in the domain: betterThan(a,b) => ~betterThan(b,a)

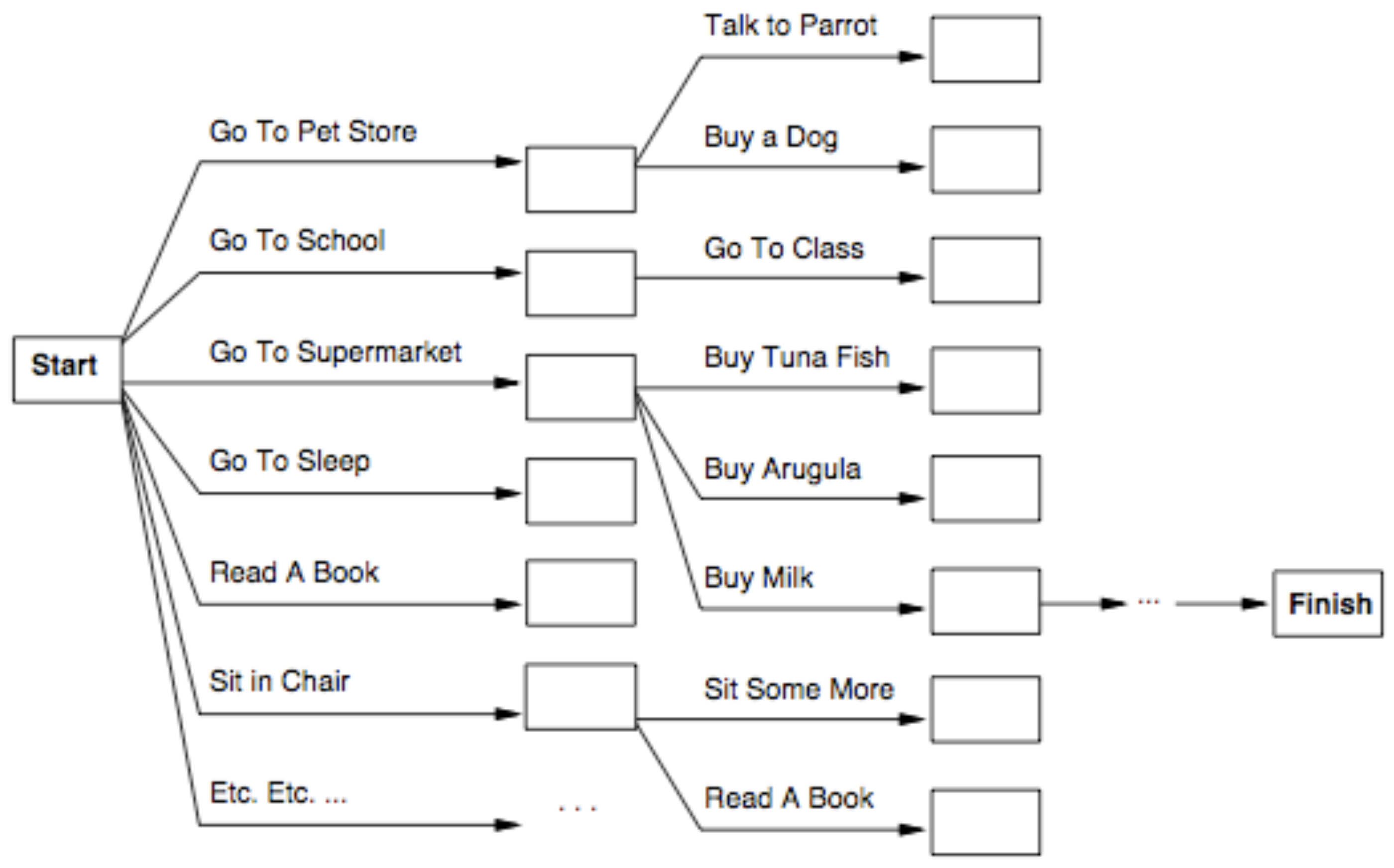
Planning

- Search vs planning
- STRIPS
- Partial Order Planning (POP)
- Graph planning

Search vs Planning

Consider the task *get milk, bananas, and a cordless drill*

Standard search algorithms seem to fail miserably:



Planning as Search

$PlanResult(p, s)$ is the situation resulting from executing p in s

$$PlanResult([], s) = s$$
$$PlanResult([a|p], s) = PlanResult(p, Result(a, s))$$

Initial state $At(Home, S_0) \wedge \neg Have(Milk, S_0) \wedge \ldots$

Actions as Successor State axioms
$Have(Milk, Result(a, s)) \;\Leftrightarrow$
$[(a = Buy(Milk) \wedge At(Supermarket, s)) \vee (Have(Milk, s) \wedge a \neq \ldots)]$

Query
$s = PlanResult(p, S_0) \wedge At(Home, s) \wedge Have(Milk, s) \wedge \ldots$

Solution
$p = [Go(Supermarket), Buy(Milk), Buy(Bananas), Go(HWS), \ldots]$

Principal difficulty: unconstrained branching, hard to apply heuristics

Problems with Search

- Overwhelmed by irrelevant actions
  - Need to know what the result of actions are (buy(x) results in have(x))
- Can't count on user to supply heuristics
  - Need the agent to be able to decompose problem into subgoals
- All comes down to representation

Planning representations (STRIPS/PDDL)

- Use knowledge (logic?) to make planning problem tractable
- States - Conjunction of literals (Poor ^ Unknown)
    - Closed world assumption - any conditions not mentioned are false
- Goals - Partially specified state (Rich ^ Famous) : state s satisfies goal g if s contains all atoms in g (Rich ^ Famous ^ Miserable)
- Action Schema
    - Action name and parameter list *Fly(p,from,to)*
    - Precondition - what must be true before action executed
    - Effect - how state changes when action executed

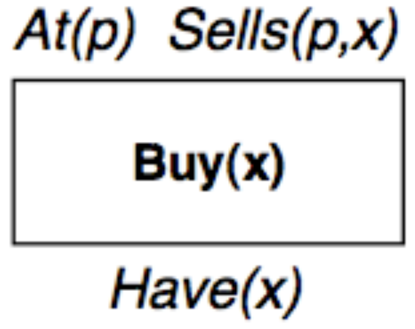
STRIPS Example

Tidily arranged actions descriptions, restricted language

ACTION: $Buy(x)$
PRECONDITION: $At(p), Sells(p, x)$
EFFECT: $Have(x)$

[Note: this abstracts away many important details!]

Restricted language $\Rightarrow$ efficient algorithm
    Precondition: conjunction of positive literals
    Effect: conjunction of literals



State space vs. Plan space

- Standard search: node = concrete world state
- Planning search: node = partial plan
- Defn: **open condition** is a precondition of a step not yet fulfilled
- Operators on partial plans
  - add a link from an existing action to an open condition
  - add a step to fulfill an open condition
  - order one step wrt another

Gradually move from incomplete/vague plans to complete correct plans

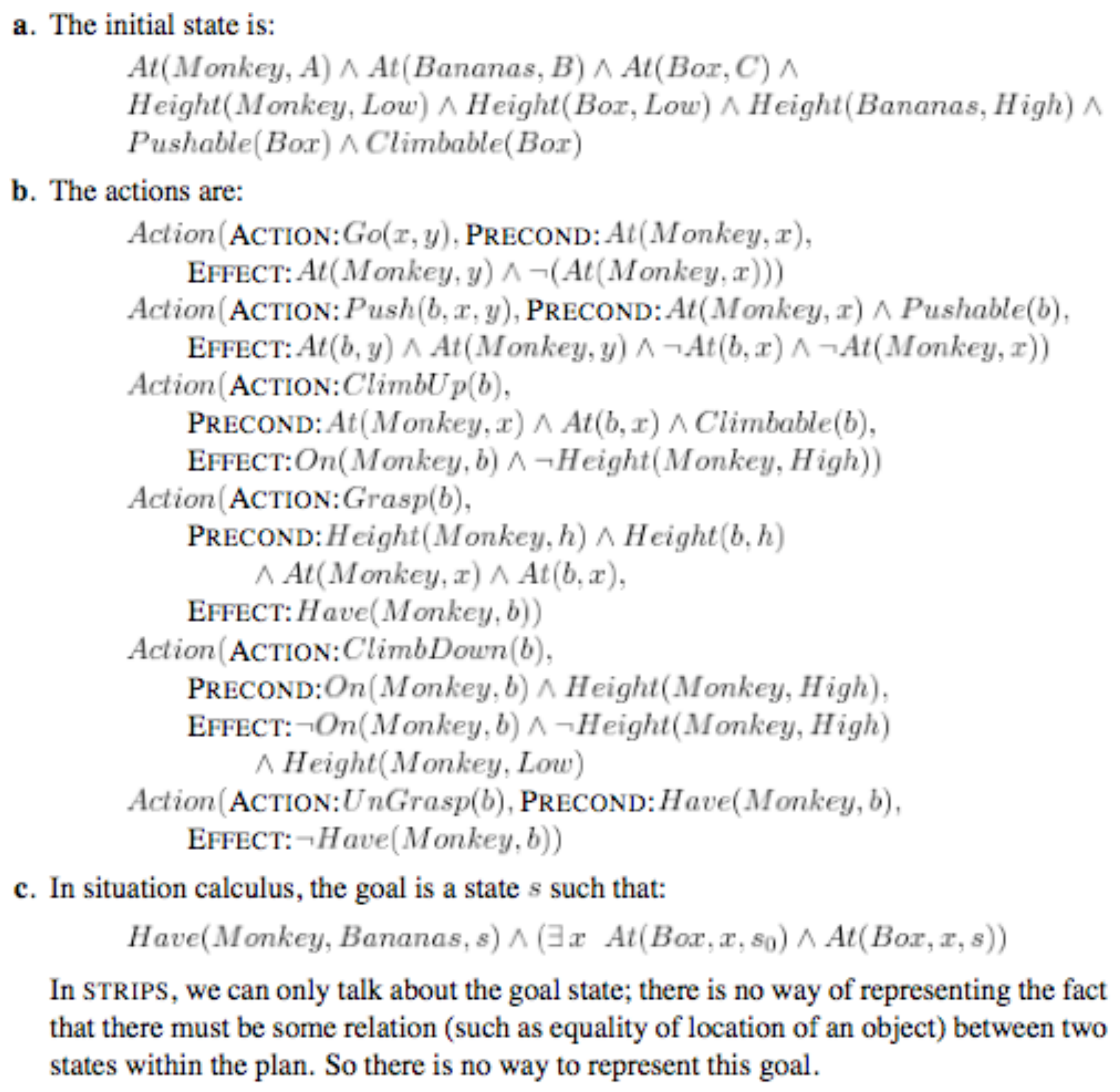
PDDL Description

- A PDDL instance is composed of:
  - An initial state;
  - The specification of the goal states – situations which the planner is trying to reach;
  - A set of actions. For each action, the following are included
    - preconditions (what must be established before the action is performed)
    - effects (what is established after the action is performed).
- Descriptions of actions and states can include "free variables" which will be replaced with literals when the language is used

Exercise

- The monkey-and-bananas problem is faced by a monkey in a laboratory with the bananas hanging out of reach from the ceiling. A box is available that will enable the monkey to reach the bananas if he climbs on it. Initially, the monkey is at A, the bananas at B, and the box at C. The monkey and box have height Low, but if the monkey climbs onto the box he will have height High, the same as the bananas. The actions available to the monkey include Go from one place to another, Push an object from one place to another, ClimbUp onto or ClimbDown from an object, and Grasp or Ungrasp an object. Grasping results in holding the object if the monkey and object are in the same place at the same height.
  - Write down the initial state description
  - Write down definitions of the six actions
  - Suppose the monkey wants to fool the scientists by grabbing the bananas but leaving the box in its original place. Write this as a general goal (not assuming the box is at C). Can this goal be solved by a STRIPS-style (classical planning) system?

Answer



a. The initial state is:

$At(Monkey, A) \wedge At(Bananas, B) \wedge At(Box, C) \wedge$
$Height(Monkey, Low) \wedge Height(Box, Low) \wedge Height(Bananas, High) \wedge$
$Pushable(Box) \wedge Climbable(Box)$

b. The actions are:

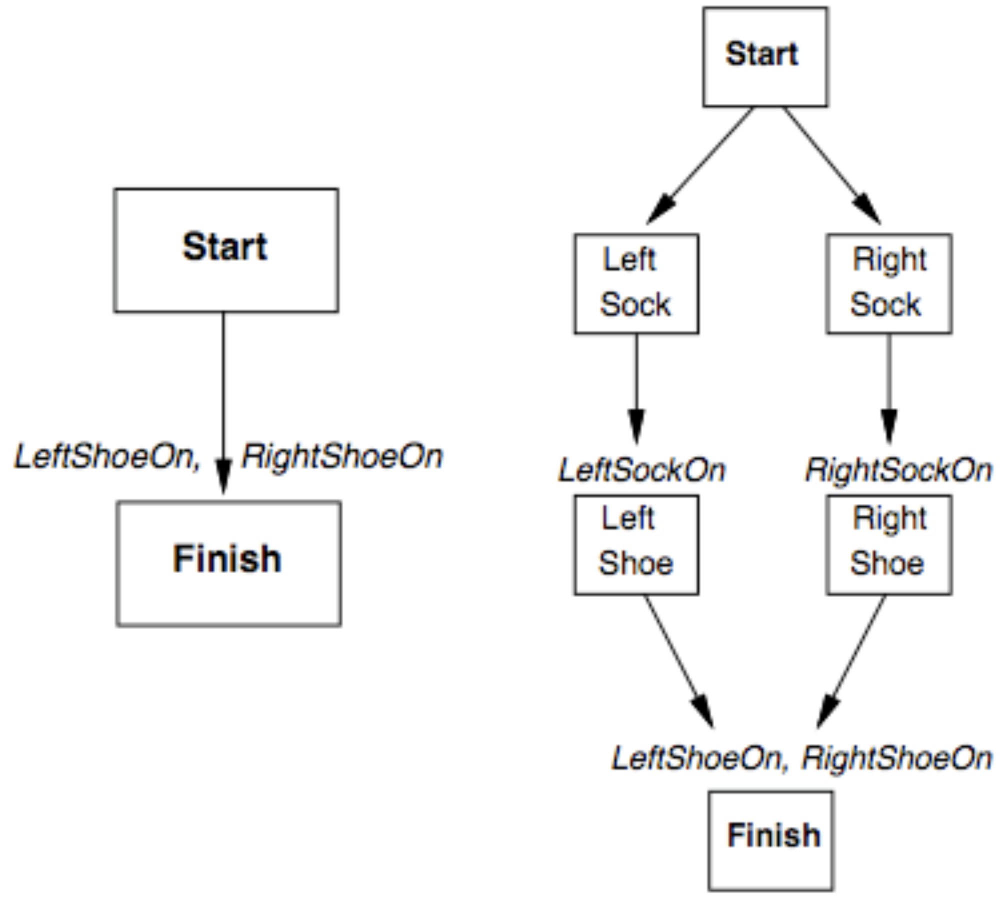
$Action($ACTION:$Go(x, y)$, PRECOND:$At(Monkey, x)$,
  EFFECT:$At(Monkey, y) \wedge \neg(At(Monkey, x)))$
$Action($ACTION:$Push(b, x, y)$, PRECOND:$At(Monkey, x) \wedge Pushable(b)$,
  EFFECT:$At(b, y) \wedge At(Monkey, y) \wedge \neg At(b, x) \wedge \neg At(Monkey, x))$
$Action($ACTION:$ClimbUp(b)$,
  PRECOND:$At(Monkey, x) \wedge At(b, x) \wedge Climbable(b)$,
  EFFECT:$On(Monkey, b) \wedge \neg Height(Monkey, High))$
$Action($ACTION:$Grasp(b)$,
  PRECOND:$Height(Monkey, h) \wedge Height(b, h)$
    $\wedge At(Monkey, x) \wedge At(b, x)$,
  EFFECT:$Have(Monkey, b))$
$Action($ACTION:$ClimbDown(b)$,
  PRECOND:$On(Monkey, b) \wedge Height(Monkey, High)$,
  EFFECT:$\neg On(Monkey, b) \wedge \neg Height(Monkey, High)$
    $\wedge Height(Monkey, Low))$
$Action($ACTION:$UnGrasp(b)$, PRECOND:$Have(Monkey, b)$,
  EFFECT:$\neg Have(Monkey, b))$

c. In situation calculus, the goal is a state s such that:

$$Have(Monkey, Bananas, s) \wedge (\exists x \;\; At(Box, x, s_0) \wedge At(Box, x, s))$$

In STRIPS, we can only talk about the goal state; there is no way of representing the fact that there must be some relation (such as equality of location of an object) between two states within the plan. So there is no way to represent this goal.

Partially ordered plans



A plan is complete iff every precondition is achieved

A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it

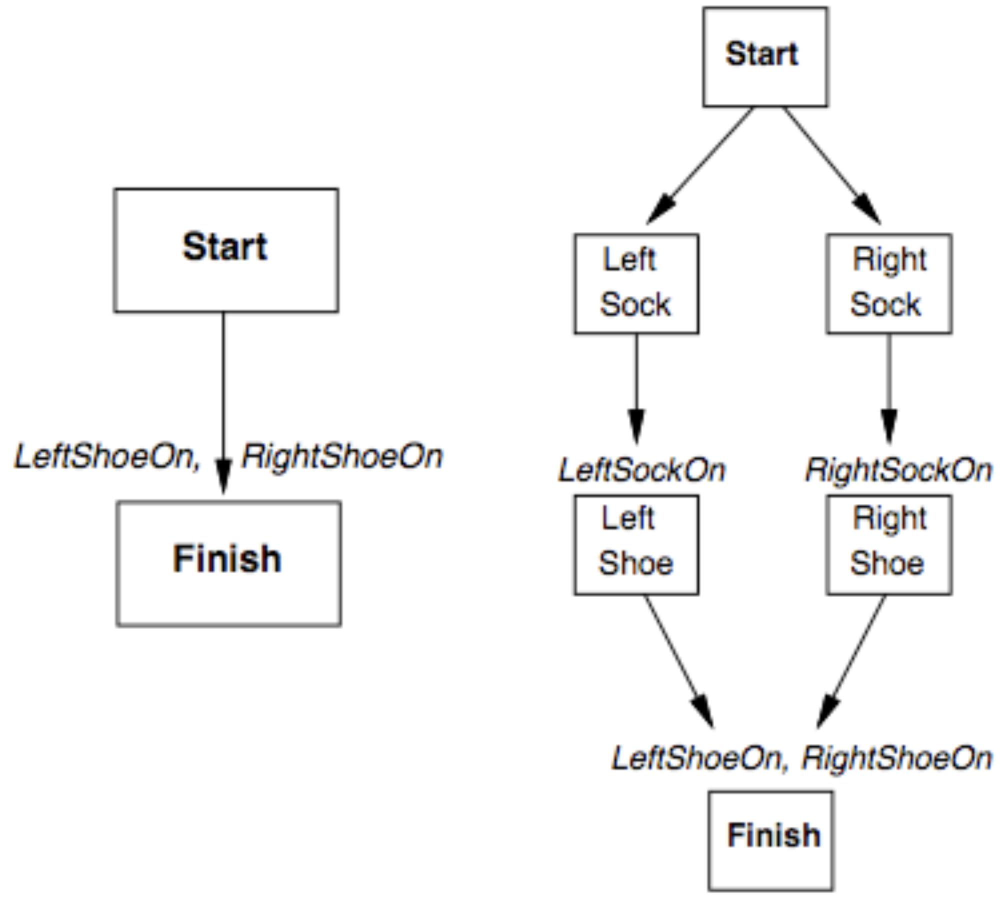
Solutions

- Consistent plan - plan with no cycles in ordering constraints and no conflicts with causal links
- Solution - Consistent plan with no open preconditions

POP as Search

- Initial plan contains *Start* and *Finish*, all preconditions in *Finish* are open preconditions
- Successor function picks open precondition p on an action B and generates a successor plan for every consistent way to choose action A that achieves p
  - Casual link a -> B, p and ordering constraint A<B are added to plan
  - Resolve conflicts between new link and existing actions and new action A and all causal links. Conflict between A->B,p and C is resolved by adding B<C or C<A. Add successor states for either or both if consistent.
- Goal test checks if solution (no open preconditions)

How to solve ?



A plan is complete iff every precondition is achieved

A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it

Heuristics for POP?

Heuristics

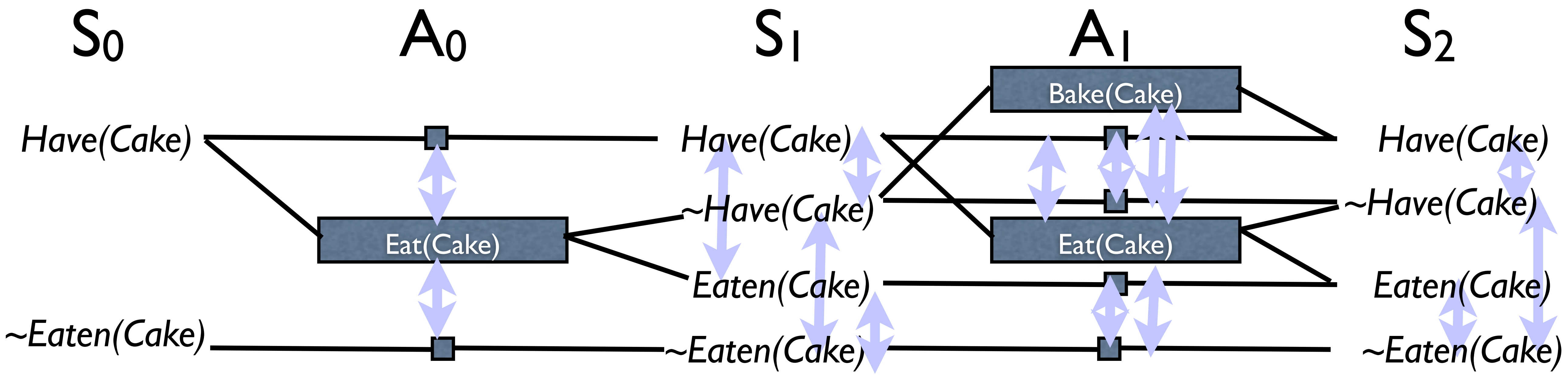
- PRO : easy to decompose into subproblems
- CON - no direct representation of states, no notion of how close to goal
- Heuristic : number of open preconditions
- Most-constrained variable heuristic - pick open condition that can be satisfied in the fewest number of ways

Planning Graphs

- Sequence of levels where level 0 is initial state
- Each level contains literals and actions
  - literals are those that could be true at time step (level) depending on actions at previous levels
  - Also consider inactions
  - Record mutex links between literals that are mutually exclusive
- Requires propositional planning problems (no variables)
- When state Si and Action Ai are identical, the graph has leveled off

Have cake and eat cake too problem

Init(Have(Cake))
Goal(Have(Cake) ^ Eaten(Cake))
Action(Eat(Cake)
    PRECOND: *Have(Cake)*
    EFFECT: *~Have(Cake) ^ Eaten(Cake)*
Action(Bake(Cake)
    PRECOND: *~Have(Cake)*
    EFFECT: *Have(Cake)*



Air transport problem

Init(At(C_1, SFO) ^ At(P_1, SF0) ^ At(P_2, JFK) ^ Cargo(C_1) ^ Cargo(C_1) ^ Cargo(C_2) ^ Plane(P_1) ^ Plane(P_2) ^ Airport(JFK) ^ Airport(SFO))
Goal(At(C_1,JFK) ^ At(C_2,SFO))
Action(Load(c,p,a),
    PRECOND: At(c,a) ^ At(p,a) ^ Cargo(c) ^ Plane(p) ^ Airport(a)
    EFFECT: ~At(c,a) ^ In(c,p))
Action(Unload(c,p,a),
    PRECOND: In(c,p) ^ At(p,a) ^ Cargo(c) ^ Plane(p) ^ Airport(a)
    EFFECT: At(c,a) ^ ~In(c,p))
Action(Fly(p,from,to),
    PRECOND: At(p, from) ^ Plane(p) ^ Airport(from) ^ Airport(to)
    EFFECT: ~At(p,from) ^ At(p,to))

Construct levels 0, 1, 2 for this problem

GraphPlan Algorithm

```
function GRAPHPLAN(problem) returns solution or failure
  graph = INITIAL-PLANNING-GRAPH(problem)
  goals = GOALS[problem]
loop do
  if goals all non-mutex in last level of graph then do
     solution = EXTRACT-SOLUTION(graph, goals, LENGTH(graph)
     if solution != failure then return solution
     else if NO-SOLUTION-POSSIBLE(graph) then return failure
  graph = EXPAND-GRAPH(graph,problem)
```

SATPlan

- Translate a planning problem into propositional axioms
- Apply a satisfiability algorithm to find a model that corresponds to a valid plan
- Choosing the right representation affects the plan
- Goal state must be associated with a particular time

Representation Issues

- Have to specify false states (open world)
- Need to add precondition axioms (conditionals) to prevent illegal actions
- Need to represent mutex links in planning graph
  - Action exclusion axioms
    - (and statements x ^ ~y, y ^ ~x)
  - State constraints (more general statements, hard to write)

Distributed Planning and Problem Solving

- Examples of :
  - Centralized planning for centralized action
  - Centralized planning for distributed action
  - Distributed planning for centralized action
  - Distributed planning for distributed action

Create an expression using the logic of on acting together/BDI

- Then pass your expression to the other groups to be interpreted

Reviewing “On Acting Together”

- How would this paper be evaluated by our course criteria?
- What should the standards be for this sort of paper?
- Hindsight : 375 citations (google scholar)