

Plan Extraction in GraphPlan

Slides draw upon
material from:
Prof. Maria Fox

Brian C. Williams
16.410-13
Session 12-13

Reading and Assignment for Planning Lectures

- Graph-based Planning
AIMA Chapter 11
- AIMA = "Artificial Intelligence: A Modern Approach,"
by Russell and Norvig.
- Assignment
 - No problem set out this week, due to mid-term.
 - Please take the practice mid-term.

Outline

- Review
- GraphPlan
 - Solution Extraction
 - Memos
 - Properties
 - Termination with Failure

Simple Spacecraft Problem

Observation-1
target
instruments

Observation-2


Observation-3

Observation-4

...

pointing

calibrated



States: Target Pointed To, Camera Calibrated?, Has Image?
 Operations: Turn, Take Image, and Calibrate.

Courtesy of NASA.

Example

Init	Actions	Goal
p_c	$p_c \rightarrow \boxed{c} \rightarrow c$	I_A
	$p_x \rightarrow \boxed{T_y} \rightarrow \begin{matrix} p_y \\ \neg p_x \end{matrix}$	
	$\begin{matrix} c \\ p_x \end{matrix} \rightarrow \boxed{Im} \rightarrow I_x$	

Operators in STRIPS Representation

TakelImage (?target, ?instr):
 Pre: Status(?instr, Calibrated), Pointing(?target)
 Eff: Image(?target)

Calibrate (?instrument):
 Pre: Status(?instr, On), Calibration-Target(?target), Pointing(?target)
 Eff: \neg Status(?inst, On), Status(?instr, Calibrated)

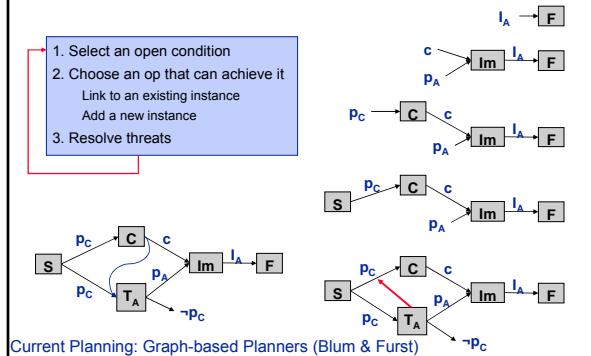
Turn (?target):
 Pre: Pointing(?direction), ?direction \neq ?target
 Eff: \neg Pointing(?direction), Pointing(?target)

Often begin with “?” to denote a parameter, as in ?var.

Based on slides by Dave Smith, NASA Ames

Planning in the Past: Partial Order Causal Link Planning (SNLP, UCPOP)

1. Select an open condition
2. Choose an op that can achieve it
 - Link to an existing instance
 - Add a new instance
3. Resolve threats

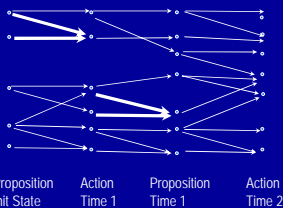


Outline

- Review
- Graph Plan
 - Solution Extraction
 - Memos
 - Properties
 - Termination with Failure

Approach: Graph Plan

1. Constructs compact constraint encoding of state space from operators and initial state, which prunes many invalid plans – Plan Graph.
2. Generates plan by searching for a consistent subgraph that achieves the goals.



Graph Properties

A Plan graph

- compactly encodes the space of consistent plans,
- while pruning . . .
 1. partial states and actions at each time i that are **not reachable** from the **initial state**.
 2. pairs of propositions and actions that are **mutually inconsistent** at time i .
 3. plans that **cannot reach the goals**.

Example Problem: Dinner Date

Initial Conditions: (and (cleanHands) (quiet))

Goal: (and (noGarbage) (dinner) (present))

Actions:

(:operator **carry** :precondition
:effect (and (noGarbage) (not (cleanHands))))

(:operator **dolly** :precondition
:effect (and (noGarbage) (not (quiet))))

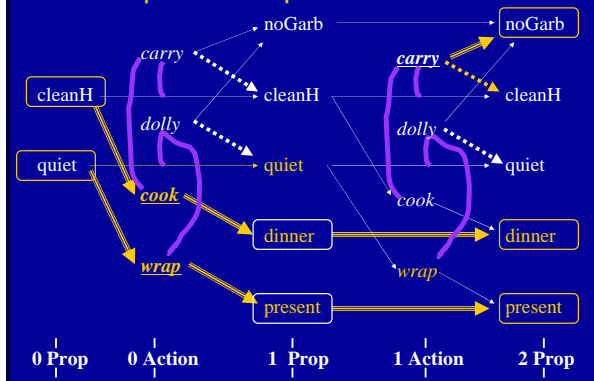
(:operator **cook** :precondition (cleanHands)
:effect (dinner))

(:operator **wrap** :precondition (quiet)
:effect (present))

+ noops

Solution: (Cook, Wrap, Carry)

Example: Graph and Solution



Graphplan

- Create plan graph level 1 from initial state
- Loop
 1. If $goal \subseteq propositions$ of the highest level (nonmutex)
 2. Then search graph for solution
 - If solution found, then return and terminate
 3. Extend graph one more level

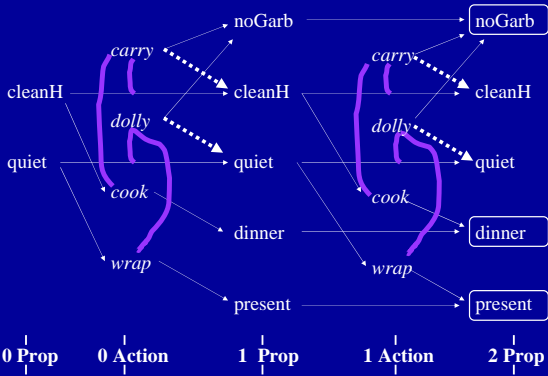
*A kind of double search: forward direction checks necessary (but insufficient) conditions for a solution, ...
Backward search verifies...*

13

Layer 1: Add Proposition Mutexs



Round 2: Extending The Plan Graph



Outline

- Review
- Graph Plan
 - Solution Extraction
 - Memos
 - Properties
 - Termination with Failure

2. Search for a Solution

Recursively find consistent actions achieving all goals at time t , then time $t-1, \dots$:

- Find action to achieve each goal G at time t
 - For each action A making G true at t
 - If A isn't mutex with previously chosen action at t , Then select it
 - Finally,
 - If no action of G works,
 - Then backtrack on previous G .
- Finally
 - If action found for each goal at time t ,
 - Then recurse on preconditions of actions selected, $t-1$,
 - Else backtrack to next solution at $t+1$.

17

2. Search for a Solution

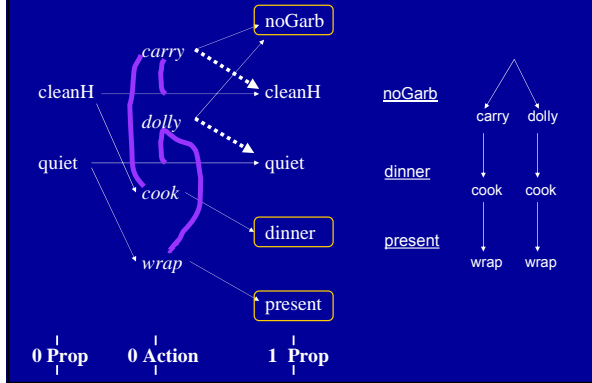
Recursively find consistent actions achieving all goals at time t , then time $t-1, \dots$:

- Find actions at $t-1$ to achieve each goal G at t , by solving CSP_t :
 - Variables: One for each goal G_i
 - Domain: For variable G_i , all actions in layer $t-1$ that add G_i .
 - Constraints: Action mutex of layer $t-1$
- Finally
 - If solution to CSP found (action found for each goal at time t),
 - Then recurse on preconditions of actions selected for layer $t-1$,
 - Else backtrack to next solution at $t+1$.

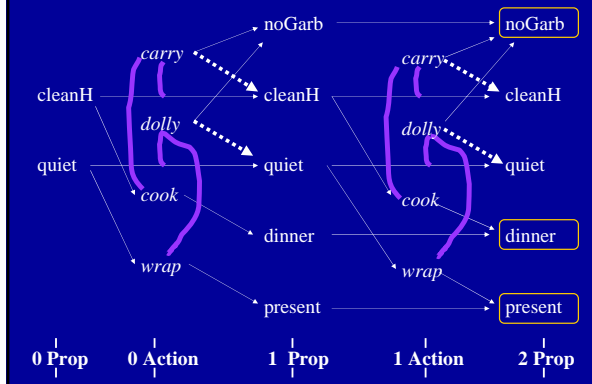
18

- No-ops are always favored.
 - To guarantee that the plan will not contain redundant plan steps.

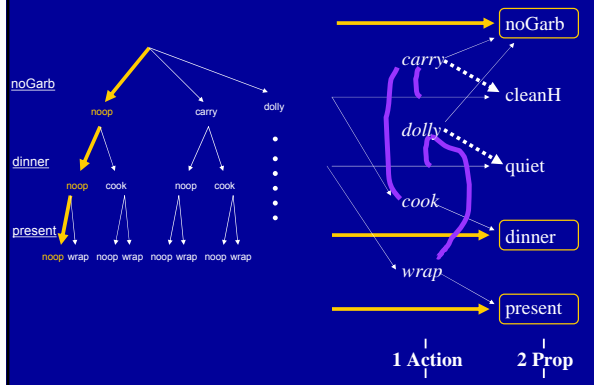
Search Action Layer 0



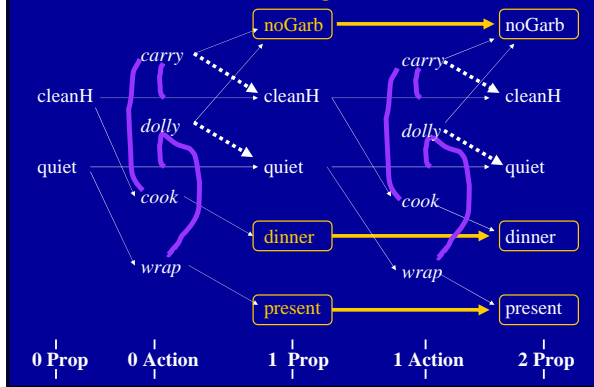
Extend & Search Action Layer 1



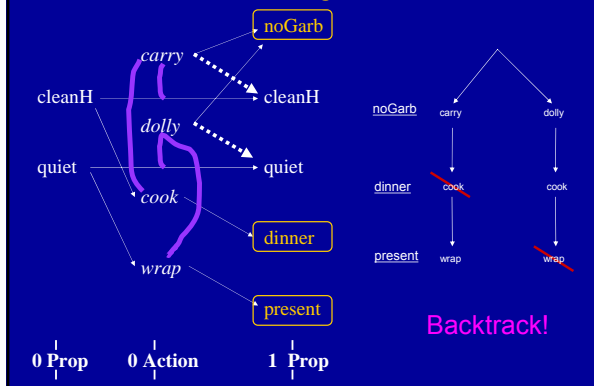
Search Action Layer 1



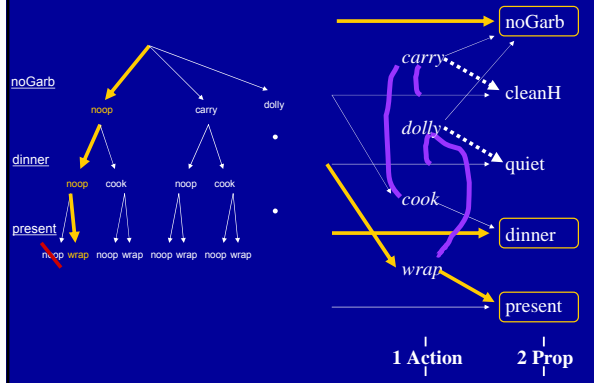
Search Action Layer 0



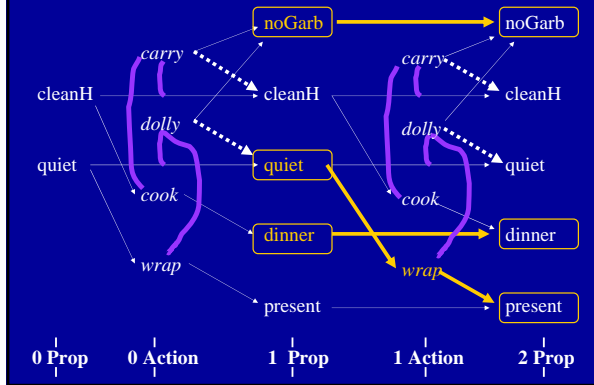
Search Action Layer 0



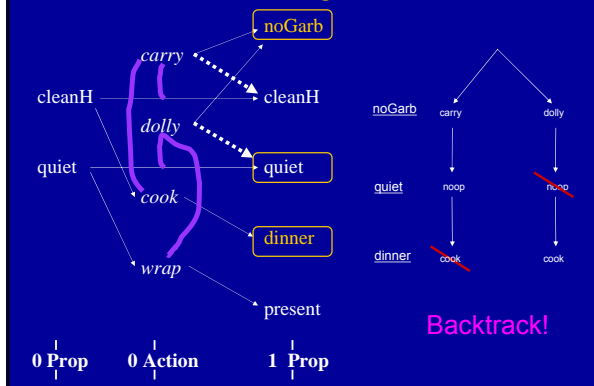
Search Action Layer 1 Again!



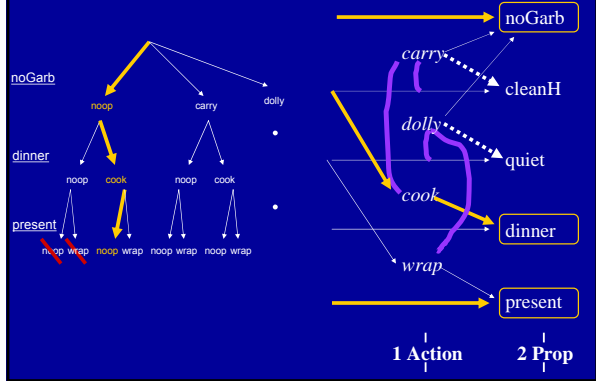
Search Action Layer 0



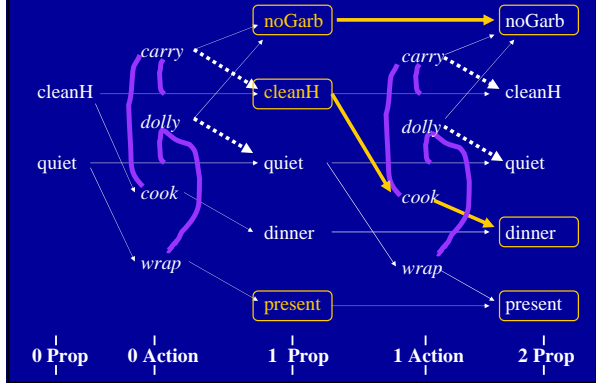
Search Action Layer 0



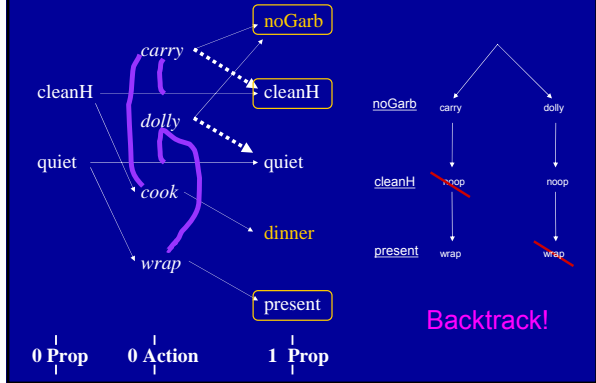
Search Action Layer 1 Again!



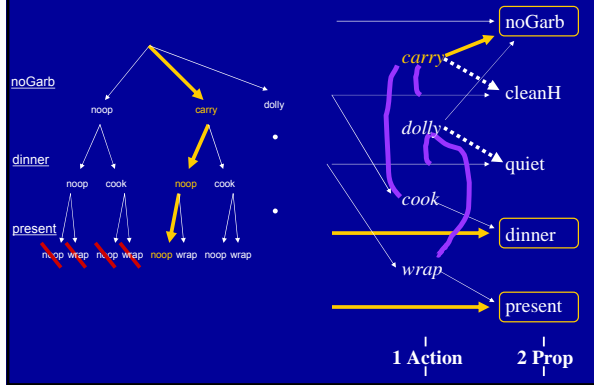
Search Action Layer 0



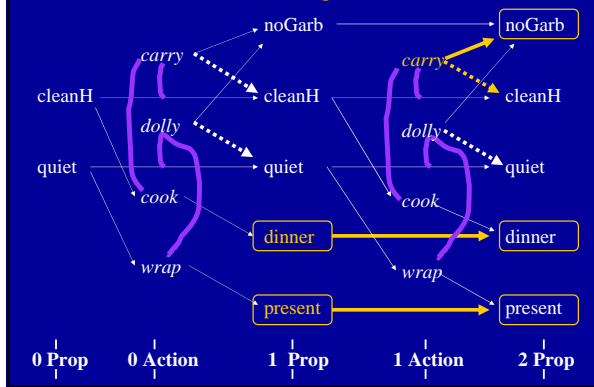
Search Action Layer 0



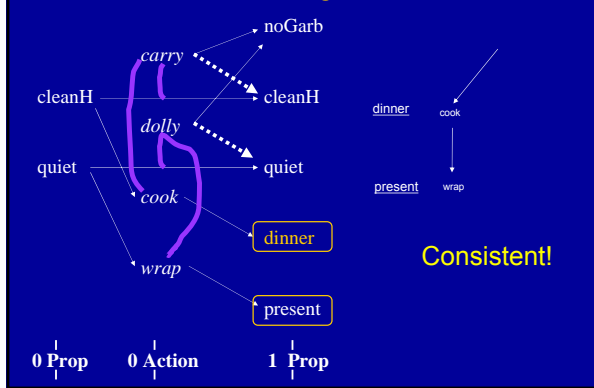
Search Action Layer 1 Again!

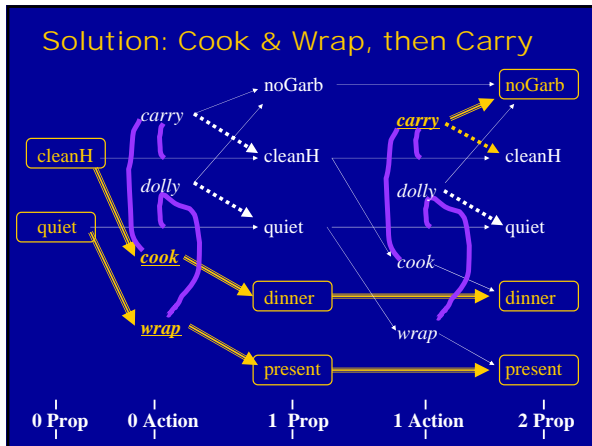


Search Action Layer 0



Search Action Layer 0

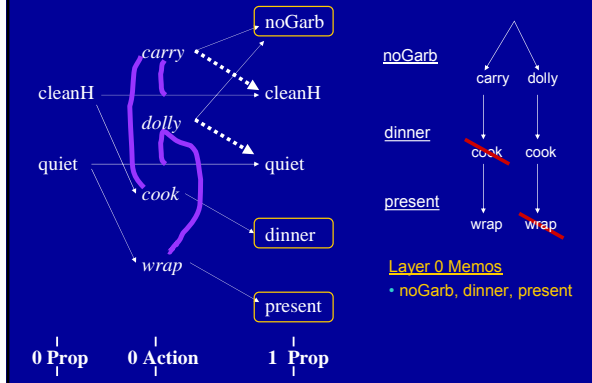




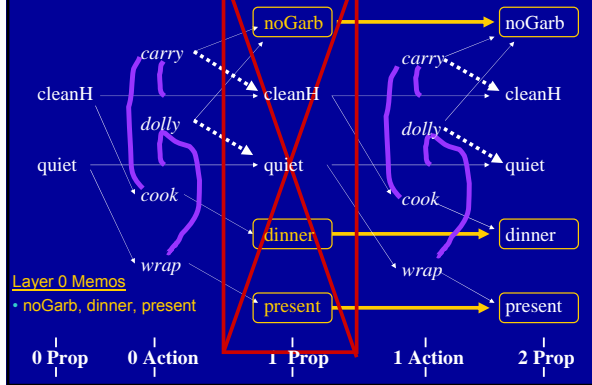
- ## Outline
- Review
 - Graph Plan
 - Solution Extraction
 - Memos
 - Properties
 - Termination with Failure

- ## Memos of Inconsistent Subgoals
- To prevent wasted search effort:
- If a goal set at layer k cannot be achieved, Then memoize set at layer k.
 - Check each new goal set at k against memos.
 - If memo, then fail,
 - Else test by solving a CSP.

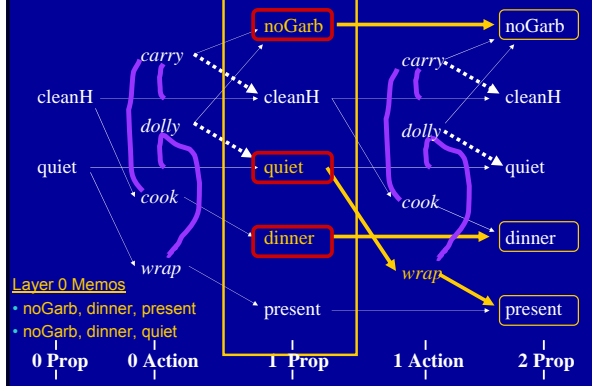
Search Layer 0: Record Memo



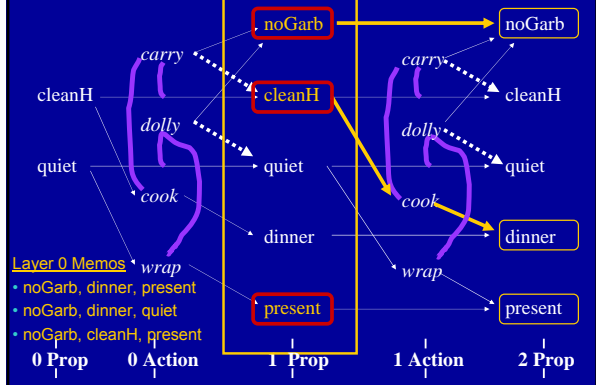
Search Layer 1: Check memos



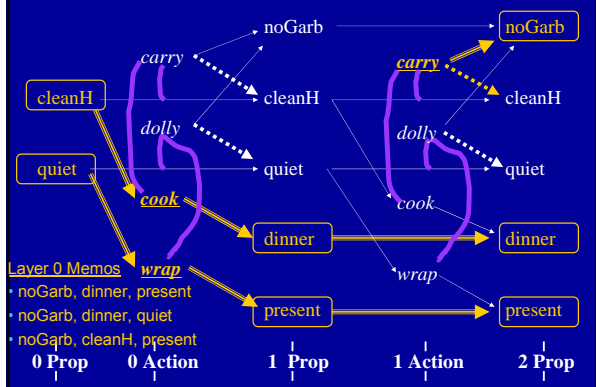
Memo 2



Memo 3



Solution: Not Recorded as a Memo



Outline

- Review
- Graph Plan
 - Solution Extraction
 - Memos
 - Properties
 - Termination with Failure

Plan Graph Properties: Fixed Points

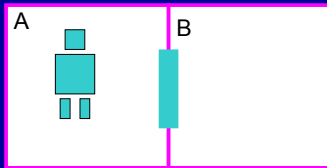
- **Propositions** monotonically **increase**
 - once they are added to a layer they are never removed in successive layers;
 - **Mutexes** monotonically **decrease**
 - once a mutex has decayed it can never reappear;
- The graph will eventually **reach a fix point**
- level where facts and mutexes no longer change.

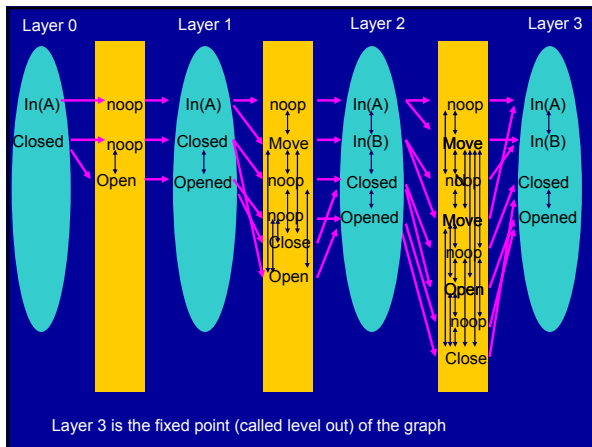
Fix point Example: Door Domain

- Move** from room 1 to room 2
- pre: robot in 1, door is open
 - add: robot in 2
 - del: robot in 1

- Open door**
- pre: door closed
 - add: door open
 - del: door closed

- Close door**
- pre: door open
 - add: door closed
 - del: door open





Graph Search Properties

- Graphplan may need to expand well beyond the fix point to find a solution.
- Why?

Gripper Example

Move from one room to another

- pre: robot in first room
- add: robot in second room
- del: robot in first room

Pick up ball

- pre: gripper free, ball in room
- add: holding ball
- del: gripper free, ball in room

Drop ball

- pre: holding ball, in room
- add: ball in room, gripper free
- del: holding ball

Gripper Example

- Fix point occurs at layer 4,
 - All facts concerning ball and robot locations are pairwise non-mutex after 4 steps.
- Solution layer depends on # balls moved.
 - E.g., for 30 balls
 - solution is at layer 59,
 - 54 layers with identical facts, actions and mutexes.

Properties: Optimality and Redundancy

- Plans guarantee **parallel optimality**.
 - Parallel plan will take as short a time as possible.
- Plans don't guarantee **sequential optimality**.
 - Might be possible to achieve all goals at a later layer using fewer actions.
- Plans do not contain **redundant steps**.
 - By preferring no-ops.

Outline

- Review
- Graph Plan
 - Solution Extraction
 - Memos
 - Properties
 - **Termination with Failure**

Termination Property

- Graphplan returns failure if and only if no plan exists.

Simple Termination

- If the fix point is reached and:
 - A goal is not asserted OR
 - Two goals are mutex
- Then return "No solution," without any search .
- Else may be higher order exclusions (memos), preventing a solution.
- Requires more sophisticated termination test.

Why Continue After FixPoint?

- propositions, actions and mutexes no longer change after fix point.
- N-ary exclusions (memos) DO change.
 - New layers add time to graph.
 - Time allows actions to be spaced so that memos decay.
 - Memos monotonically decrease
 - Any goal set achievable at layer i , is achievable at $i + n$.
- Track memos & terminate on their fix point.

Recap: Graph Plan

- Graphplan developed in 1995 by Avrim Blum and Merrick Furst, at CMU.
- Graphplan searches a compact encoding of the state space, constructed from the operators and initial state.
- Encoding pre-prunes many invalid plans that violate reachability and mutual exclusion.
- Graphplan has been extended to reason with temporally extended actions, metric and non-atomic preconditions and effects.
