Algorithmic Human-Robot Interaction

Developmental Milestones/Project Inspiration

Task Planning IV: Planning and Search – Detailed Dive

CSCI 7000

Prof. Brad Hayes

Computer Science Department

University of Colorado Boulder

Final Projects

Behavior coaching using reward saliency and models of how people consider reward

Shruthi Sukumar and Lakham Kamireddy

Navigation assistant for vision-impaired: path planning, scene analysis, verbal summarization of scene/map

Shivendra Agrawal, Dhanendra Soni, Ian Loefgren

Conversational planner to augment plan cost function through visualization

Matthew Luebbers and Shohei Wakayama

Learning motion planning/grasp preferences from human feedback (e.g., don't move sharp end of scissors next to people)

Jack Kawell and Nishank Sharma

Sim2Real via AR or Pantomime

Ryan Leonard, Karthik Palavalli, Ashwin Vasan, Chandan Naik

Thursday's Papers (2/14):

Probabilistically Safe Robot Planning with Confidence-Based Human Predictions
Jaime Fisac et al.

PRO presenter:
CON presenter:

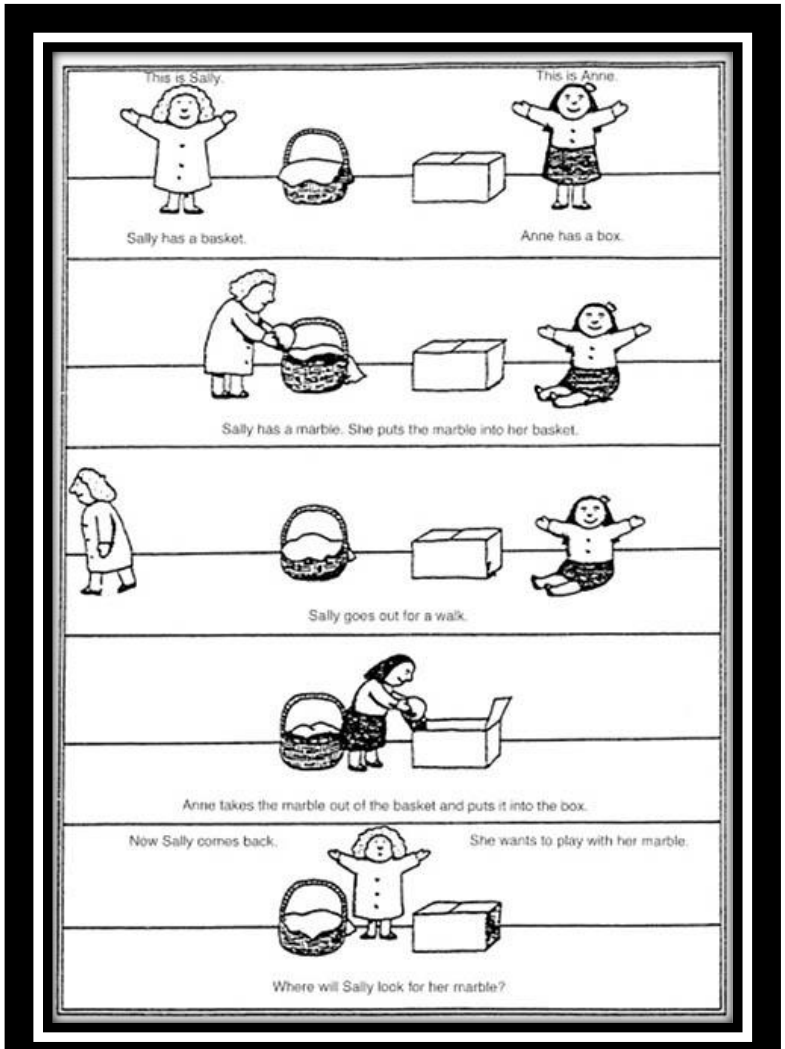
An Implemented Theory of Mind to Improve Human-Robot Shared Plans Execution
Sandra Devin and Rachid Alami

PRO presenter: Matthew Luebbers
CON presenter: Shohei Wakayama

Project Inspiration

Some developmental milestones worth observing…

Sally-Anne Test of False Belief



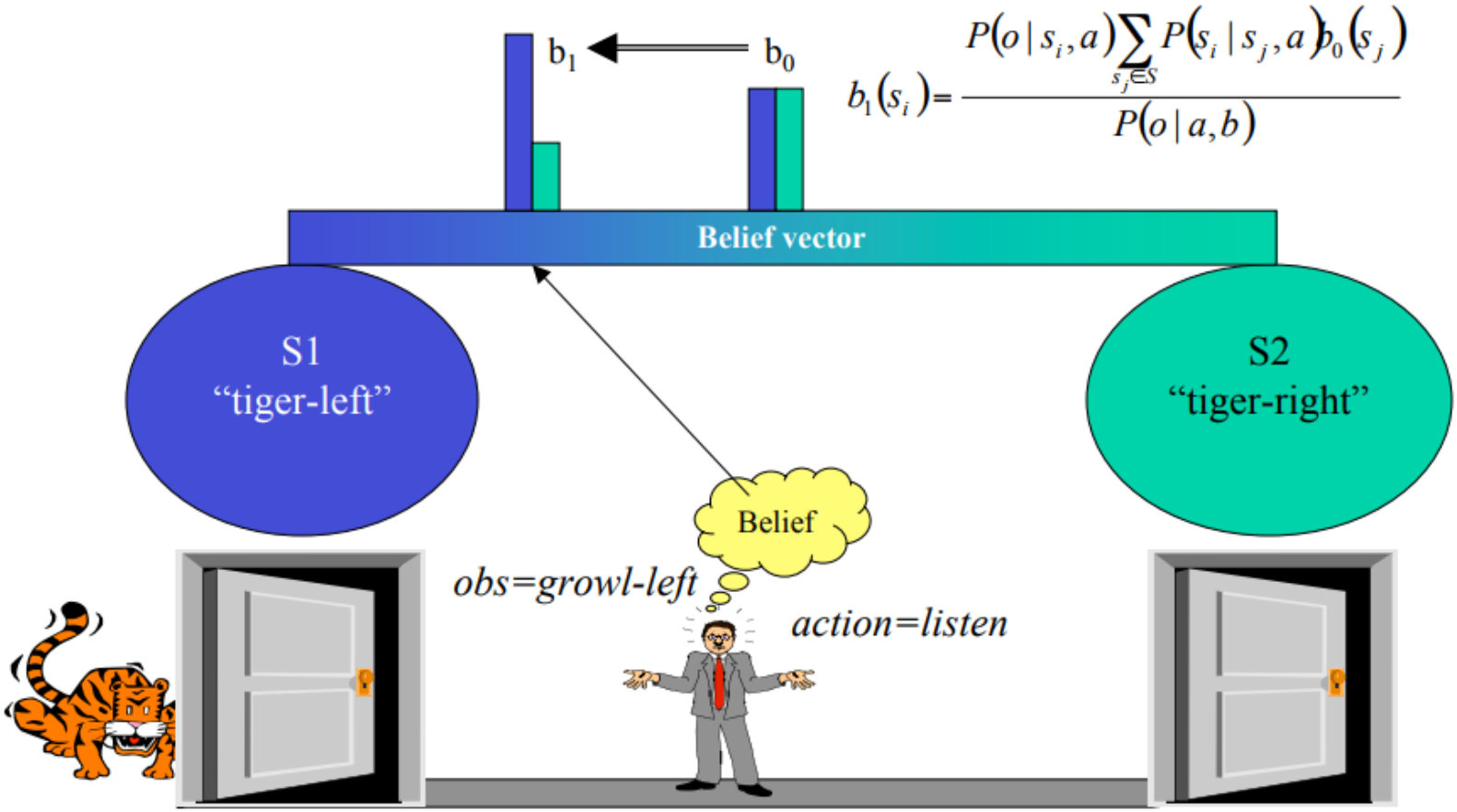
Children under the age of four do not do well on this test!

Altruism in Children

Heider and Simmel Animation

Last Time…

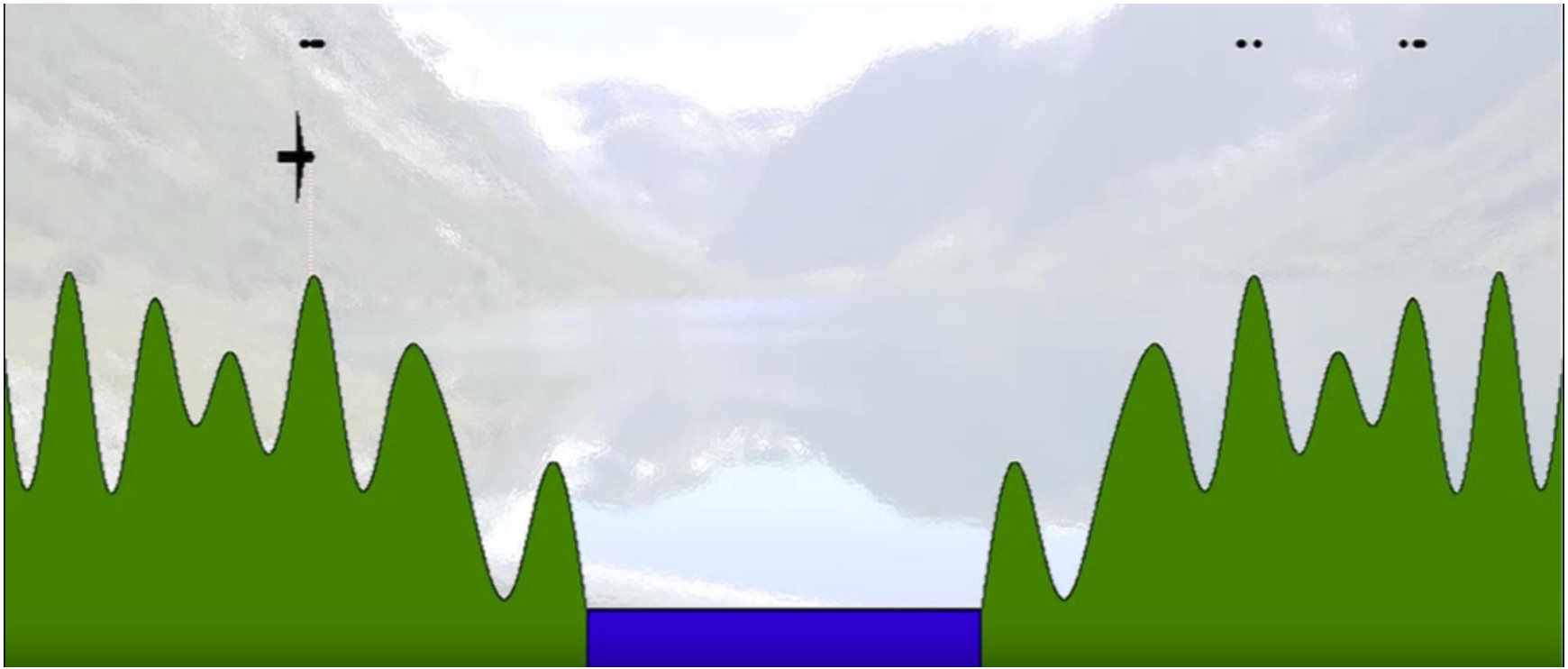
POMDP: The Tiger Problem



Particle Filter: Exhaust/Resample

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.

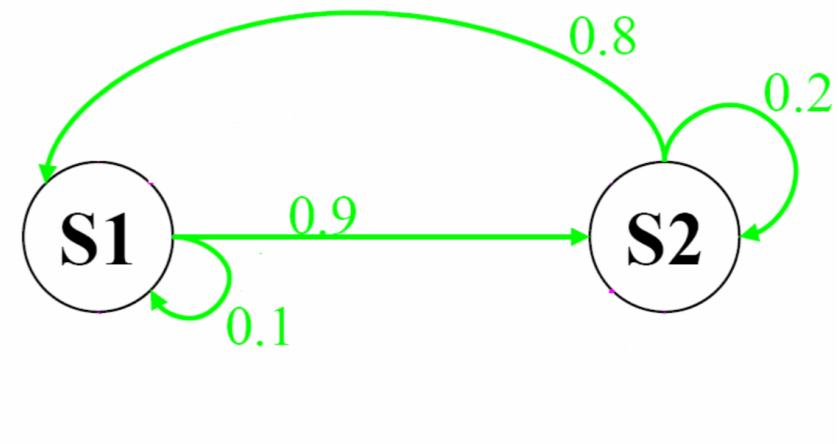


Markov Model Chart

Are the states completely observable? \ Do we have control over the state transitions? (Are we picking which actions are executed)	NO	YES
YES	Markov Chain	MDP
NO	HMM	POMDP

Markov Chains

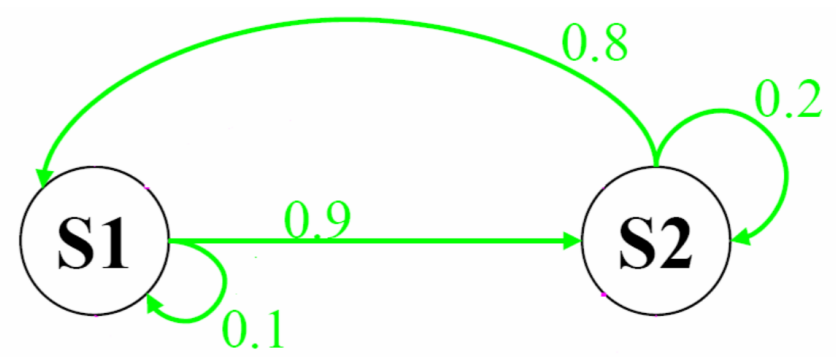
- Finite number of discrete states
- Probabilistic transitions between states
- Next state determined only by the current state
  - This is the Markov property



Rewards: S1 = 10, S2 = 0

Hidden Markov Model

- Finite number of discrete states
- Probabilistic transitions between states
- Next state determined only by the current state
- We're unsure which state we're in
  - The current states emits an observation

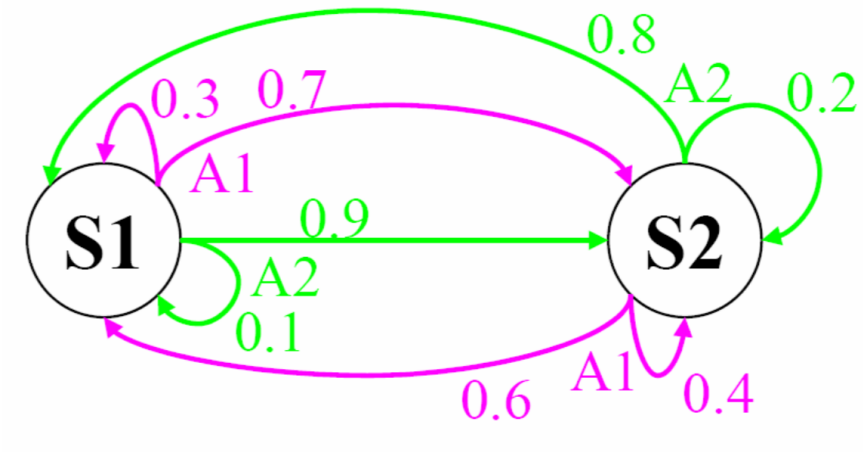


Rewards: S1 = 10, S2 = 0

Do not know state:
S1 emits O1 with prob 0.75
S2 emits O2 with prob 0.75

Markov Decision Process

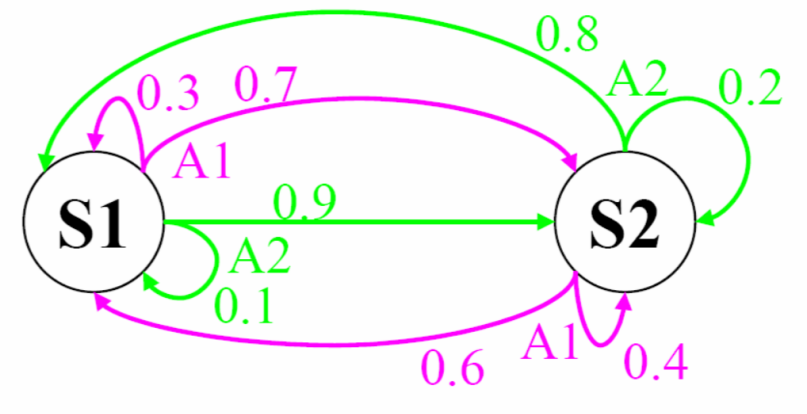
- Finite number of discrete states
- Probabilistic transitions between states ***and*** controllable actions in each state
- Next state determined only by the current state ***and*** current action
  - This is still the Markov property



Rewards: S1 = 10, S2 = 0

Partially Observable Markov Decision Process

- Finite number of discrete states
- Probabilistic transitions between states and controllable actions
- Next state determined only by the current state and current action
- We're unsure which state we're in
  - The current state emits observations

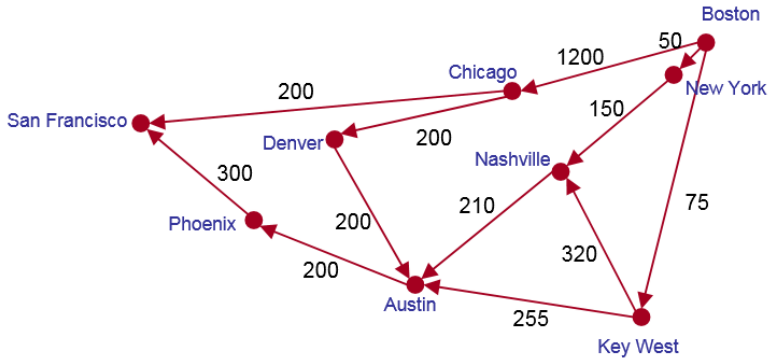


Rewards: S1 = 10, S2 = 0

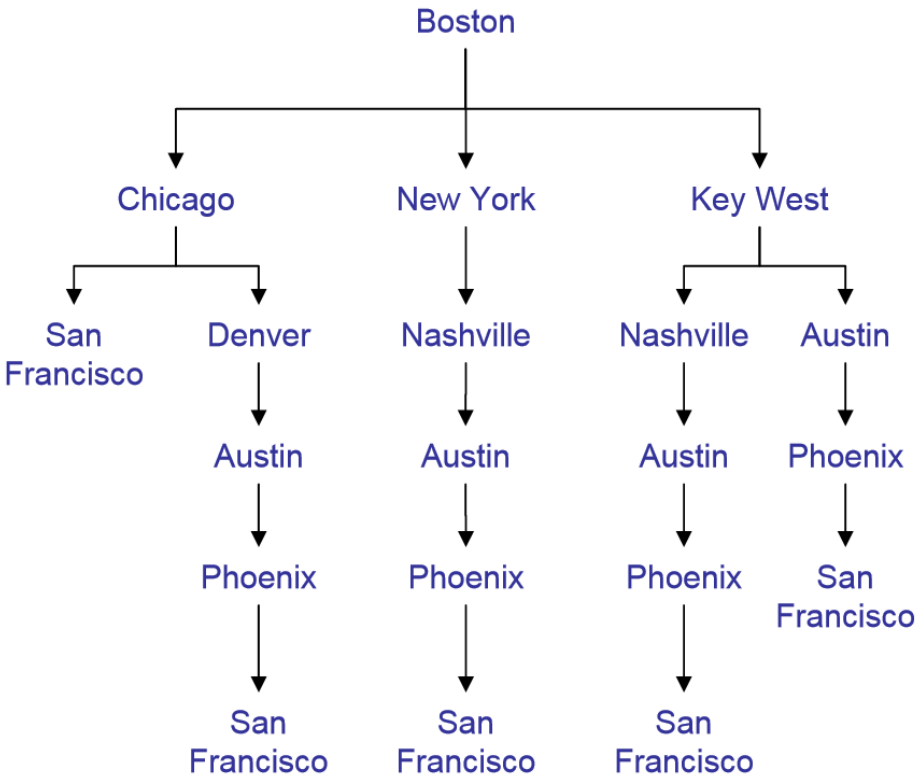
Do not know state:
S1 emits O1 with prob 0.75
S2 emits O2 with prob 0.75

Planning Deep Dive

Search as a Problem-Solving Technique

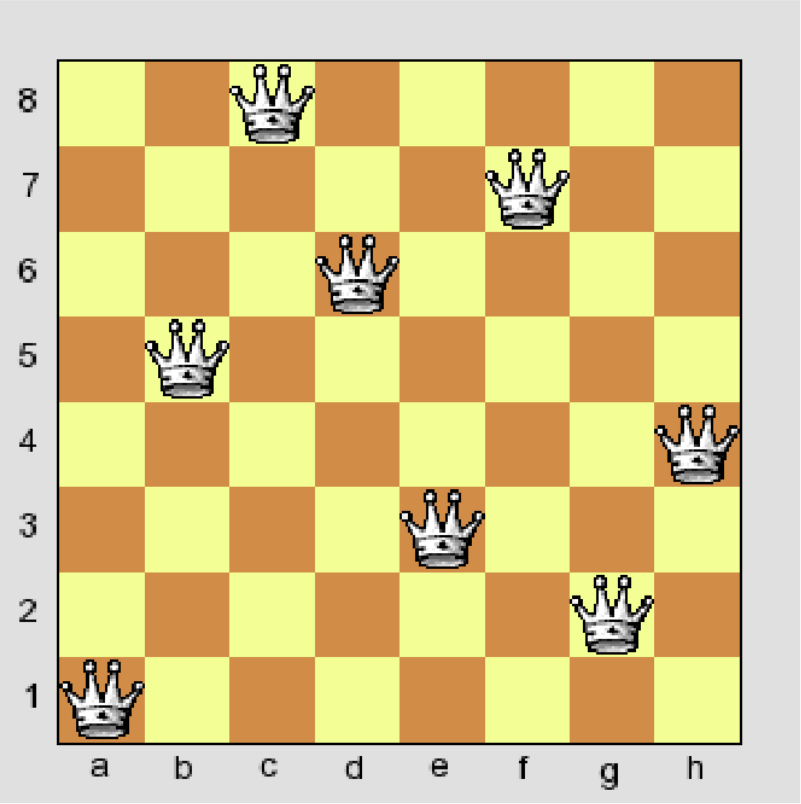


Depth
0
1
2
3
4
5



Branching Factor b=3

Problem Formulation Matters! The 8 Queens Problem



- Formulation #1:
  - Place a queen on any open square
  - Repeat until all queens are placed
  - State space of $\frac{64!}{56!} = 1.78 * 10^{14}$
- Formulation #2:
  - Place a queen on any row 1 square
  - Place a queen on any row 2 square
  - State space of $8^8 = 1.68 * 10^7$
- Formulation #3:
  - Place a queen on any row 1 square
  - Place a queen on any row 2 square not sharing a column…
  - State space of $8! = 40{,}320$

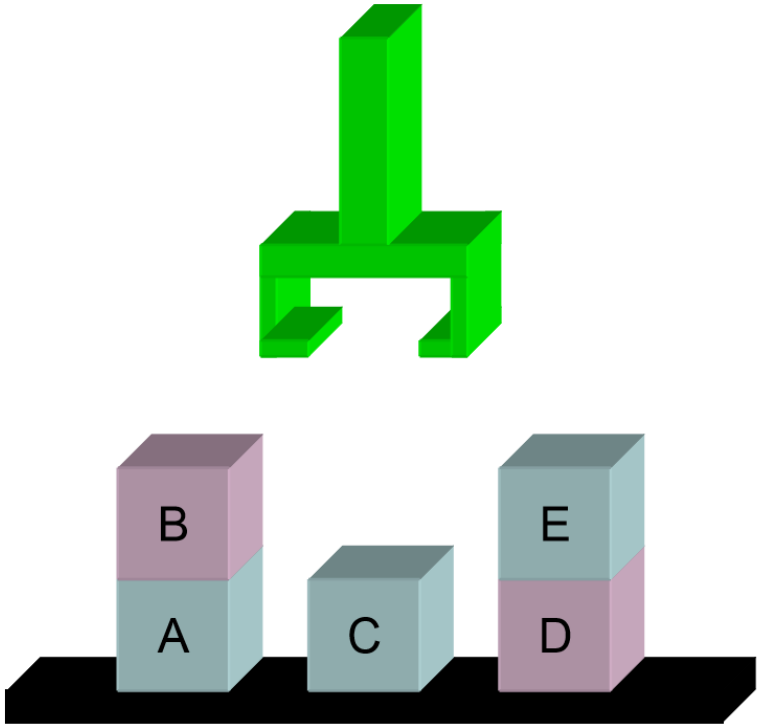
STRIPS: Language for Classical Planning

- A **problem** in Strips is a tuple $P = \langle F, O, I, G \rangle$:
  - F stands for set of all **atoms** (boolean vars)
  - O stands for set of all **operators** (actions)
  - $I \subseteq F$ stands for **initial situation**
  - $G \subseteq F$ stands for **goal situation**
- Operators $o \in O$ **represented** by
  - the **Add** list $Add(o) \subseteq F$
  - the **Delete** list $Del(o) \subseteq F$
  - the **Precondition** list $Pre(o) \subseteq F$

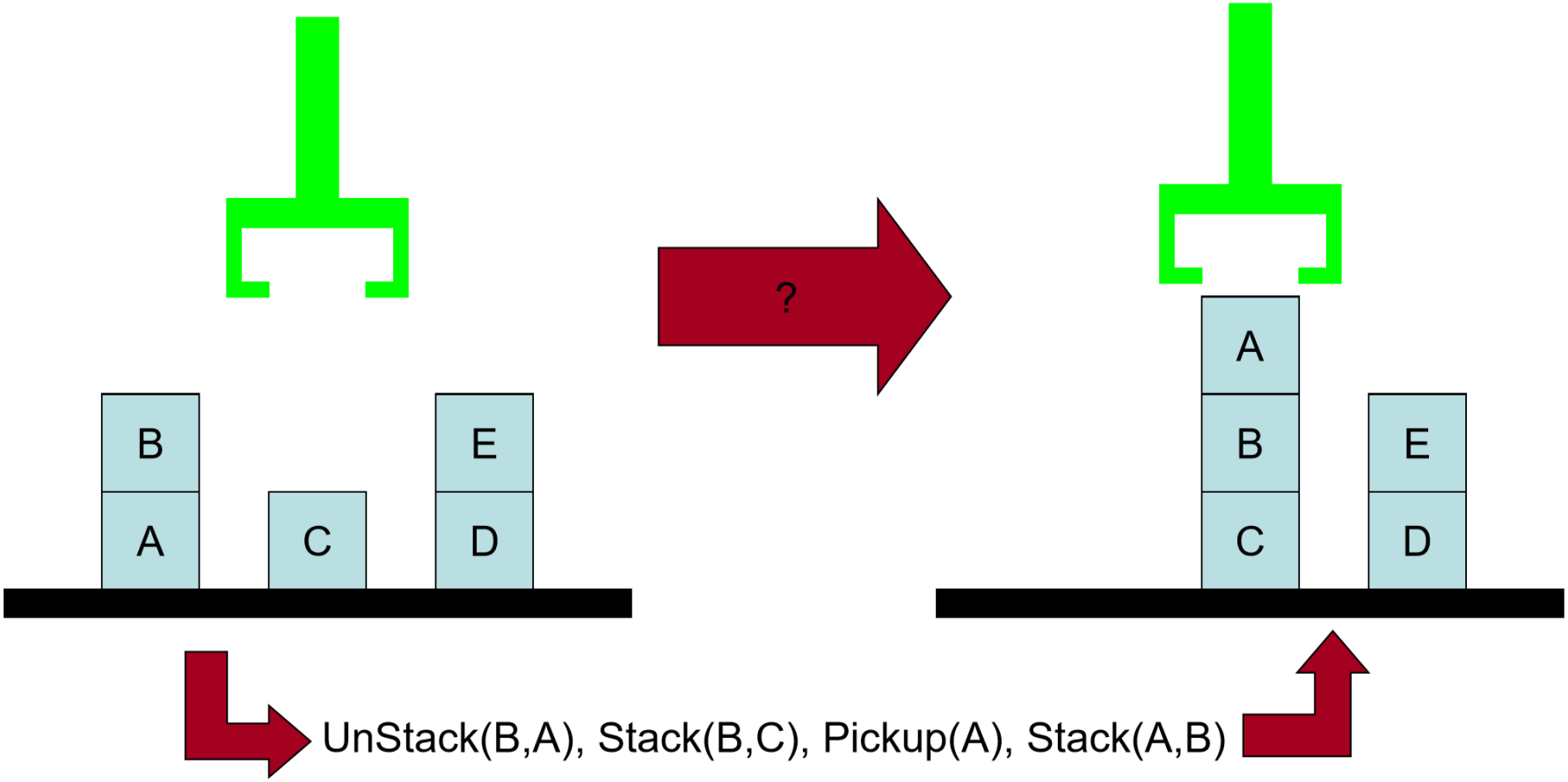
  - Pickup(X)
    - P: grip(∅) ∧ clear(X) ∧ ontable(X)
    - A: grip(X)
    - D: onTable(X) ∧ grip(∅)

Blocks World

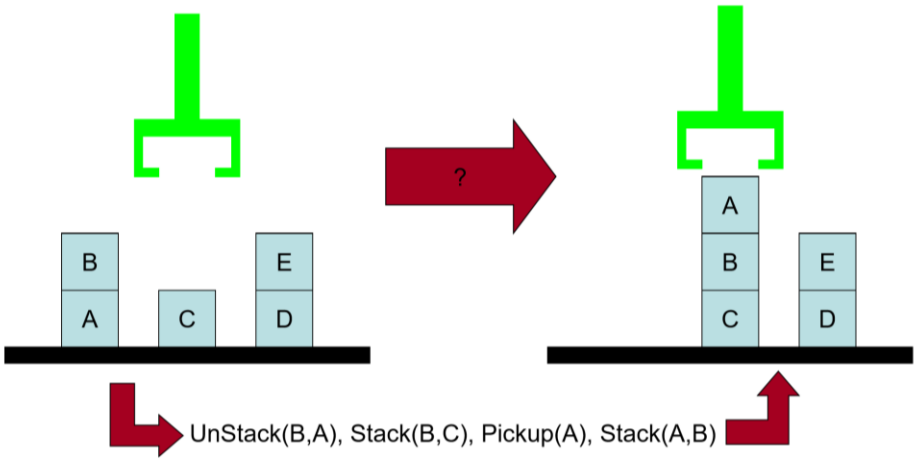
- Robot Gripper
- Square objects on a perfect, flat table
- Predicates to describe world:
  - On(x,y)
  - TopClear(x)
  - Grip(x)



Planning in Blocks World

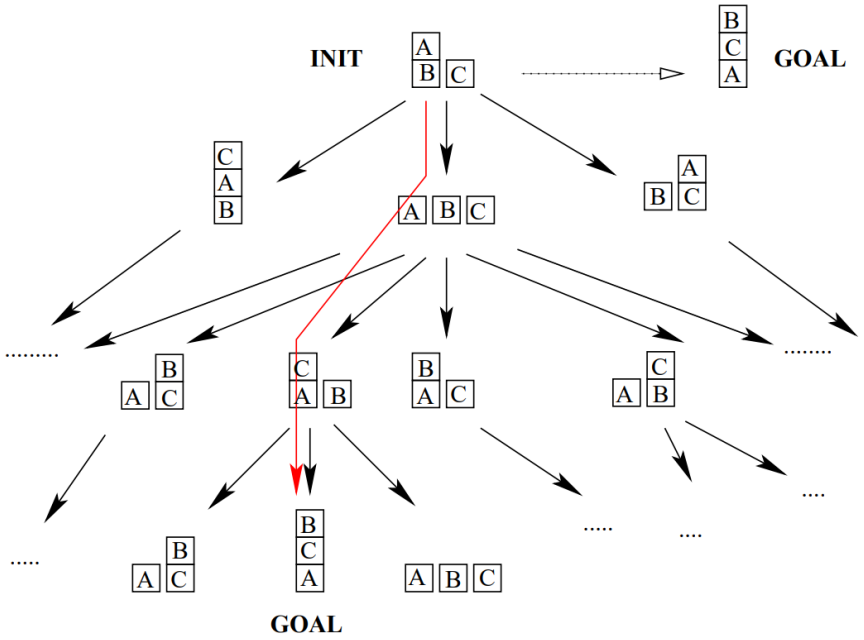


Blocks World

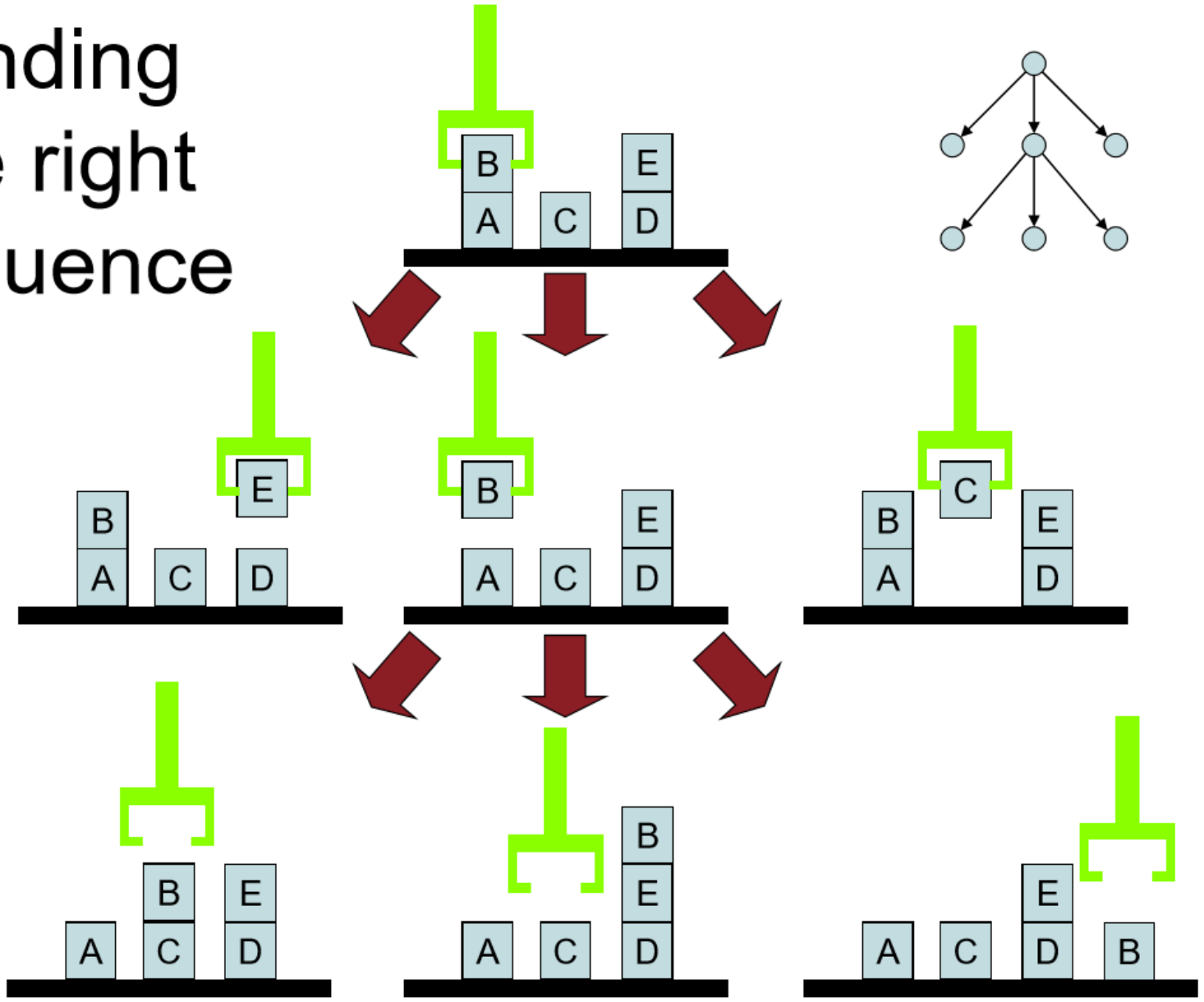


Given actions that move `clear' block to the table or onto another `clear' block,
find a plan to achieve the goal

How can we find a path in a graph whose size is exponential in the number of blocks (parameterized actions)?

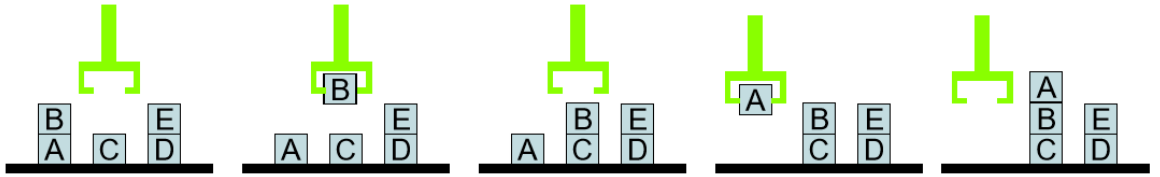


Finding the right Sequence



- Pickup(X)
  - P: grip(∅) ∧ clear(X) ∧ ontable(X)
  - A: grip(X)
  - D: onTable(X) ∧ grip(∅)
- PutDown(x)
  - P: grip(X)
  - A: onTable(X) ∧ grip(∅) ∧ clear(X)
  - D: grip(X)
- Stack(X,Y)
  - P: clear(Y) ∧ grip(X)
  - A: on(X,Y) ∧ grip(∅) ∧ clear(X)
  - D: clear(Y) ∧ grip(X)
- UnStack(X,Y)
  - P: clear(X) ∧ grip(∅) ∧ on(X,Y)
  - A: grip(X) ∧ clear(Y)
  - D: grip(∅) ∧ on(X,Y)

Update of Knowledge



- Grip(∅)
- On(B,A)
- Clear(B)
- OnTable(A)
- Clear(C)
- OnTable(C)
- On(E,D)
- Clear(E)
- OnTable(D)

UnStack(B,A)

- Grip(B)
- Clear(A)
- Clear(B)
- OnTable(A)
- Clear(C)
- OnTable(C)
- On(E,D)
- Clear(E)
- OnTable(D)

Stack(B,C)

- Grip(∅)
- Clear(A)
- Clear(B)
- OnTable(A)
- On(B,C)
- OnTable(C)
- On(E,D)
- Clear(E)
- OnTable(D)

Pickup(A)

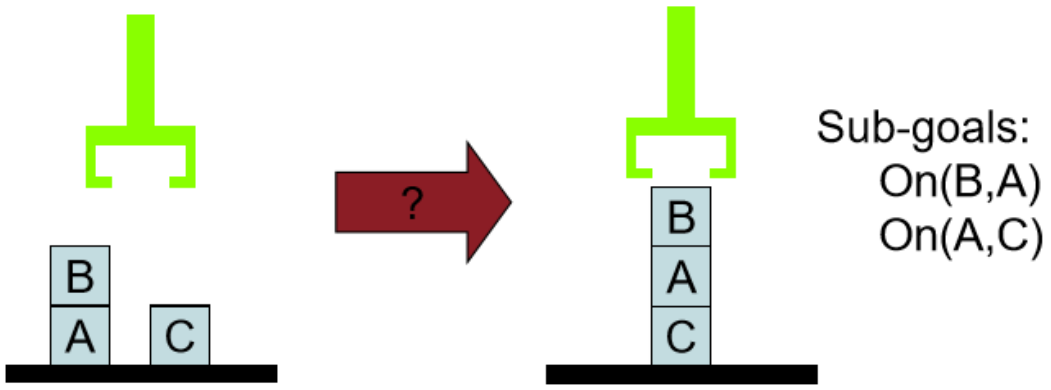
- Grip(A)
- Clear(A)
- Clear(B)
- ~~OnTable(A)~~
- On(B,C)
- OnTable(C)
- On(E,D)
- Clear(E)
- OnTable(D)

Stack(A,B)

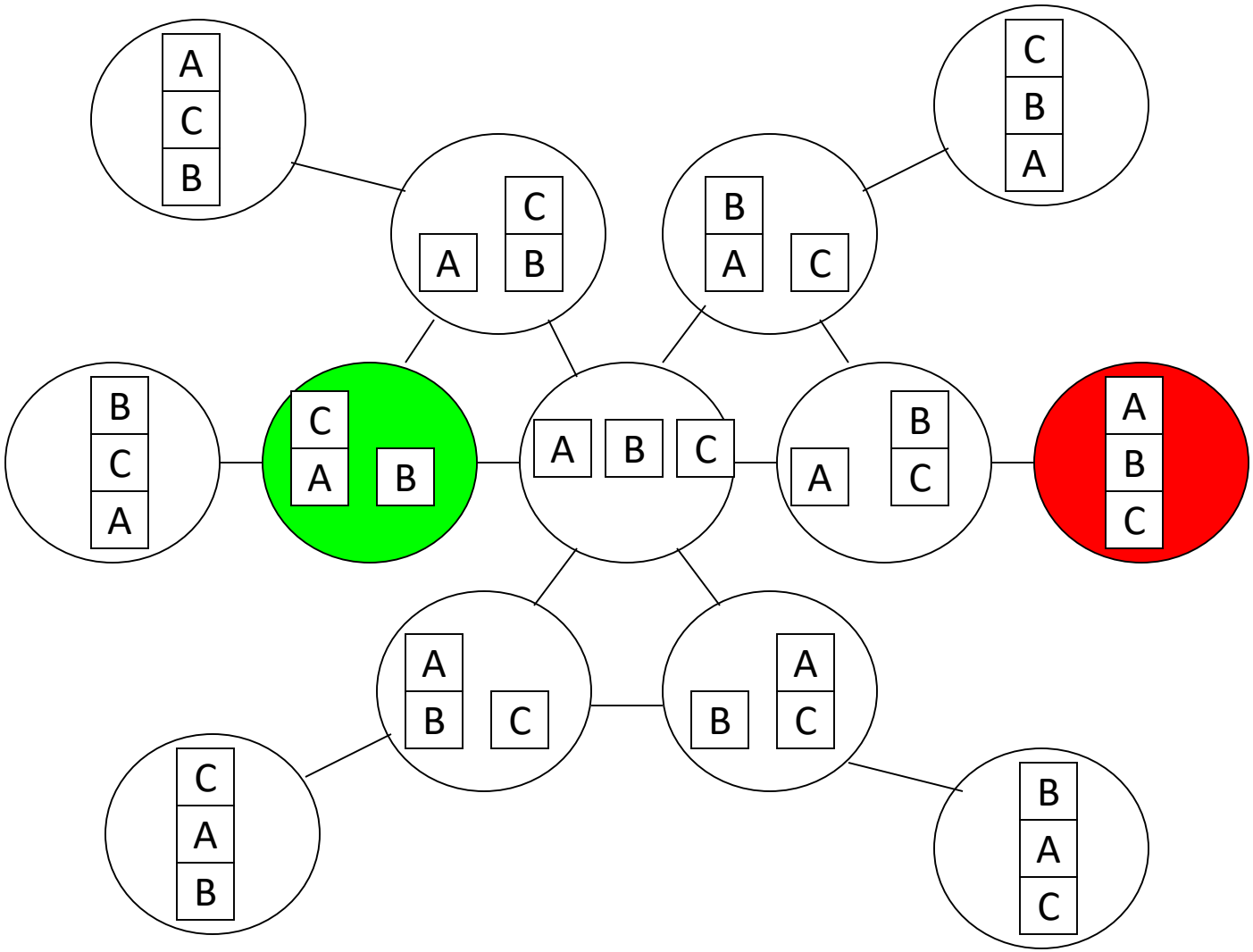
- Grip(∅)
- Clear(A)
- ~~Clear(B)~~
- On(A,B)
- On(B,C)
- OnTable(C)
- On(E,D)
- Clear(E)
- OnTable(D)

(Minor) Problems

- Exponential sequence of moves
- Does the tree have a fixed depth?
  - Cycles
- Incompatible sub-goals

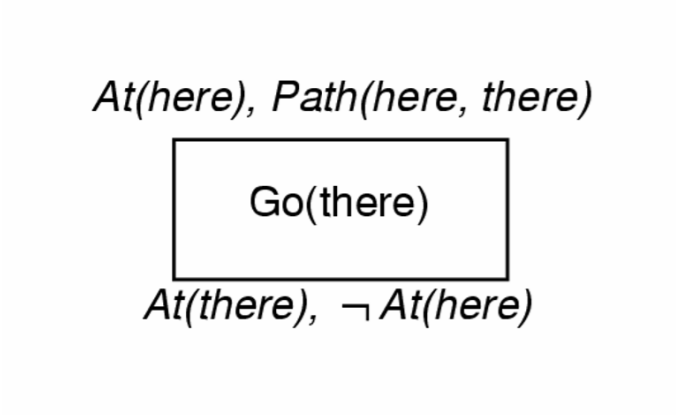


Using A* in Planning



h(n) = “# of goals remaining to be satisfied” g(n) = “# of steps so far”

STRIPS Operators Revisited



- Operators are triplets of descriptors
  - A set of Preconditions
  - A set of Additions
  - A set of Deletions
- Graphical Representation

Predicates

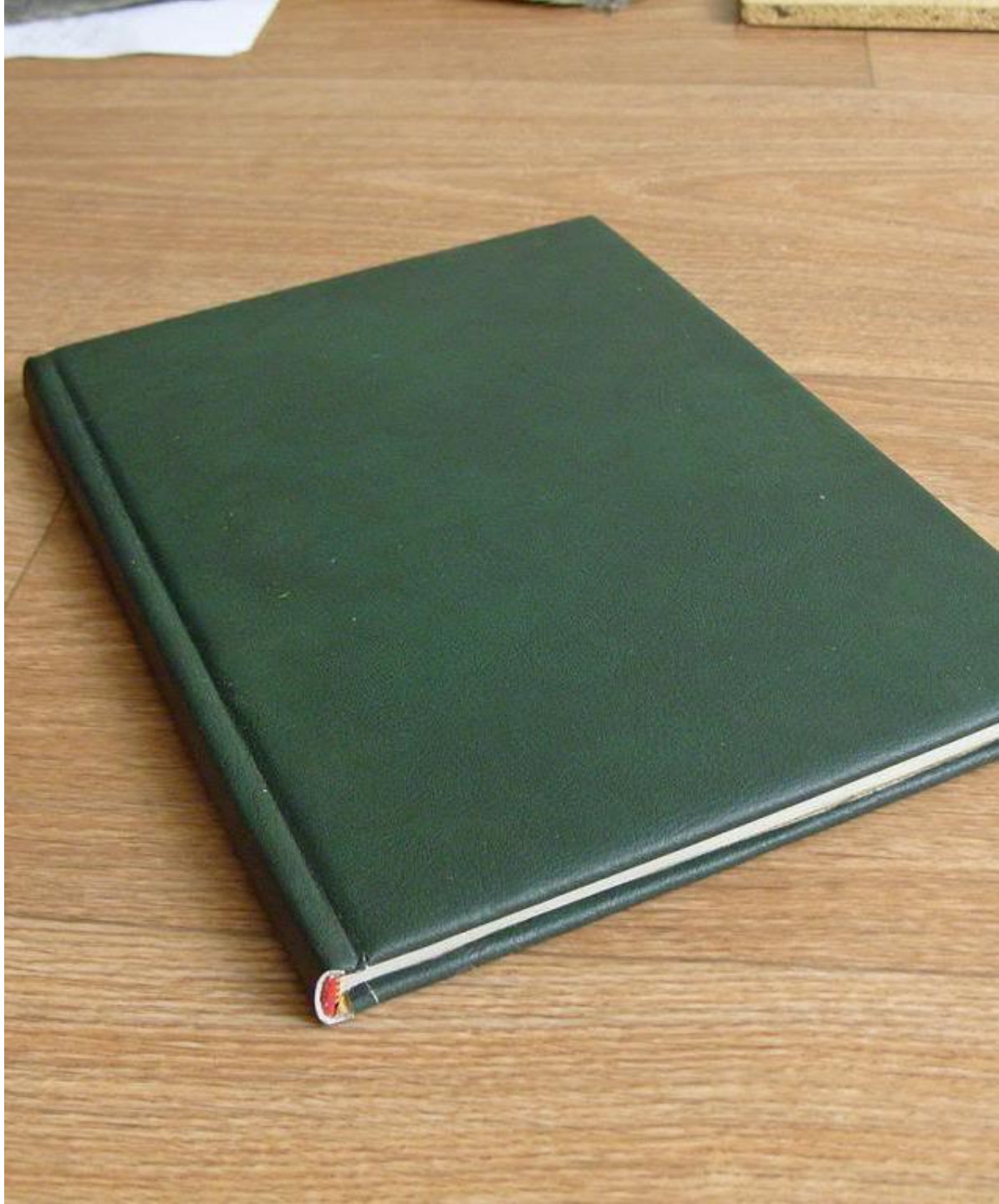
Two types of predicates:

Direct measurements of state

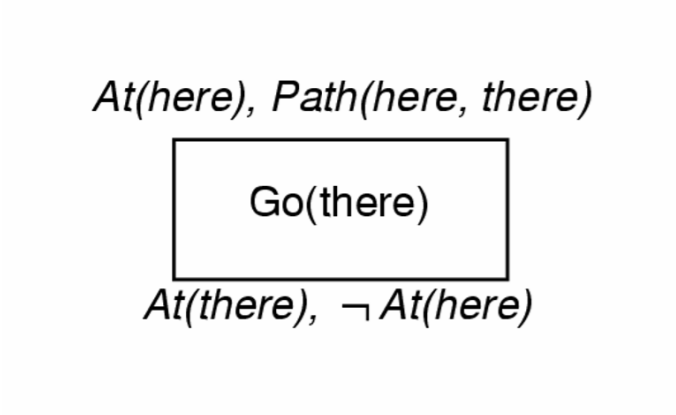
- *At(object,x,y,z):*
  At(book_0,0,0,1.6)

Indirect from state variable

- *On_top(object,object):*
  On_top(book, table)



STRIPS Operators Revisited

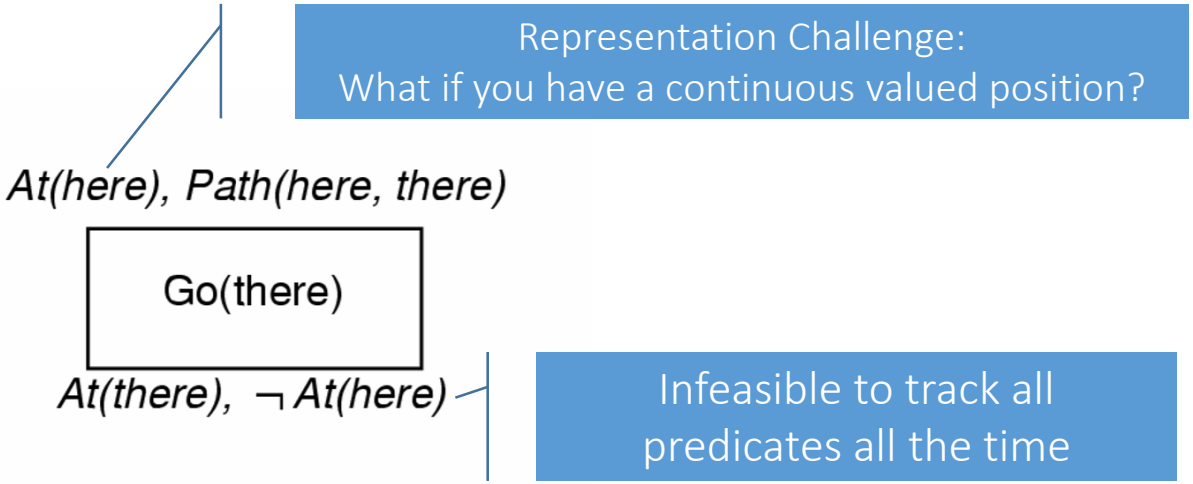


- Operators are triplets of descriptors
  - A set of Preconditions
  - A set of Additions
  - A set of Deletions
- Graphical Representation

Set predicate values in ranges [-1,0) or (0,1]

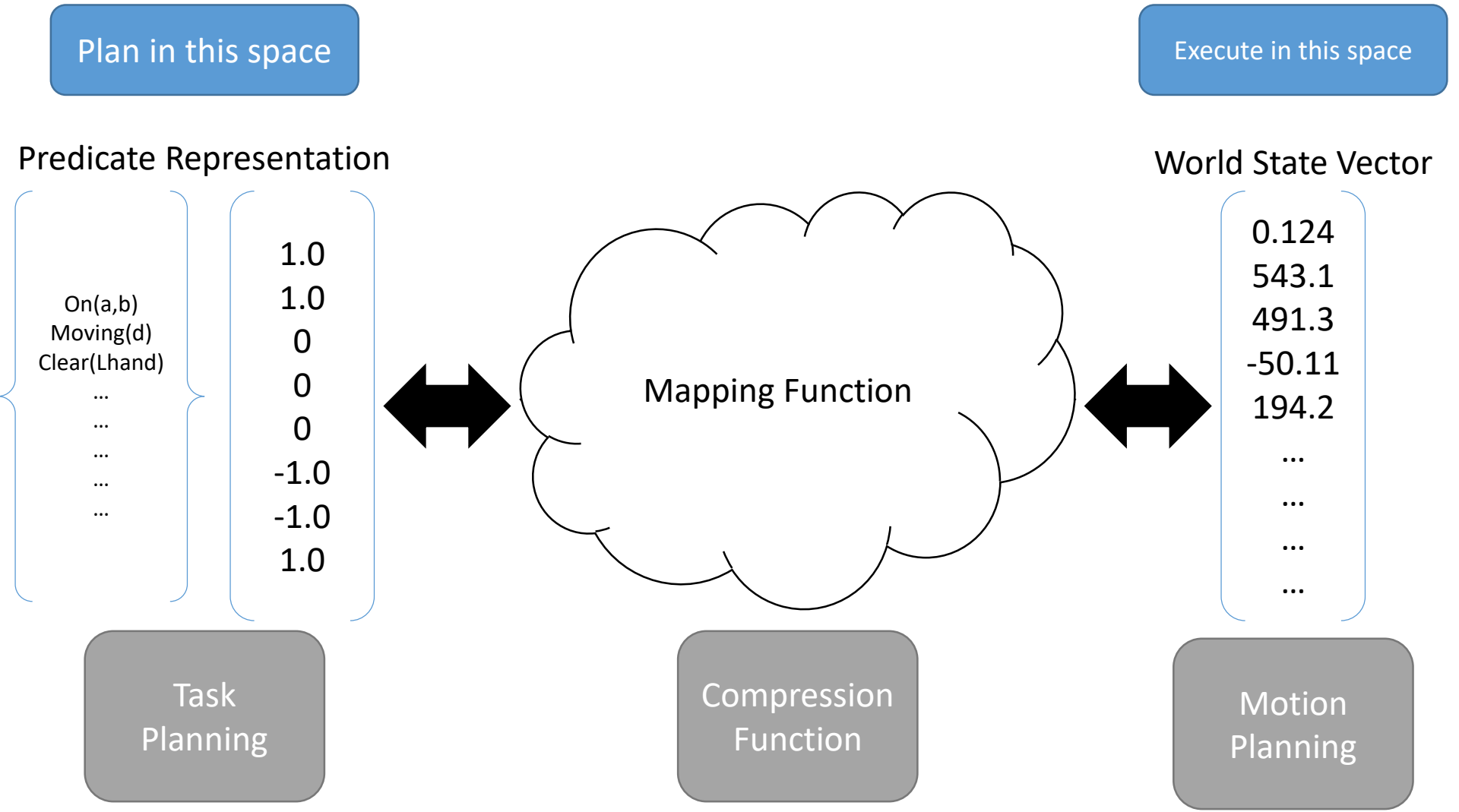
Predicates are ternary, 0 is “doesn’t exist”

STRIPS Operators Revisited

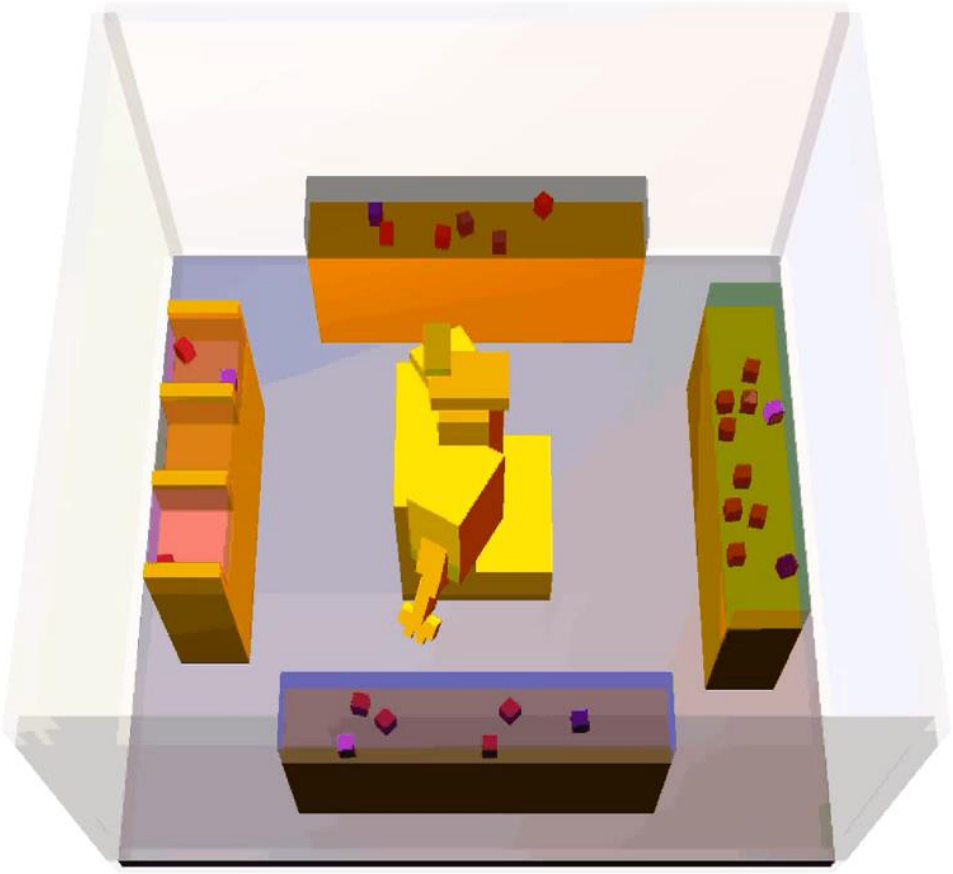


- Operators are triplets of descriptors
  - A set of Preconditions
  - A set of Additions
  - A set of Deletions
- Graphical Representation

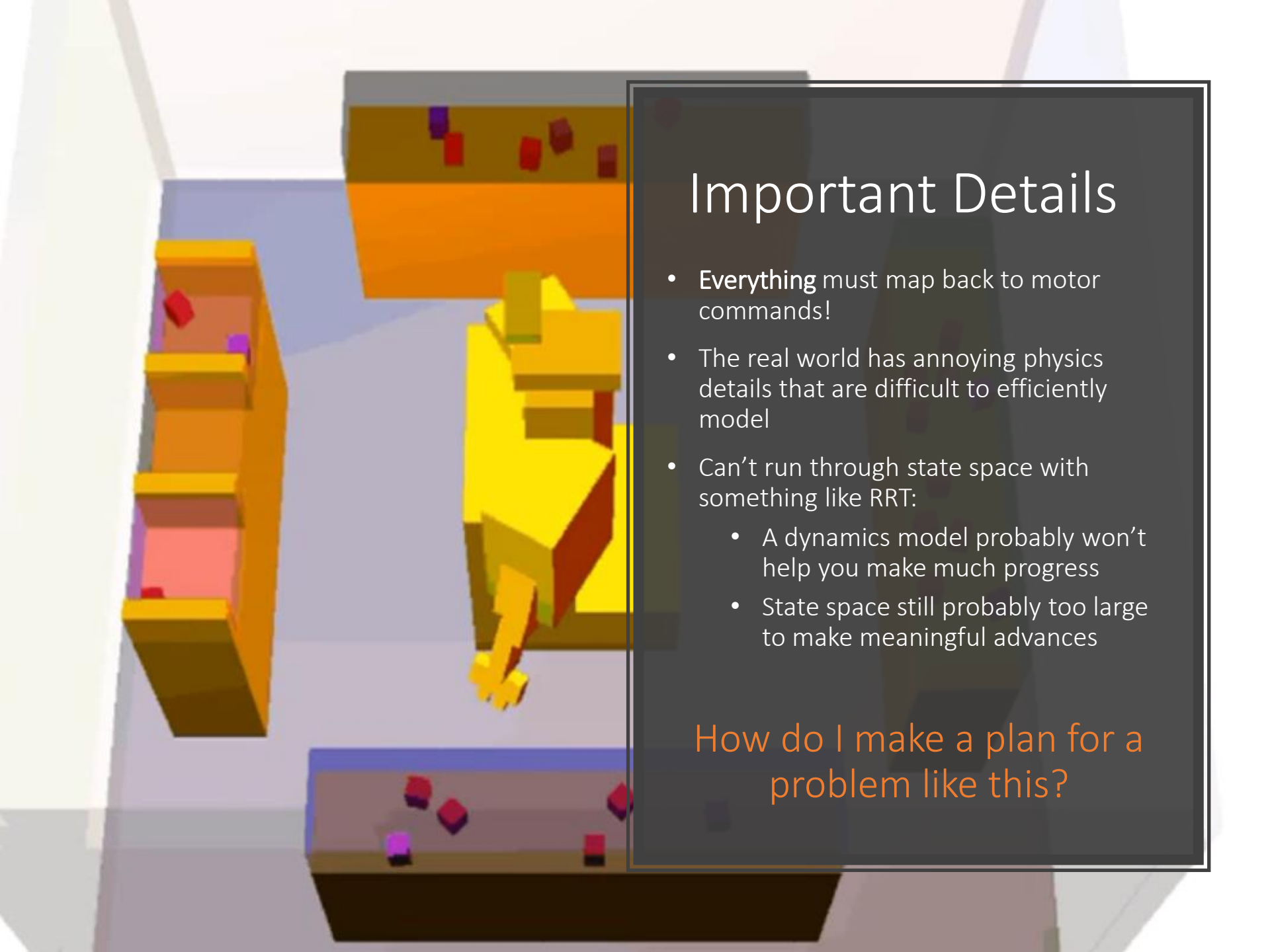
Lifted vs. Grounded Planning



Choosing the right granularity is not always easy



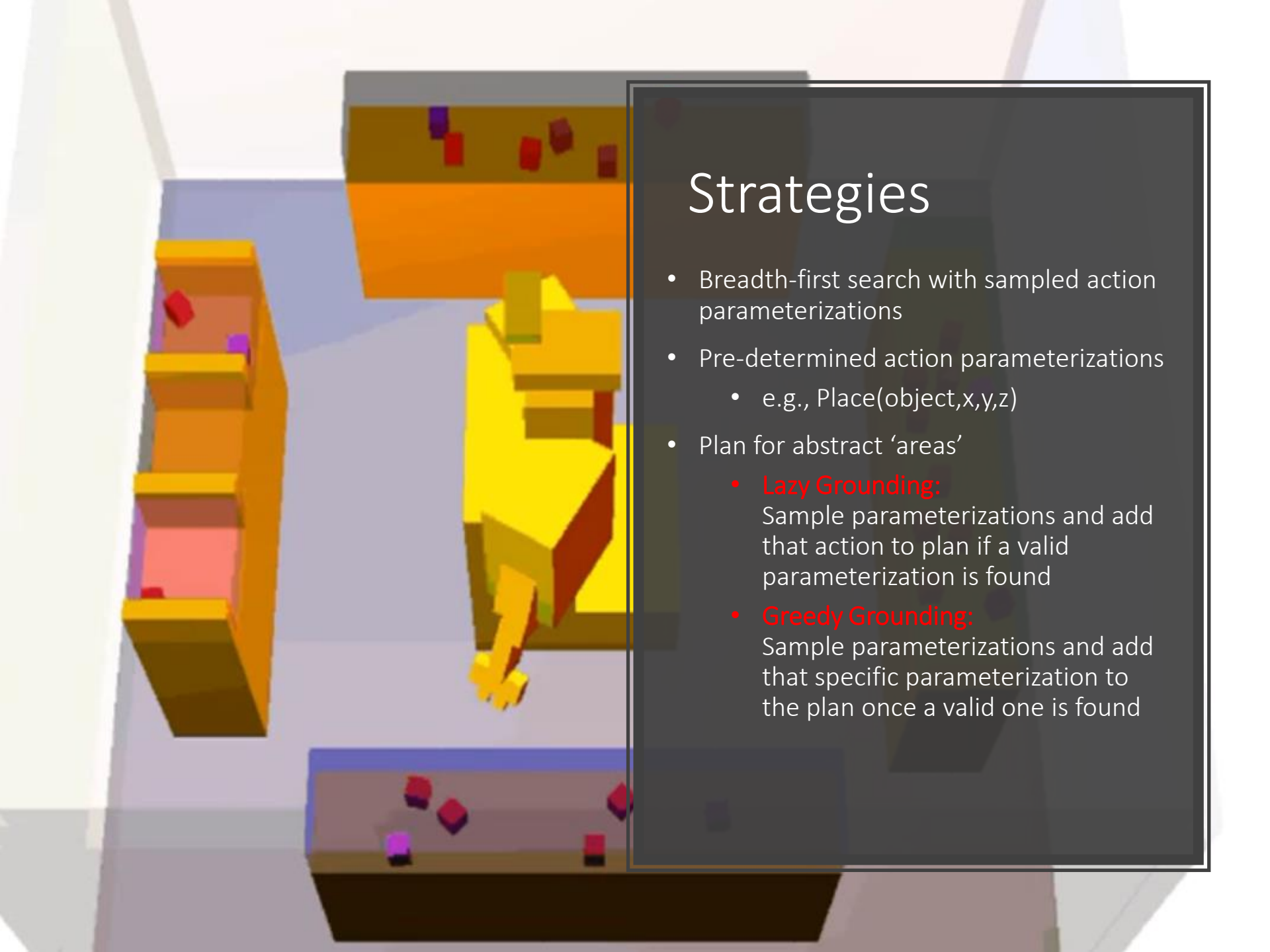
(source: FFRob -- http://web.mit.edu/caelan/www/research/ffrob/)



Important Details

- **Everything** must map back to motor commands!
- The real world has annoying physics details that are difficult to efficiently model
- Can't run through state space with something like RRT:
  - A dynamics model probably won't help you make much progress
  - State space still probably too large to make meaningful advances

How do I make a plan for a problem like this?

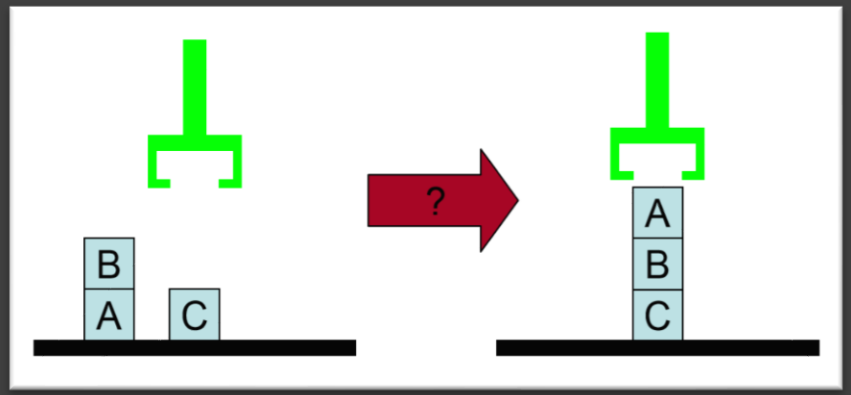


Strategies

- Breadth-first search with sampled action parameterizations
- Pre-determined action parameterizations
  - e.g., Place(object,x,y,z)
- Plan for abstract ‘areas’
  - Lazy Grounding:
    Sample parameterizations and add that action to plan if a valid parameterization is found
  - Greedy Grounding:
    Sample parameterizations and add that specific parameterization to the plan once a valid one is found

Non-Interleaved Planning

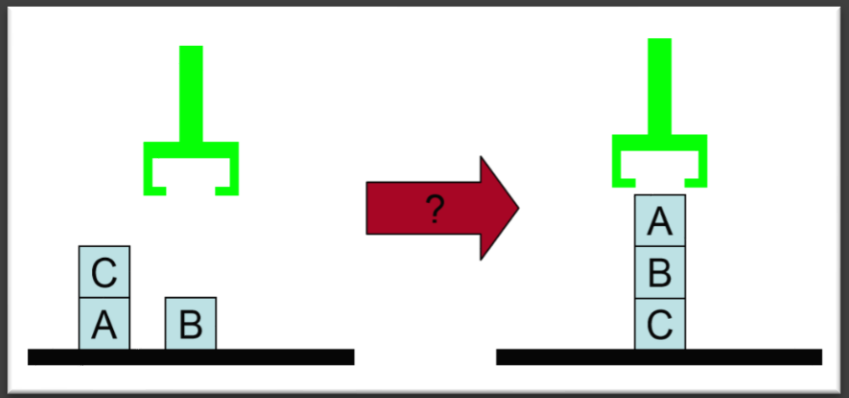
- Most parts of the world are independent, facilitating divide-and-conquer strategies
  - Plan to achieve individual/subsets of final goal state's predicate objectives



- Goal: On(B,C) ∧ On(A,B)
- Achieve goal #1, then achieve goal #2

Problems with Non-interleaved Planners: **Sussman Anomaly**

- Switch the location of B & C in the initial state
- **Same goal: On(B,C) ∧ On(A,B)**



- Achieving goals sequentially doesn't work: **On(B,C)** is violated when achieving **On(A,B)**.

Creating a Lifted Plan

Naïve strategy:

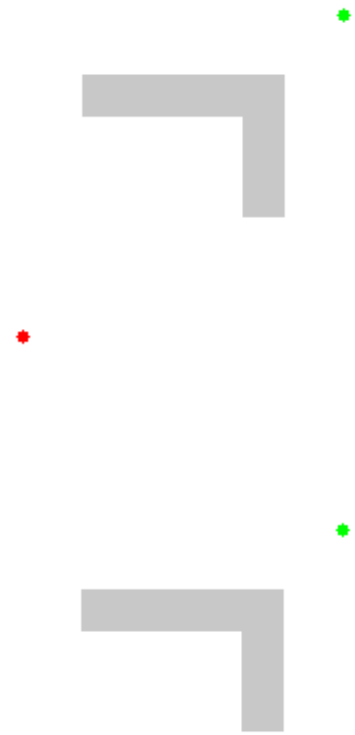
- Breadth-first search over actions
- Pre-determined max search depth

Better strategy:

- Employ planning heuristics to inform action selection and generate sub-goal orderings

Key Problem: Branching Factor

A linear increase in search depth requires an exponential increase in visited states



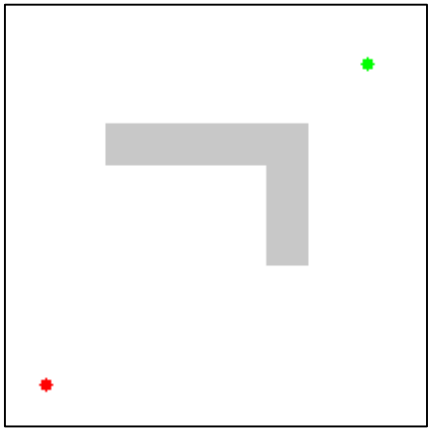
Heuristics are required to help guide action selection:

Our search algorithm needs a notion of direction!

A* Search Algorithm

Adds heuristic to Dijkstra's Algorithm to bias exploration.

Finds shortest path between two nodes in a graph



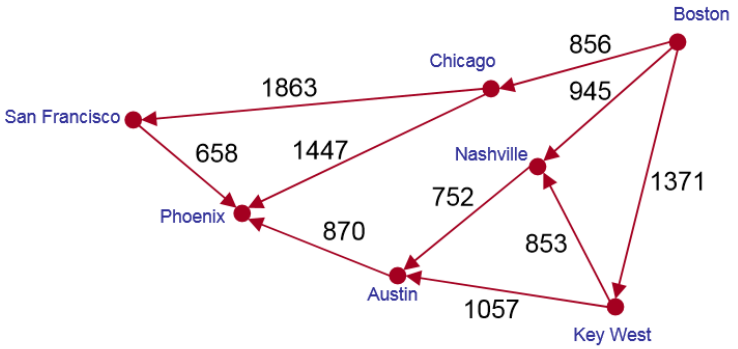
Insight

$$f(n) = g(n) + h(n)$$

$f(n)$ = Cost so far + Est. cost to go

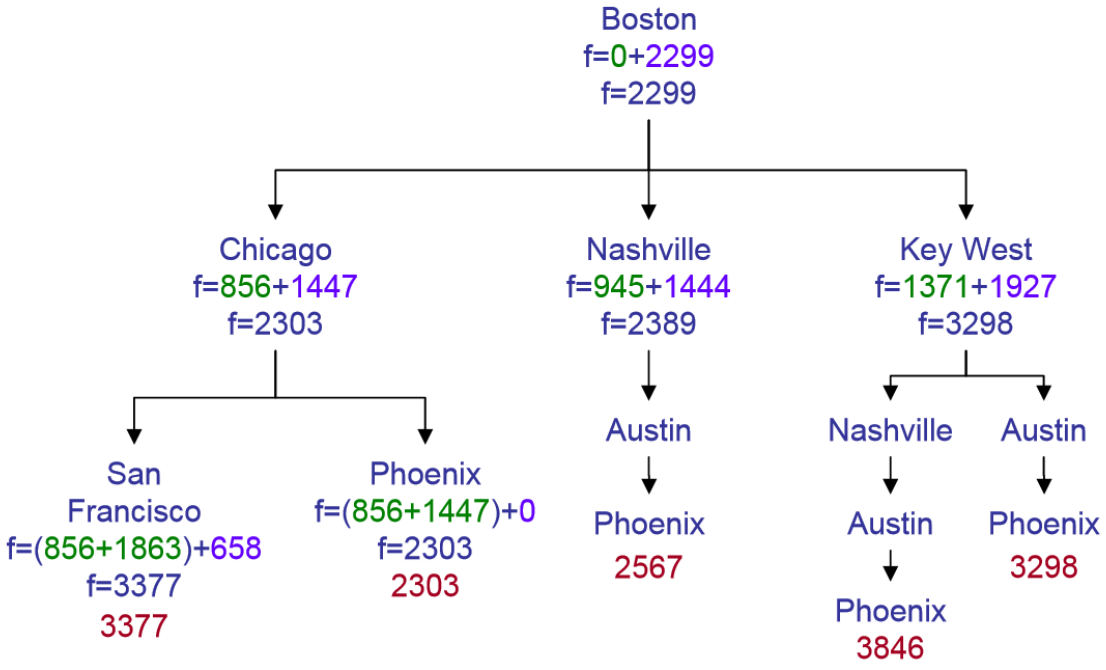
Admissible Heuristic: An underestimate of the shortest possible path from node n to the goal.

A* Search Algorithm



- Combine Greedy search with Uniform Cost Search
- Minimize the total path cost (f) =
  actual path so far (g) +
  estimate of future path to goal (h)

	Distance to Phoenix
Boston	2299
Chicago	1447
Nashville	1444
Key West	1927
Austin	870
San Francisco	658



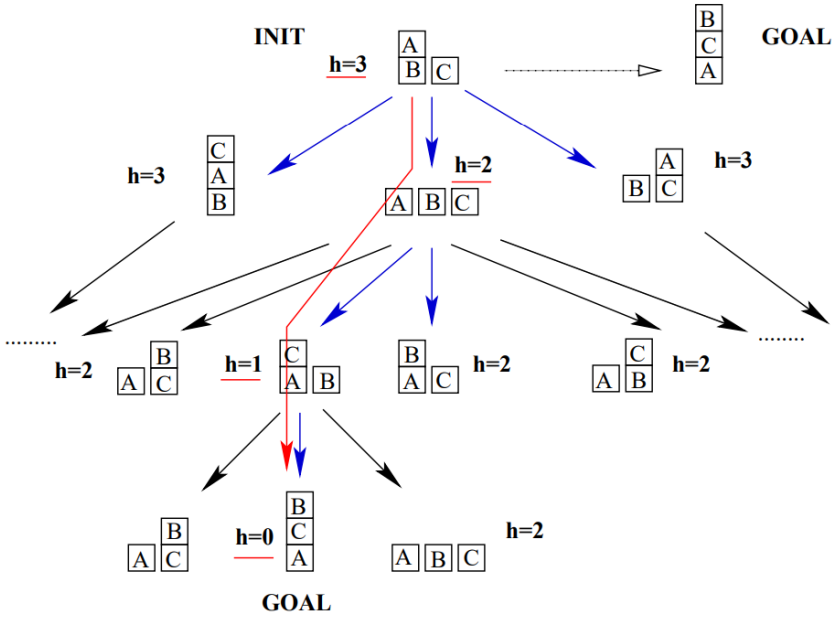
Total Distance Flown

Developing Heuristics

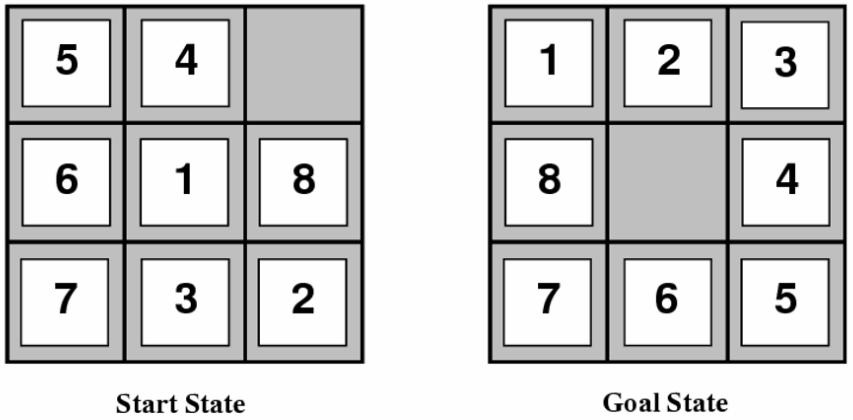
- **Heuristic evaluations** h(s) provide 'sense-of-direction'

- Derived efficiently in a **domain-independent** fashion from **relaxations** where effects made **monotonic** (delete relaxation).

- **"Fast-Forward" Heuristic:** Remove all deletions from operators



Heuristic Selection



- Must be admissible (never over-estimate)
- Heuristics for the 8-Puzzle

Heuristic Selection Effects

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

- Compare iterative-deepening with A* using *h1* (# misplaced tiles) and *h2* (city block distance)
- Effective branching factor b*
  - Number of expanded nodes = $1 + b^* + (b^*)^2 + \ldots + (b^*)^{depth}$
  - b* remains relatively constant across many measurements

Planning Heuristics

- FastForward Heuristic
  - Remove deletions from all operators and run planner
  - Resulting plan gives a heuristic for # steps from current state to goal
  - FF Plan can provide ordering constraints on actions, even within partially specified plans
- Learned Cost Function
  - Learning Real-Time A* (LRTA*)
  - Update heuristic dynamically: Value function V(s)
  - Connection back to reinforcement learning: **V(s) and Q(s,a)** for search heuristic

FastForward Heuristic

- Modifying the formulation to be easier is called a **relaxation** of the original problem.
- **FastForward** uses a **delete-relaxation** to transform the original problem P into P^+ where the operators (actions) have no deletions
- Solving P^+ optimally should be much easier than solving P optimally!
  - It isn't. These are both NP-hard ☹
- But finding **any solution** is a lot easier!

Learning Real-Time A*

- Hill-climbing algorithm where the **best child node** is selected and others **discarded**, and the heuristic is updated dynamically.

Cost paid already	Cost to goal

- A* Cost Formula: $f(n) = g(n) + h(n)$
  - Replace $h(n)$ with $V(s')$
  - Replace $g(n)$ with $Q(s, a)$ where a has high $T(s, a, s')$
- Q(s,a) = c(s,a) + V(s')

Admissible Heuristics and Pruning Strategies

Admissible heuristics from delete-relaxation:

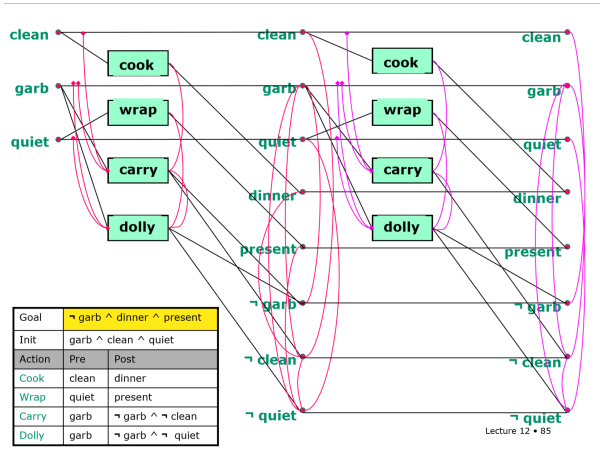
- h_{max} = i iff goal g is reachable in i steps from s
  - Admissible because of GRAPHPLAN Algorithm
- h_{FF} = number of **different** actions in $\pi(g)$

Search Pruning Strategies:

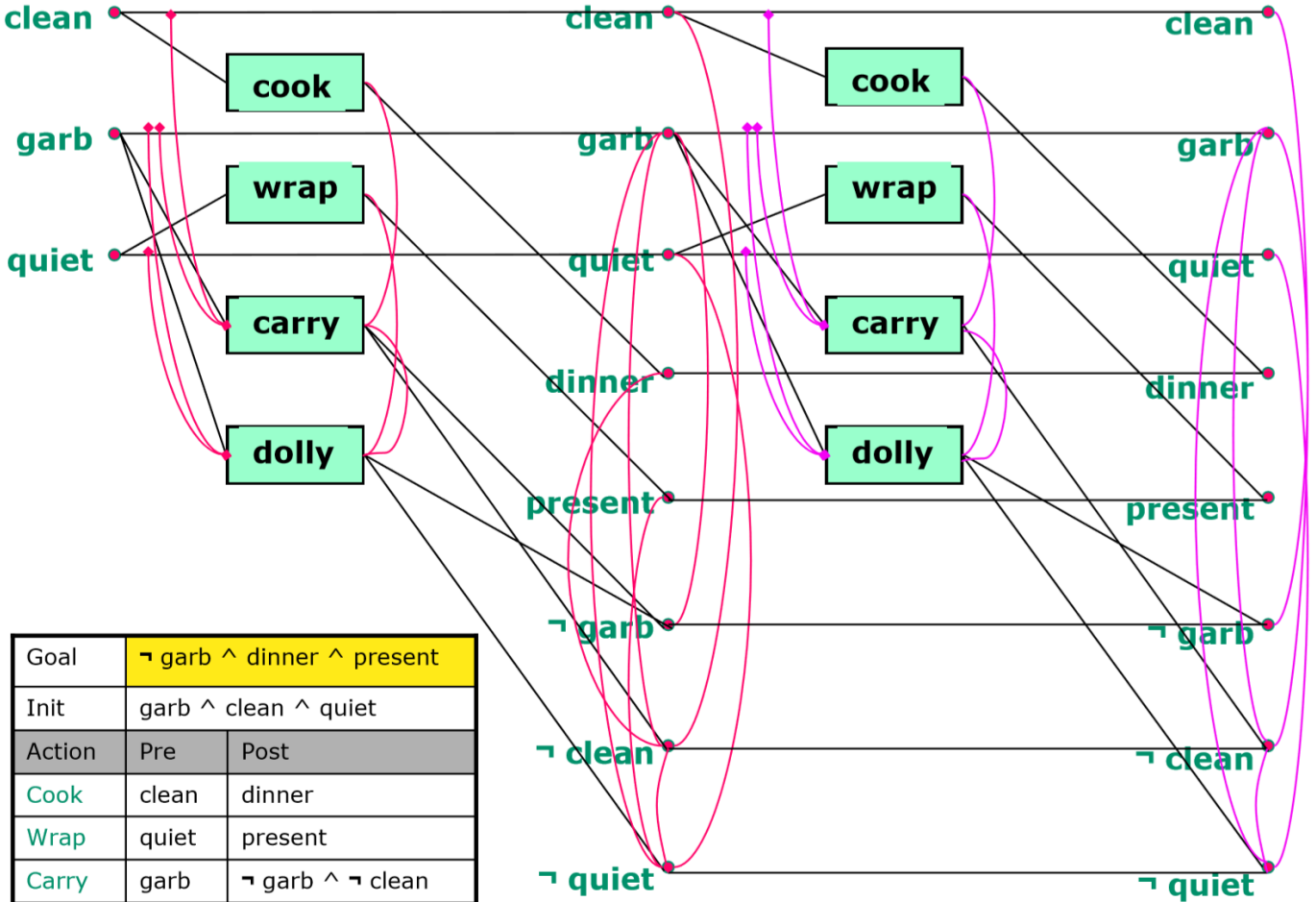
- Novelty: Assign each state $novelty(s) = i$ if i predicates take on a value never seen in path from s_0 to s
- IW-search (Iterative Width): Ignore any nodes at the search frontier with $novelty(s) > i$.
  - Call planner for increasing i until problem solved or i exceeds number of variables in the problem.

GRAPHPLAN for h_{max}

- Alternating layers of State and Action in leveled graph
- Preconditions connect predicates in layer $2t$ to actions in layer $2t + 1$
- Effects connect actions to predicates in next layer
- All predicates in layer $2t$ connected to themselves in layer $2(t + 1)$ by “maintenance actions”
- First layer where goal predicates exist is lower bound
- Need to compute MUTEX actions and predicates



GRAPHPLAN Graph



Goal	¬ garb ^ dinner ^ present	
Init	garb ^ clean ^ quiet	
Action	Pre	Post
Cook	clean	dinner
Wrap	quiet	present
Carry	garb	¬ garb ^ ¬ clean
Dolly	garb	¬ garb ^ ¬ quiet

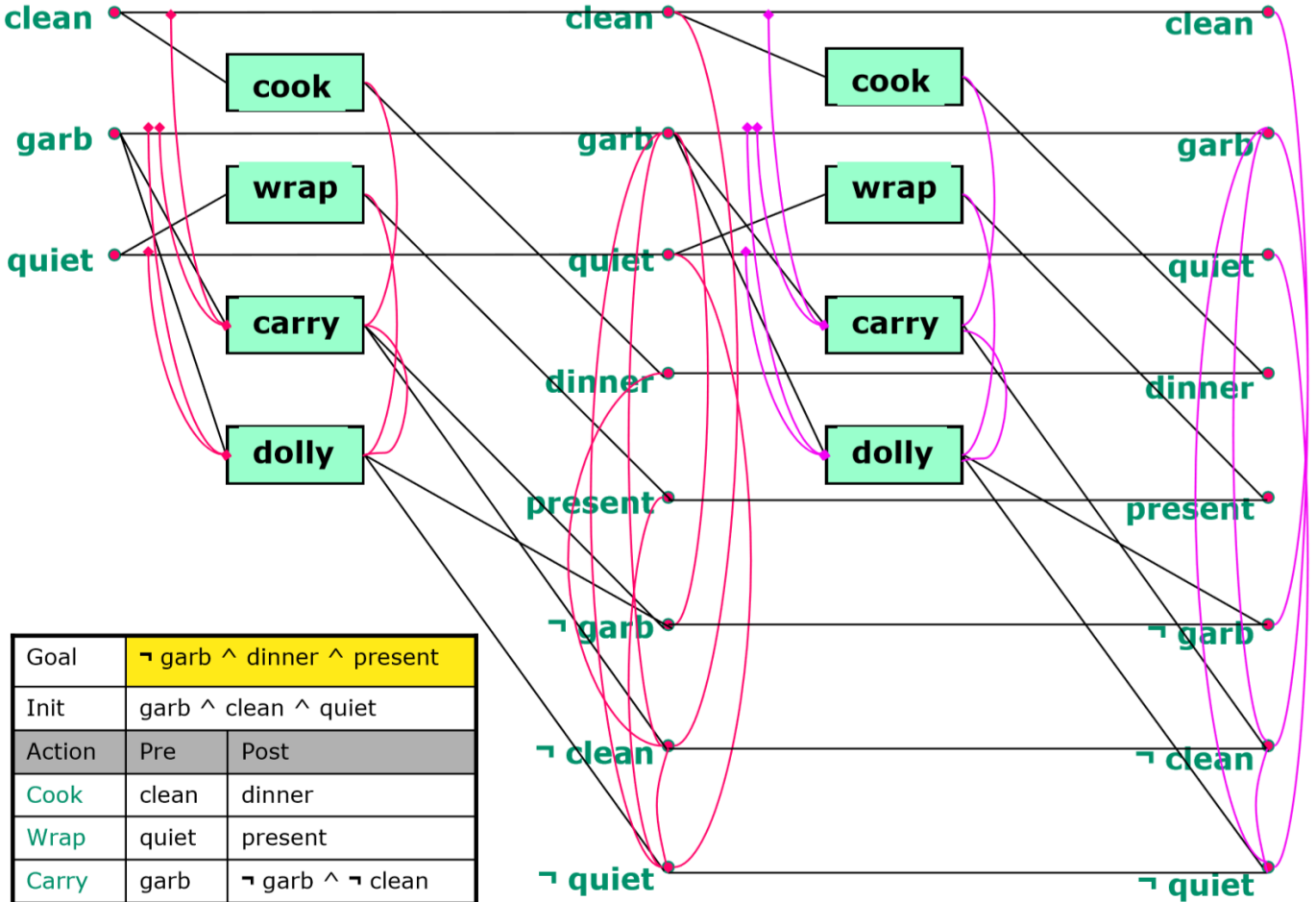
GRAPHPLAN MUTEX

- Actions
  - **Inconsistent**: Effect of one action is negation of other's effect
  - **Interference**: Action deletes precondition of other
  - **Competing Needs**: Two actions have mutex preconditions at previous level
- Predicates
  - **Negation**: Two predicates are negations of each other
  - **Inconsistent Support**: All means of achieving the pair are pairwise mutex

Sample Problem: Birthday Dinner

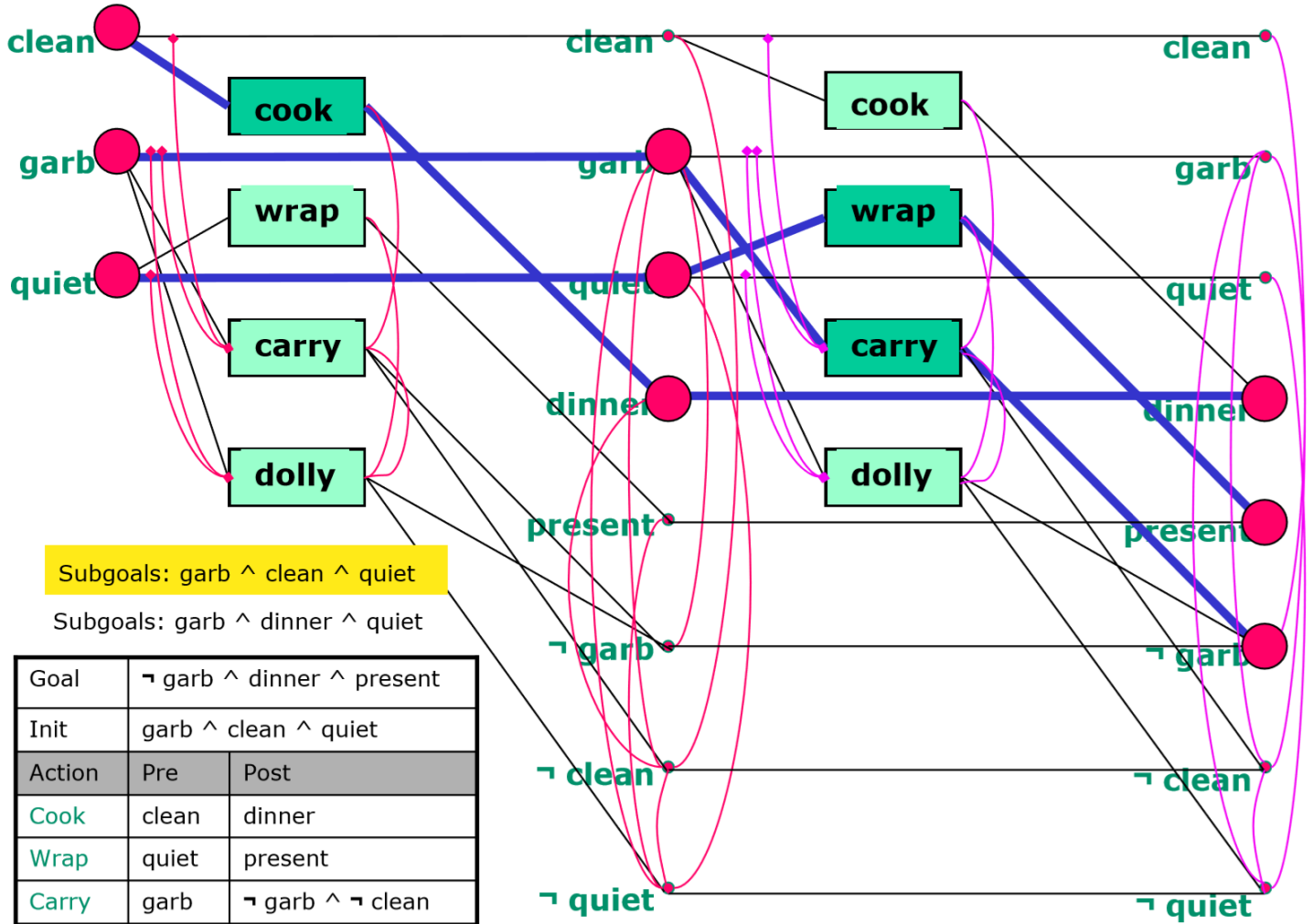
- Goal: ~Garbage & Dinner & Present
- Init: Garbage & Clean & Quiet
- Actions
  - Cook
    - Pre: clean
    - Effect: dinner
  - Wrap
    - Pre: quiet
    - Effect: present
  - Carry
    - Pre: garbage
    - Effect: ~Garbage & ~Clean
  - Dolly
    - Pre: garbage
    - Effect: ~garbage & ~quiet

GRAPHPLAN Graph



Goal	¬ garb ^ dinner ^ present	
Init	garb ^ clean ^ quiet	
Action	Pre	Post
Cook	clean	dinner
Wrap	quiet	present
Carry	garb	¬ garb ^ ¬ clean
Dolly	garb	¬ garb ^ ¬ quiet

GRAPHPLAN Graph Solution



Goal	¬ garb ^ dinner ^ present	
Init	garb ^ clean ^ quiet	
Action	Pre	Post
Cook	clean	dinner
Wrap	quiet	present
Carry	garb	¬ garb ^ ¬ clean
Dolly	garb	¬ garb ^ ¬ quiet