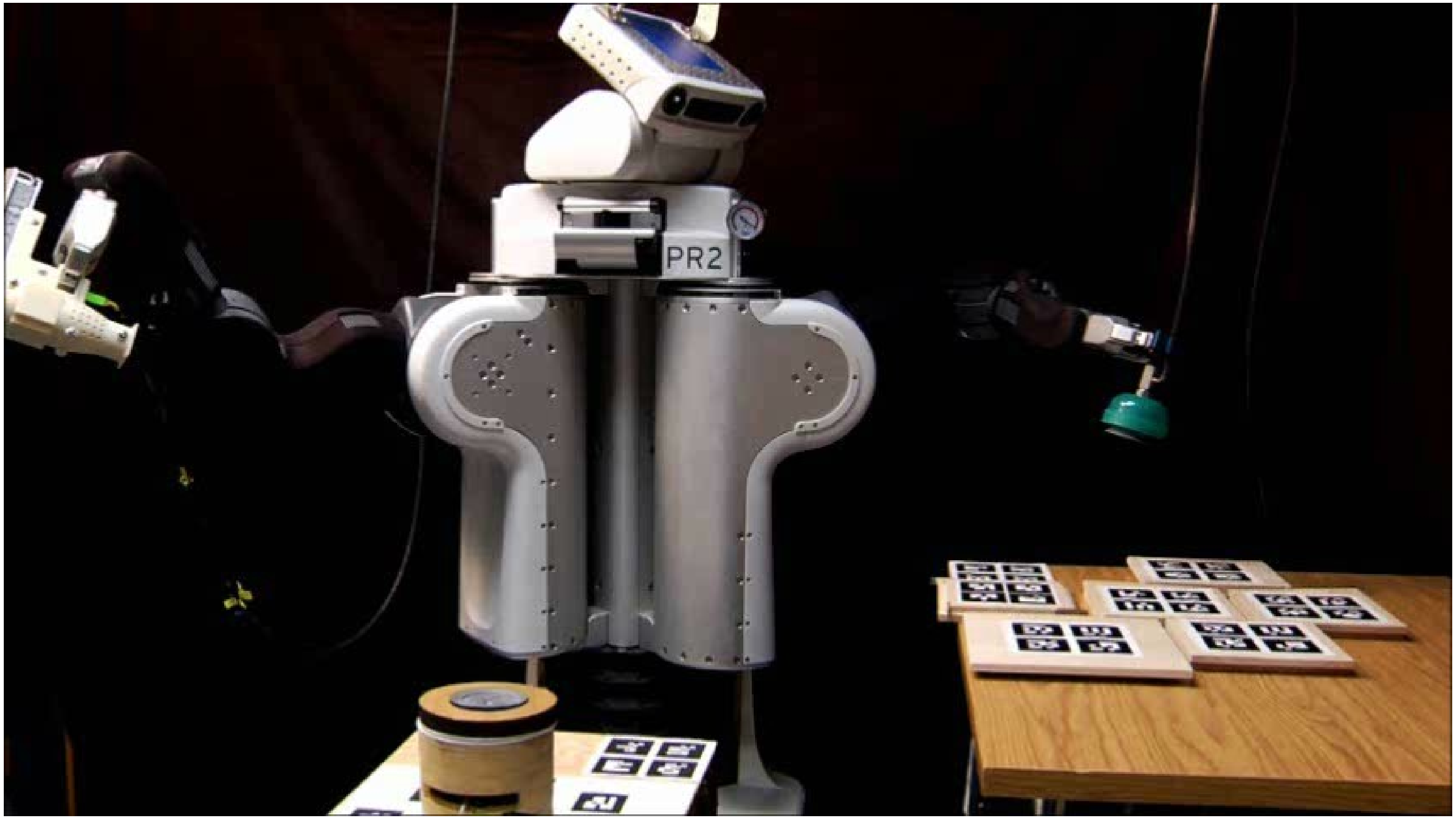
16-782
Planning & Decision-making in Robotics

Planning Representations:
Symbolic Representation for Task Planning

Maxim Likhachev
Robotics Institute
Carnegie Mellon University

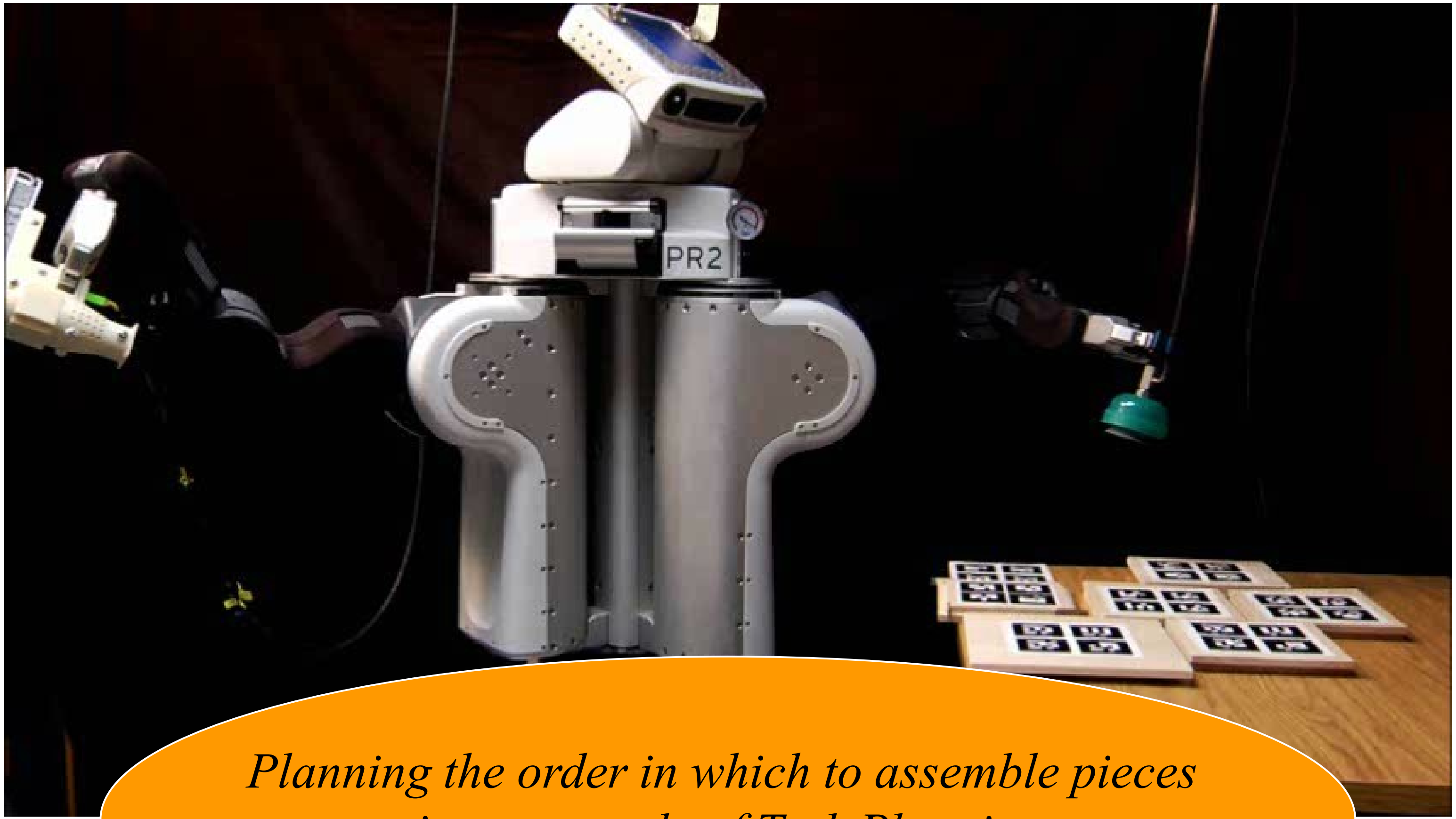
Planning to Construct a Birdcage

- Robot takes in a 3D model of a birdcage it needs to build



Planning to Construct a Birdcage

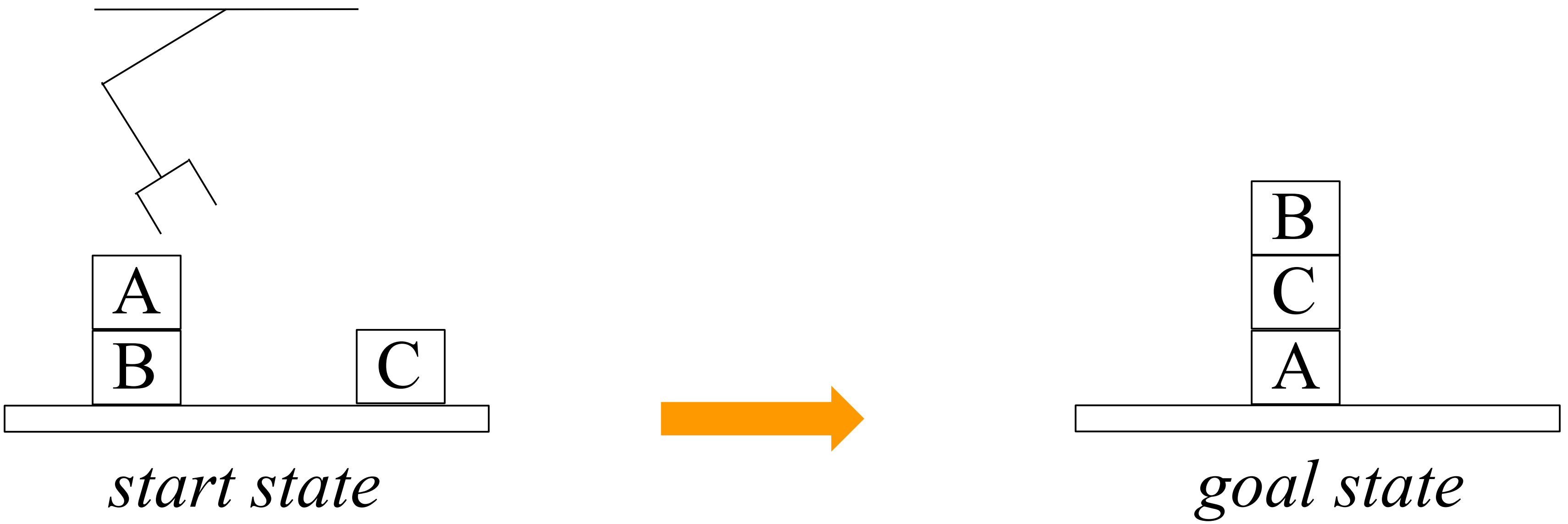
- Robot takes in a 3D model of a birdcage it needs to build



Planning the order in which to assemble pieces is an example of Task Planning

Famous "Blocksworld" Example

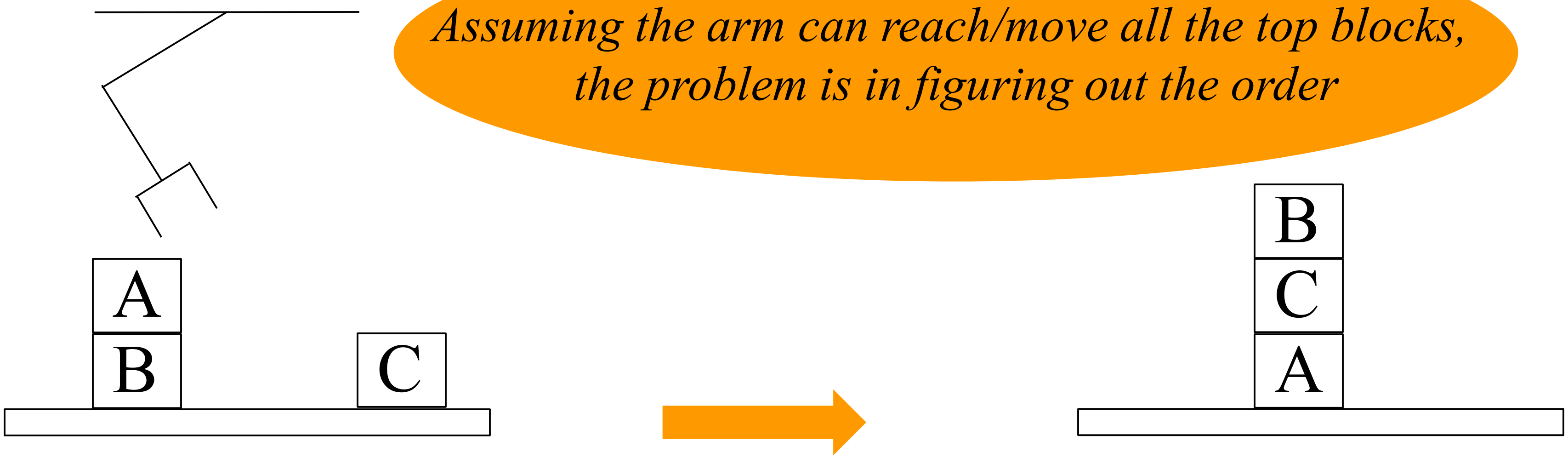
- Planning to re-order the blocks



Famous "Blocksworld" Example

- Planning to re-order the blocks

Assuming the arm can reach/move all the top blocks, the problem is in figuring out the order



start state *goal state*

Famous “Blocksworld” Example

- Planning to re-order the blocks

Actions:

Move(b,x,y) – moves block b from x to y

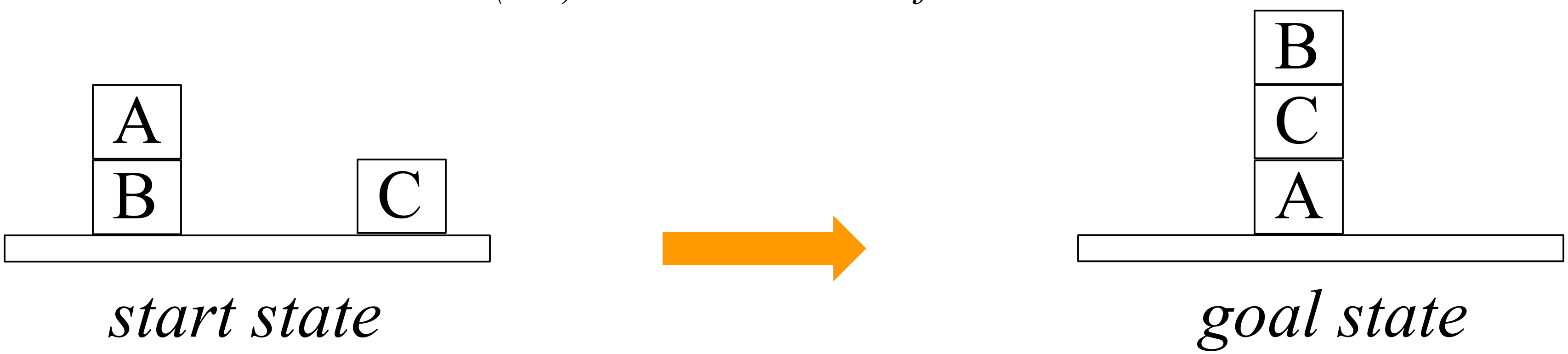
MoveToTable(b,x) – moves block b from x to table y

Famous "Blocksworld" Example

- Planning to re-order the blocks

Actions:

Move(b,x,y) – moves block b from x to y
MoveToTable(b,x) – moves block b from x to table



What is a plan that achieves the goal?

Defining it as a Graph Search (State-space Search)

- Planning to re-order the blocks

Actions:

Move(b,x,y) – moves block b from x to y
MoveToTable(b,x) – moves block b from x to table

start state *goal state*

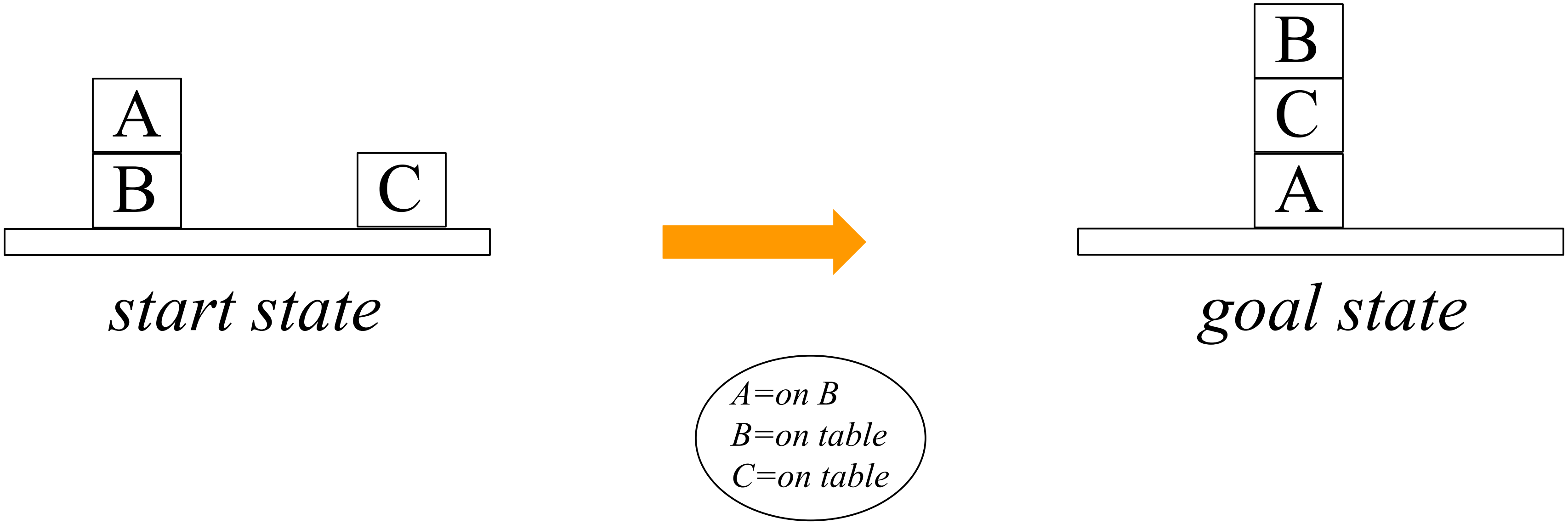
Any ideas for how to represent a state in a graph?

Defining it as a Graph Search (State-space Search)

- Planning to re-order the blocks

Actions:

Move(b,x,y) – moves block b from x to y
MoveToTable(b,x) – moves block b from x to table



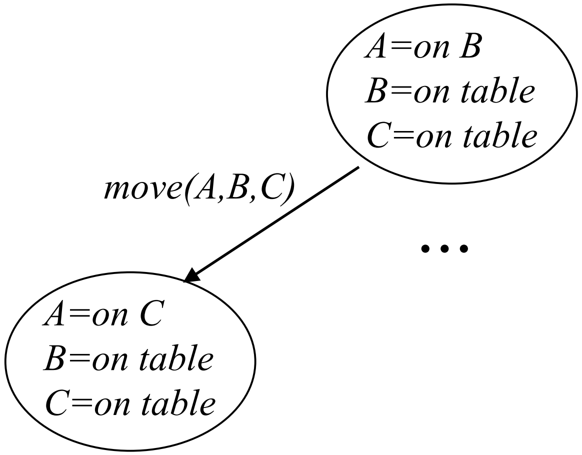
Defining it as a Graph Search (State-space Search)

- Planning to re-order the blocks

Actions:

Move(b,x,y) – moves block b from x to y

MoveToTable(b,x) – moves block b from x to table

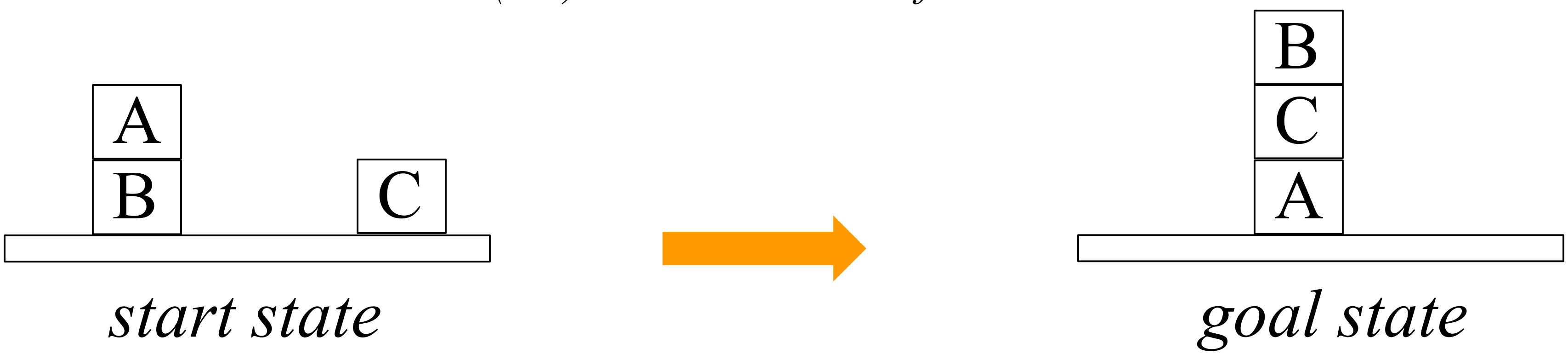


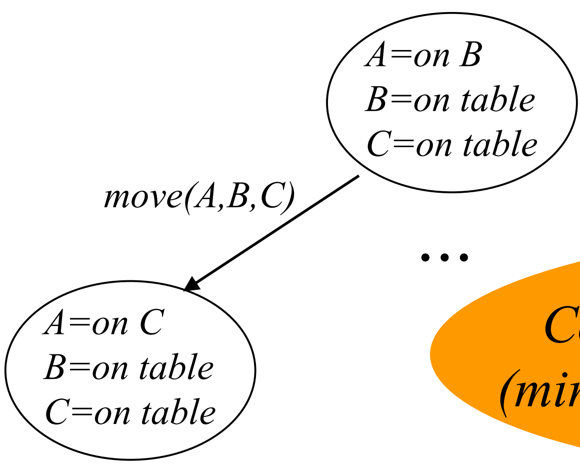
Defining it as a Graph Search (State-space Search)

- Planning to re-order the blocks

Actions:

Move(b,x,y) – moves block b from x to y
MoveToTable(b,x) – moves block b from x to table





Cost of each edge is often set to 1 (minimization of the total # of actions)

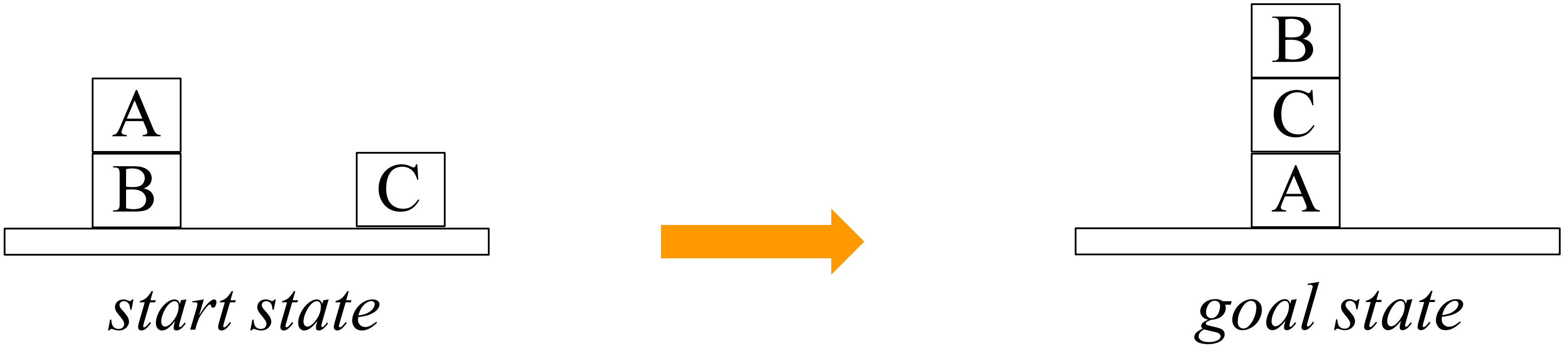
Defining it as a Graph Search (State-space Search)

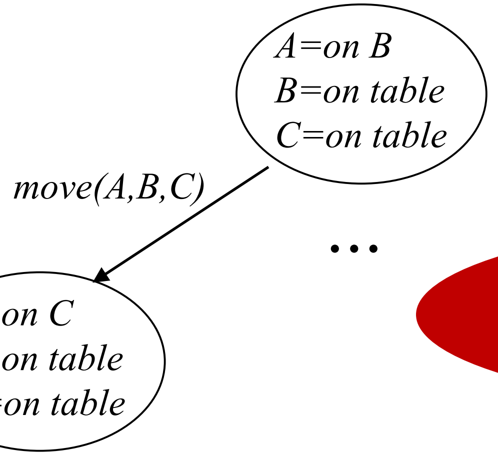
- Planning to re-order the blocks

Actions:

Move(b,x,y) – moves block b from x to y

MoveToTable(b,x) – moves block b from x to table





Any ideas for heuristics?

Generic Representation of Symbolic Planning Problems

We would like to be able to represent ANY planning problem
with a single representational language that allows for the definition of:
STATES, ACTIONS, GOAL

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

Goal Representation:

Action Representation:

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals [redacted]

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

Action Representation:

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals [redacted]

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Repr

***Closed-world assumption**:*
any conditions not mentioned in the state are assumed to be false

Action Representation:

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals

[redacted]

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

desired conjunction of positive(true) literals

[redacted]

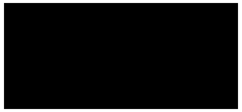
Action Representation:

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals



(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

desired conjunction of positive(true) literals

Action Representation:

What is it for this goal?

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals

[redacted]

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

desired conjunction of positive(true) literals

Could be partially-specified

[redacted]

Goal: any state where A is directly on the table

Action Representation:

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals

[redacted]

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

desired conjunction of positive(true) literals

Could be partially-specified

[redacted]

Goal: any state where A is directly on the table

Action Representation:

What is it for this goal?

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals

[redacted]

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

desired conjunction of positive(true) literals

Action Representation:

***Preconditions**: conjunction of positive(true) literals that must be held true in order for the action to be applicable*

***Effect**: conjunction of positive(true) literals showing how the state will change (what should be deleted and added)*

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals [redacted]

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

desired conjunction of positive(true) literals

What are preconditions & effect for MoveToTable(b,x) action?

Action Representation:

***Preconditions**: conjunction of positive(true) literals that must be held true in order for the action to be applicable*

***Effect**: conjunction of positive(true) literals showing how the state will change (what should be deleted and added)*

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals [redacted]

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

desired conjunction of positive(true) literals

What are preconditions & effect for MoveToTable(b,x) action?

Action Rep

Precondi *...itive(true) literals that must*

b... ...plicable

Eff *...s showing how the*

state will change ... deleted and added)

MoveToTable(b,x)

Precond: On(b,x)^Clear(b)^Block(b)^Block(x)

Effect: On(b,Table)^Clear(x)^~On(b,x)

Generic Representation of Symbolic Planning Problems

- STRIPS (=Stanford Research Institute Problem Solver)

State Representation:

conjunction of positive(true) literals

(e.g, On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C))

Goal Representation:

desired conjunction of positive(true) literals

What are preconditions & effect for for Move(b,x,y) action?

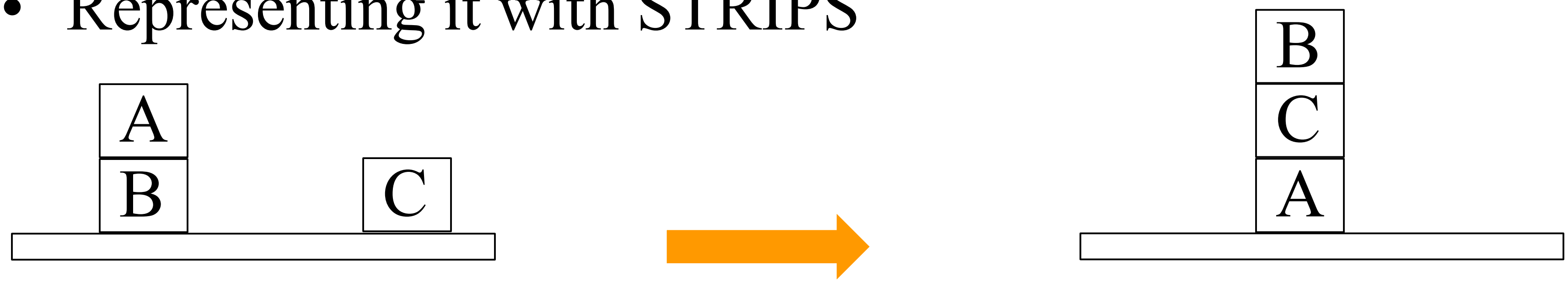
Action Rep

***Preconditions**: conjunction of positive(true) literals that must be held true in order for the action to be applicable*

***Effect**: conjunction of positive(true) literals showing how the state will change (what should be deleted and added)*

Back to the Example

- Representing it with STRIPS



Start state:

On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C)

Goal state:

On(B,C)^On(C,A)^On(A,Table)

Actions:

<u>MoveToTable(b,x)</u>

Precond: On(b,x)^Clear(b)^Block(b)^Block(x)

Effect: On(b,Table)^Clear(x)^~On(b,x)

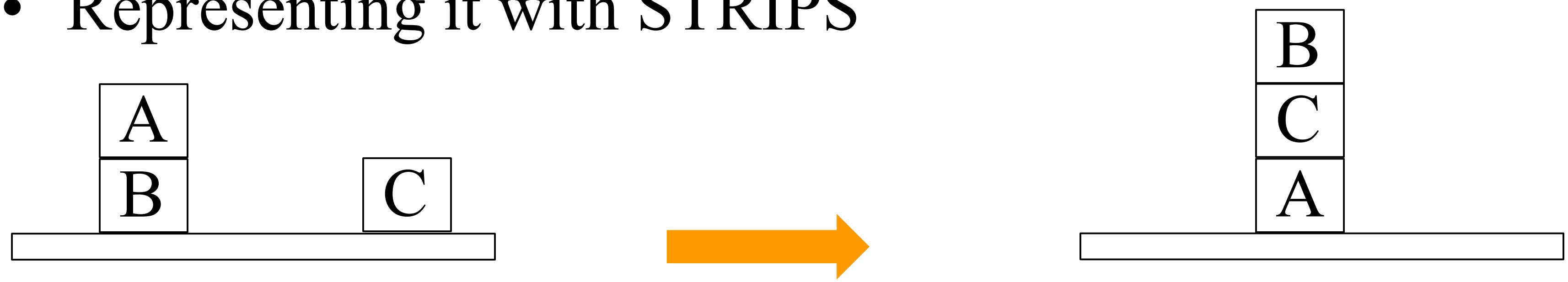
<u>Move(b,x,y)</u>

Precond: On(b,x)^Clear(b)^Clear(y)^Block(b)^Block(y)^(b~=y)

Effect: On(b,y)^Clear(x)^~On(b,x)^~Clear(y)

Back to the Example

- Representing it with STRIPS



Start state:

On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C)

Goal state:

On(B,C)^On(C,A)^On(A,Table)

Actions:

Problem (domain) specification

<u>MoveToTable(b,x)</u>

Precond: On(b,x)^Clear(b)^Block(b)^Block(x)

Effect: On(b,Table)^Clear(x)^~On(b,x)

<u>Move(b,x,y)</u>

Precond: On(b,x)^Clear(b)^Clear(y)^Block(b)^Block(y)^(b~=y)

Effect: On(b,y)^Clear(x)^~On(b,x)^~Clear(y)

Back to the Example

- Representing it with STRIPS

We can now write a (domain-independent) program that takes in such specifications and automatically provides a function GetSuccessors(state S, action A) required for implicit graph construction

This graph can be searched with A or any other search*

Start state:

On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C)

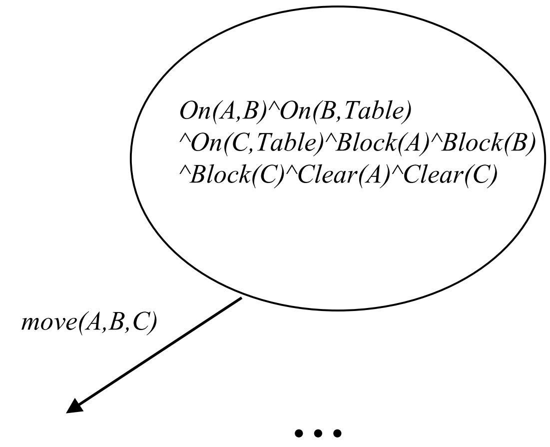
Goal state:

On(B,C)^On(C,A)^On(A,Table)

Actions:

<u>MoveToTable(b,x)</u>
Precond: On(b,x)^Clear(b)^Block(b)^Block(x)
Effect: On(b,Table)^Clear(x)^~On(b,x)

<u>Move(b,x,y)</u>
Precond: On(b,x)^Clear(b)^Clear(y)^Block(b)^Block(y)^(b~=y)
Effect: On(b,y)^Clear(x)^~On(b,x)^~Clear(y)



Back to the Example

- Representing it with STRIPS

We can now write a (domain-independent) program
that takes in such specifications
and automatically provides a function GetSuccessors(state S, action A)
required for implicit graph construction

Start state:

This graph can be
searched with A or any other search*

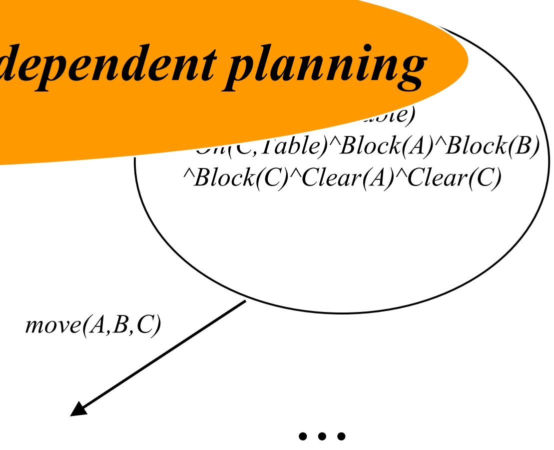
On(A,B)^On(B,Table)^On(C,Table)^ ... Clear(C)

Goal state:

*This is often referred to as **domain-independent planning***

On(B,C)^On(C,A)^On(...

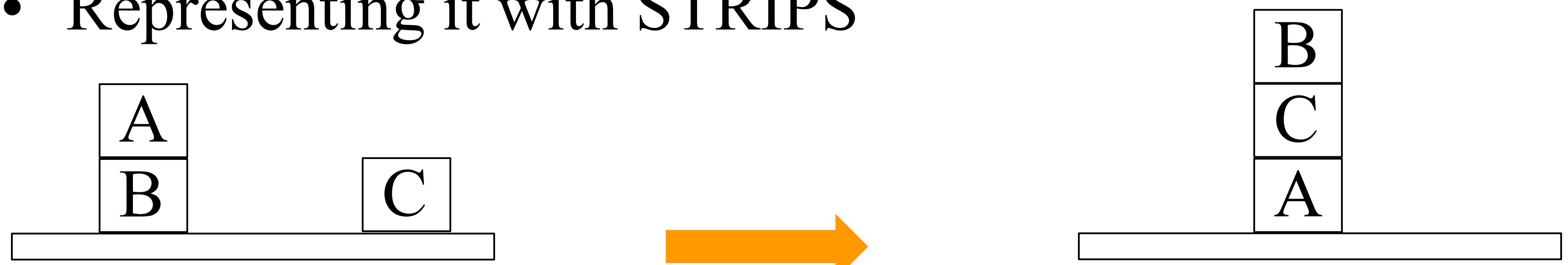
Actions:



MoveToTable(b,x)
Precond: On(b,x)^Clear(b)^Block(b)^Block(x)
Effect: On(b,Table)^Clear(x)^~On(b,x)
Move(b,x,y)
Precond: On(b,x)^Clear(b)^Clear(y)^Block(b)^Block(y)^(b~=y)
Effect: On(b,y)^Clear(x)^~On(b,x)^~Clear(y)

Back to the Example

- Representing it with STRIPS



Any ideas for domain-independent heuristics?

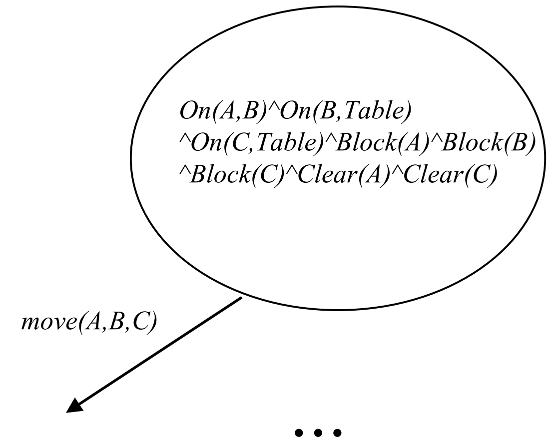
Start state:

On(A,B)^On(B,Table)^On(C,Table)^Block(A)^Block(B)^Block(C)^Clear(A)^Clear(C)

Goal state:

On(B,C)^On(C,A)^On(A,Table)

Actions:



MoveToTable(b,x)

Precond: On(b,x)^Clear(b)^Block(b)^Block(x)

Effect: On(b,Table)^Clear(x)^~On(b,x)

Move(b,x,y)

Precond: On(b,x)^Clear(b)^Clear(y)^Block(b)^Block(y)^(b~=y)

Effect: On(b,y)^Clear(x)^~On(b,x)^~Clear(y)

What You Should Know…

- How to represent a particular planning problem using STRIPS language and how this translates into a graph
- The motivation behind creating domain-independent planning representations such as STRIPS