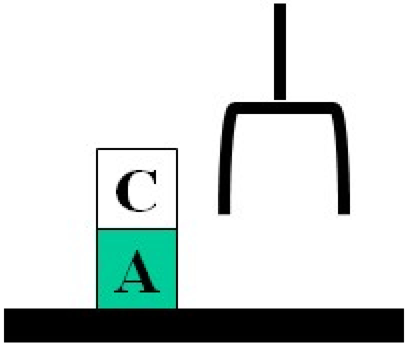


Planning (AI)

Dr. Ganesh Bhutkar
VIT, Pune INDIA
ganesh.bhutkar@vit.edu

TY BTech Comp - 2020-21

Planning

- **Planning** is finding a **sequence of actions** that achieves a given **goal,** when executed from a given **initial world state**.
- That is, given
  - **a set of operator descriptions** (defining the possible primitive actions by the agent),
  - **an initial state description** and
  - **a goal state description,**

  compute a plan, which is
  - a **sequence of operator instances**, such that executing them in the initial state will change the world to a state satisfying the goal-state description.
- **Goals** are usually specified as a **conjunction** of goals to be achieved.

Planning vs. Problem Solving

- **Planning** and **Problem Solving** methods can often solve the same sorts of problems.
- Planning is more **powerful** because of the representations and methods used.
- **States, goals, and actions** are decomposed into sets of sentences; usually in first-order logic.
- Search often proceeds through **plan space** rather than **state space**.
- **Subgoals** can be **planned independently**, reducing the complexity of the planning problem.

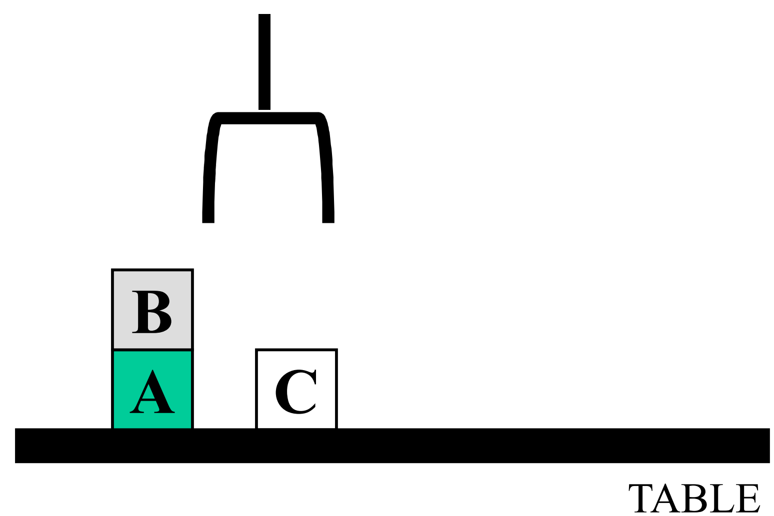
Planning: Typical Assumptions

- **Atomic time**: Each action is indivisible.
- **No concurrent actions** are allowed. (Though all actions do not need to be ordered with respect to each other in the plan.)
- **Deterministic actions:** The result of actions are completely determined - there is no uncertainty in their effects.
- **Agent** is the sole cause of change in the world.
- **Agent is omniscient**: He has complete knowledge of the state of the world.
- **Closed World Assumption**: Everything known to be true in the world, is included in the state description. Anything not listed is false.

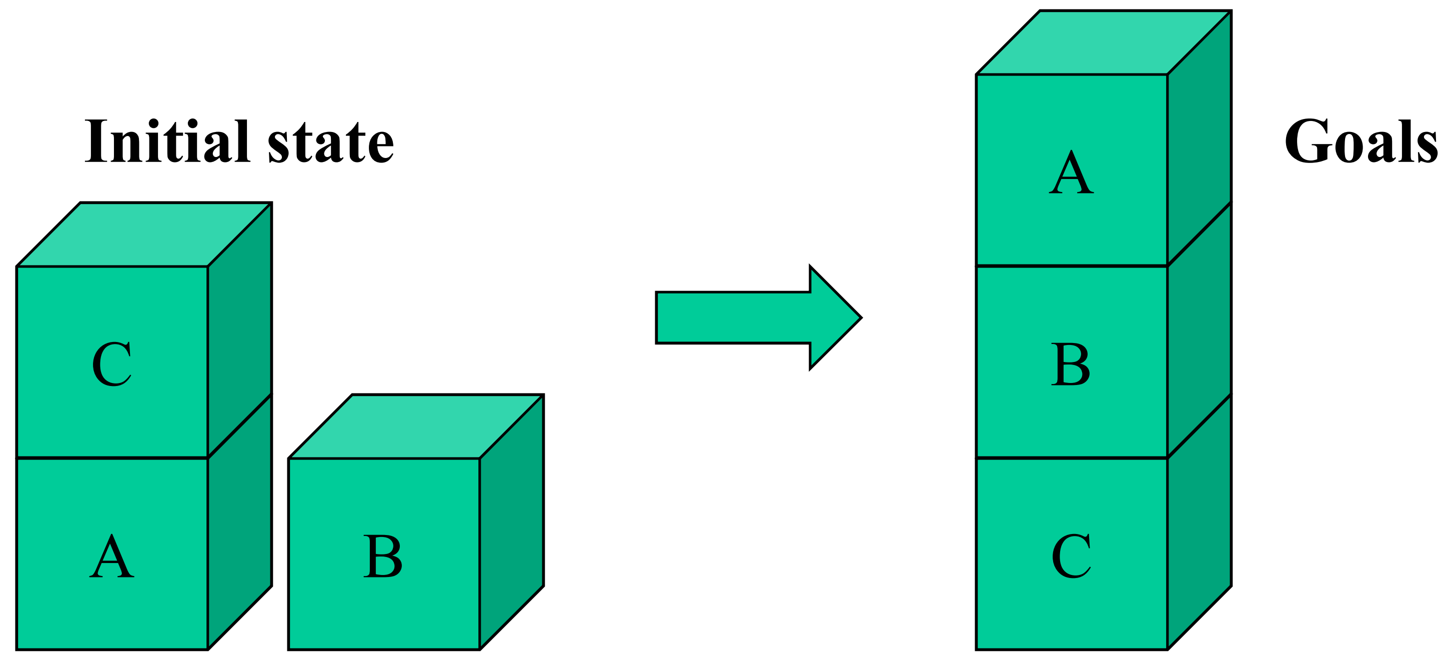
Blocks World

- The **blocks world** is a micro-world that consists of a table, a set of blocks and a robot hand.
- Some **domain constraints**:
  - Only one block can be on another block
  - Any number of blocks can be on the table
  - The hand can only hold one block
- **Typical representation:**
  STRIPS language

  ontable(A)
  ontable(C)
  on(B,A)
  handempty
  clear(B)
  clear(C)

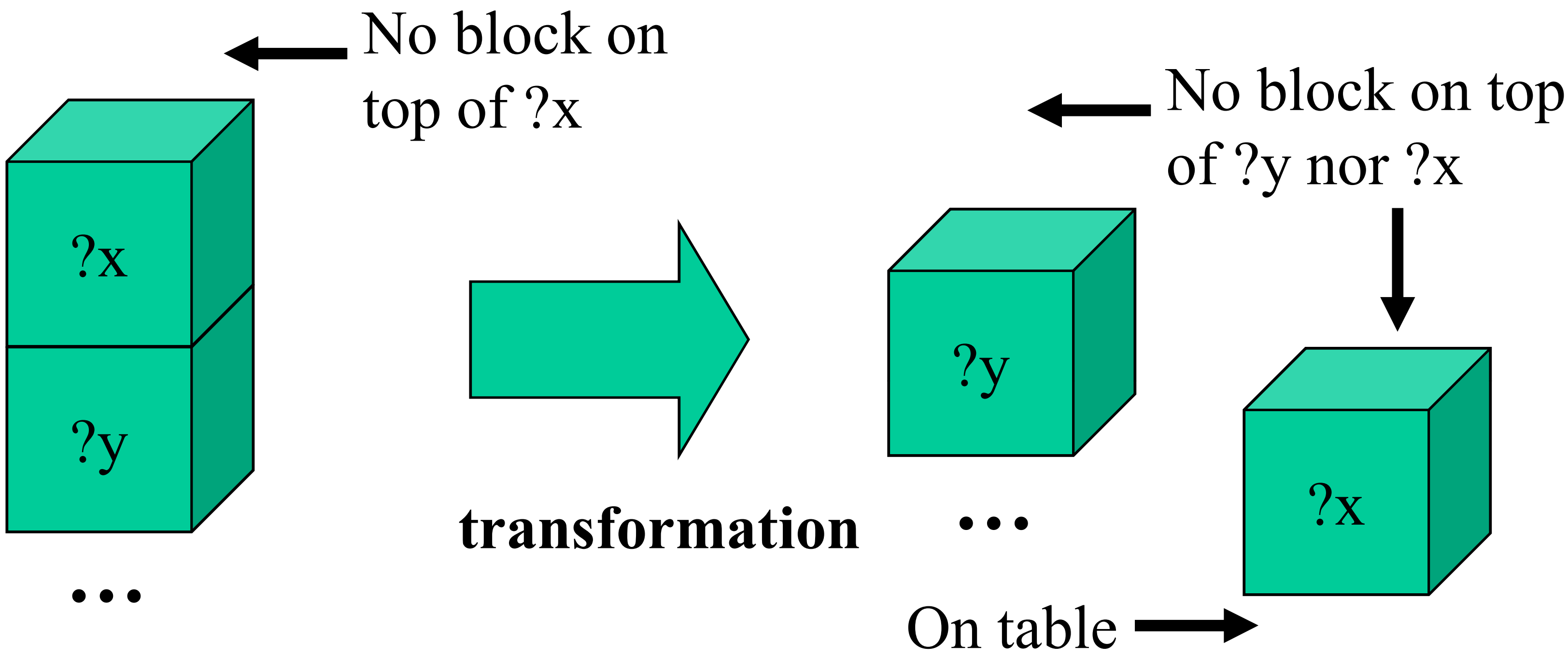


General-Purpose Planning: State & Goals



- **Initial state**: (onTable A) (on C A) (onTable B) (clear B) (clear C)
- **Goals**: (onTable C) (on B C) (on A B) (clear A)

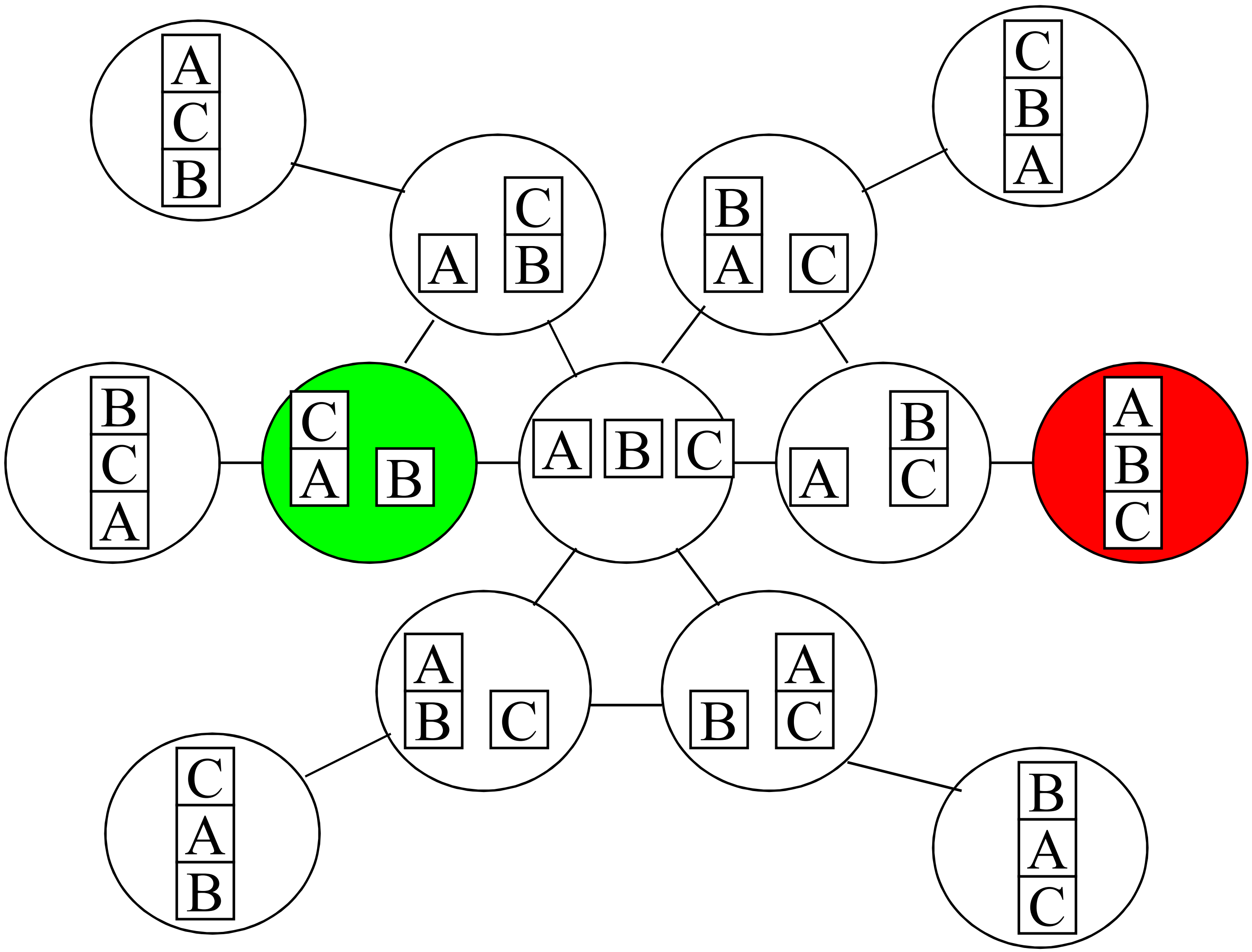
General-Purpose Planning: Operators



Operator: (Unstack ?x)

- **Preconditions**: (on ?x ?y) (clear ?x)
- **Effects**:
  - **Add**: (on ?x table) (clear ?y) & **Delete**: (on ?x ?y)

Planning: Search Space

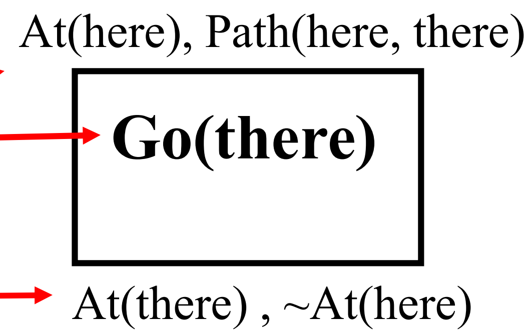


STRIPs Language for Planning

- **STandford Research Institute Problem Solver (STRIPS)**
- STRIPS - State of the world = conjunction of positive, ground, function-free literals
- At(Home) AND IsAt(Umbrella, Home) AND CanBeCarried(Umbrella) AND HandEmpty
- **Not OK as part of the state**:
  - NOT(At(Home)) (negative)
  - At(x)
  - At(Kitchen(Home)) (uses the function Kitchen)
- **Any literal not mentioned is assumed false.**
  - Other languages make different assumptions, e.g., negative literals part of state, unmentioned literals unknown.

Operator / Action Representation

- **Operators** contain three components:
  - **Action description**
  - **Precondition** - conjunction of positive literals
  - **Effect** - conjunction of positive or negative literals which describe how situation changes when operator is applied.



- Example:

  Op[Action: Go(there),

  Precond: At(here) ^ Path(here,there),

  Effect: At(there) ^ ~At(here)]
- All variables are universally quantified.
- Situation variables are implicit.
  - preconditions must be true in the state immediately before operator is applied; effects are true immediately after.

Block World Operators

- Here are the classic basic operations for the blocks world:
  - **stack(X,Y)**: put block X on block Y
  - **unstack(X,Y)**: remove block X from block Y
  - **pickup(X)**: pickup block X
  - **putdown(X)**: put block X on the table
- Each will be represented by
  - a list of **preconditions**
  - a list of **new facts** to be added **(add-effects)**
  - a list of **facts to be removed (delete-effects)**
  - optionally, a set of (simple) variable constraints
- **For example:**

  Preconditions (stack(X,Y), [holding(X),clear(Y)])

  deletes (stack(X,Y), [holding(X),clear(Y)]).

  adds (stack(X,Y), [handempty,on(X,Y),clear(X)])

Block World Operators - II

Operator - **stack(X,Y)**,

Pre [holding(X),clear(Y)],

Add [handempty,on(X,Y),clear(X)],

Delete [holding(X),clear(Y)],

Constr [X\==Y,Y\==table,X\==table]).

Operator - **unstack(X,Y)**,

[on(X,Y), clear(X), handempty],

[holding(X),clear(Y)],

[handempty, clear(X),on(X,Y)],

[X\==Y,Y\==table,X\==table]).

Operator - **pickup(X)**,

[ontable(X), clear(X), handempty],

[holding(X)],

[ontable(X),clear(X),handempty],

[X\==table]).

Operator - **putdown(X)**,

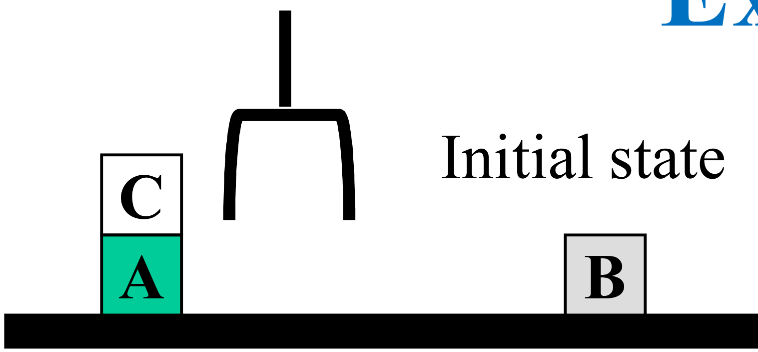
[holding(X)],

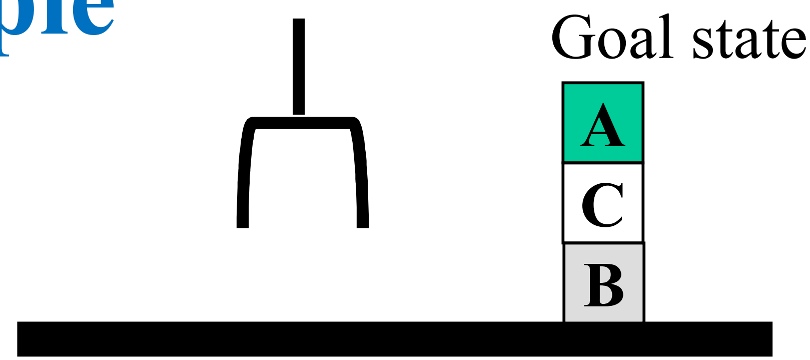
[ontable(X),handempty,clear(X)],

[holding(X)],

[X\==table]).

Example





1. Place on stack original goals

Goal Stack:

On(A,C) & On(C,B)

Database:

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

2. Since top goal is unsatisfied compound goal, list its unsatisfied subgoals on top of it:

Stack:

On(C,B)
On(A,C)
On(A,C) & On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

3. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and: a. Replace the goal with the instantiated rule; b. Place the rule’s instantiated precondition formula on top of stack

Stack:

Holding(C) & Clear(B)

stack(C,B) **On(C,B)**

On(A,C)

On(A,C) & On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

4. Since top goal is unsatisfied compound goal, list its subgoals on top of it:

Stack:

Clear(B)
Holding(C)
Holding(C) & Clear(B)
stack(C,B)
On(A,C)
On(A,C) & On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

5. Single goal on top of stack matches data base, so remove it:

Stack:

~~**Clear(B)**~~

Holding(C)

Holding(C) & Clear(B)

stack(C,B)

On(A,C)

On(A,C) & On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

6. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and: a. Replace the goal with the instantiated rule; b. Place the rule's instantiated precondition formula on top of stack

Stack:

On(C,A) &Clear(C) & Handempty
unstack(C,A) ⟷ **Holding(C)**
Holding(C) & Clear(B)
stack(C,B)
On(A,C)
On(A,C) &On(C,B)

Database: (unchanged)

Example

7. Compound goal on top of stack matches data base, so remove it:

Stack:

~~On(C,A) &Clear(C) & Handempty~~
unstack(C,A)
Holding(C) & Clear(B)
stack(C,B)
On(A,C)
On(A,C) &On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

8. Top item is rule, so:
  a. Remove rule from stack;
  b. Update database using rule;
  c. Keep track of rule (for solution)

Stack:

~~**unstack(C,A)**~~
Holding(C) & Clear(B)
stack(C,B)
On(A,C)
On(A,C) &On(C,B)

Database:

unstack(X,Y):

Add - [holding(X),clear(Y)]

Delete -[handempty,clear(X),on(X,Y)]

CLEAR(B)
ONTABLE(A)
ONTABLE(B)
HOLDING(C)
CLEAR(A)

Solution: {unstack(C,A)}

Example

9. Compound goal on top of stack matches data base, so remove it:

Stack:

~~**Holding(C) & Clear(B)**~~
stack(C,B)
On(A,C)
On(A,C) &On(C,B)

Database:
(unchanged)

CLEAR(B)
ONTABLE(A)
ONTABLE(B)
HOLDING(C)
CLEAR(A)

Solution: {unstack(C,A)}

Example

10. Top item is rule, so:
a. Remove rule from stack;
b. Update database using rule;
c. Keep track of rule (for solution)

Stack:

~~**stack(C,B)**~~
On(A,C)
On(A,C) &On(C,B)

Database:

stack(X,Y):
Add - [handempty,on(X,Y),clear(X)]
Delete - [holding(X),clear(Y)]

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C,A), stack(C,B)}

Example

11. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and: a. Replace the goal with the instantiated rule; b. Place the rule's instantiated precondition formula on top of stack

Stack:

Holding(A) &Clear(C)
stack(A,C)
On(A,C) &On(C,B)

On(A,C)

Database:
(unchanged)

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C,A), stack(C,B)}

Example

12. Since top goal is unsatisfied compound goal, list its unsatisfied sub-goals on top of it:

Stack:

~~**CLEAR(C)**~~
Holding(A)
Holding(A) &Clear(C)
stack(A,C)
On(A,C) &On(C,B)

Database:
(unchanged)

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C,A), stack(C,B)}

Example

13. Since top goal is unsatisfied compound goal, list its unsatisfied sub-goals on top of it:

Stack:

Holding(A)
Holding(A) &Clear(C)
stack(A,C)
On(A,C) &On(C,B)

Database:
(unchanged)

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C,A), stack(C,B)}

Example

14. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and: a. Replace the goal with the instantiated rule; b. Place the rule's instantiated precondition formula on top of stack

Stack:

Ontable(A) &Clear(A) &Handempty
pickup(A)
Holding(A) &Clear(C)
stack(A,C)
On(A,C) &On(C,B)

Holding(A)

Database:
(unchanged)

Solution: {unstack(C,A), stack(C,B)}

Example

15. Compound goal on top of stack matches data base, so remove it:

Stack:

~~**Ontable(A) &Clear(A) &Handempty**~~

pickup(A)

Holding(A) &Clear(C)

stack(A,C)

On(A,C) &On(C,B)

Database:
(unchanged)

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C,A), stack(C,B)}

Example

16. Top item is rule, so:

a. Remove rule from stack;
b. Update database using rule;
c. Keep track of rule (for solution)

Stack:

~~**pickup(A)**~~
Holding(A) &Clear(C)
stack(A,C)
On(A,C) &On(C,B)

Database:

pickup(X):
Add - [holding(X)]
Delete - [ontable(X),clear(X),handempty]

ONTABLE(B)
ON(C,B)
CLEAR(C)
HOLDING(A)

Solution: {unstack(C,A), stack(C,B), pickup(A)}

Example

17. Compound goal on top of stack matches data base, so remove it:

Stack:

~~Holding(A) &Clear(C)~~

stack(A,C)

On(A,C) &On(C,B)

Database: (unchanged)

ONTABLE(B)
ON(C,B)
CLEAR(C)
HOLDING(A)

Solution: {unstack(C,A), stack(C,B), pickup(A)}

Example

18. Top item is rule, so:
  a. Remove rule from stack;
  b. Update database using rule;
  c. Keep track of rule (for solution)

Stack:

~~**stack(A,C)**~~
On(A,C) &On(C,B)

Database:

stack(X,Y):
Add - [handempty,on(X,Y),clear(X)]
Delete - [holding(X),clear(Y)]

ONTABLE(B)
ON(C,B)
ON(A,C)
CLEAR(A)
HANDEMPTY

Solution: {unstack(C,A), stack(C,B), pickup(A), stack(A,C)}

Example

19. Compound goal on top of stack matches data base, so remove it:

Stack:

~~**On(A,C) &On(C,B)**~~

Database:

ONTABLE(B)
ON(C,B)
ON(A,C)
CLEAR(A)
HANDEMPTY

Solution: {unstack(C), stack(C,B), pickup(A), stack(A,C)}

---

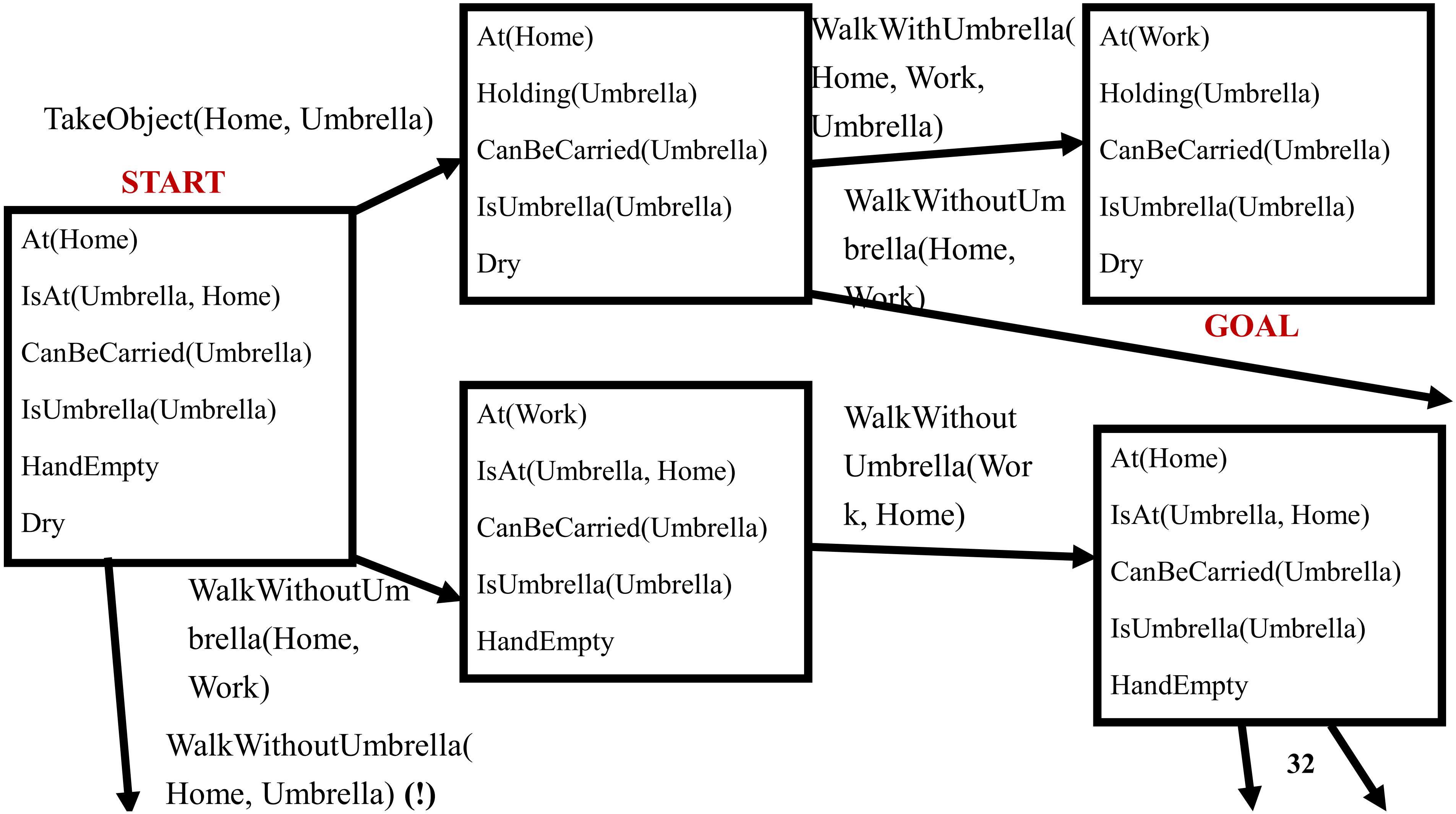
20. Stack is empty, so stop. ⇒

Solution: {unstack(C,A), stack(C,B), pickup(A), stack(A,C)}

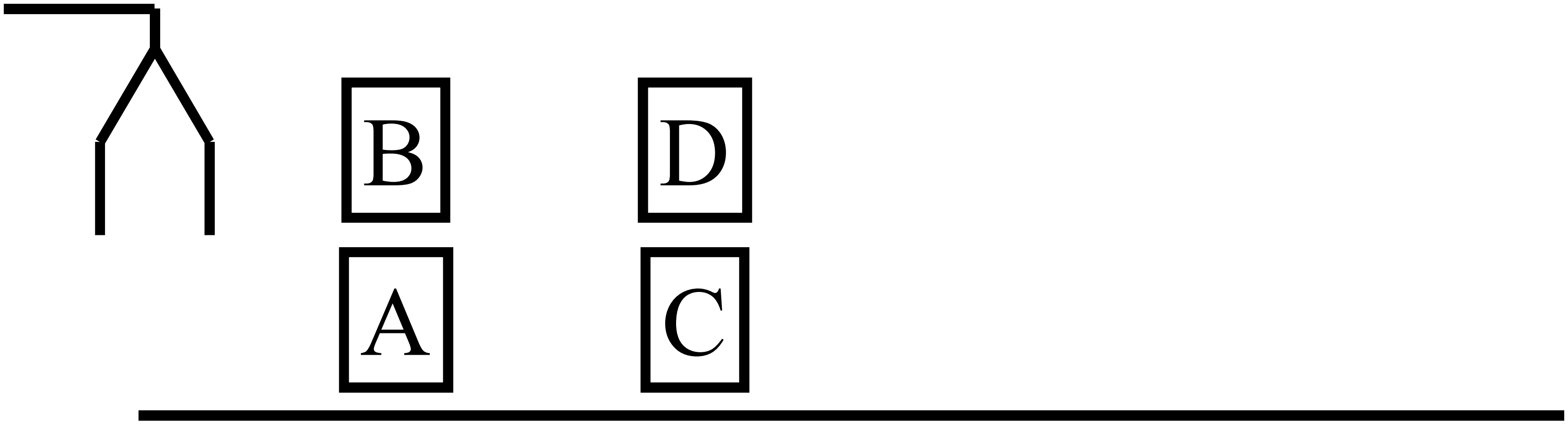
Forward State-Space Search

(Progression Planning)

- **Successors:** All states that can be reached with an action, whose preconditions are satisfied in current state.

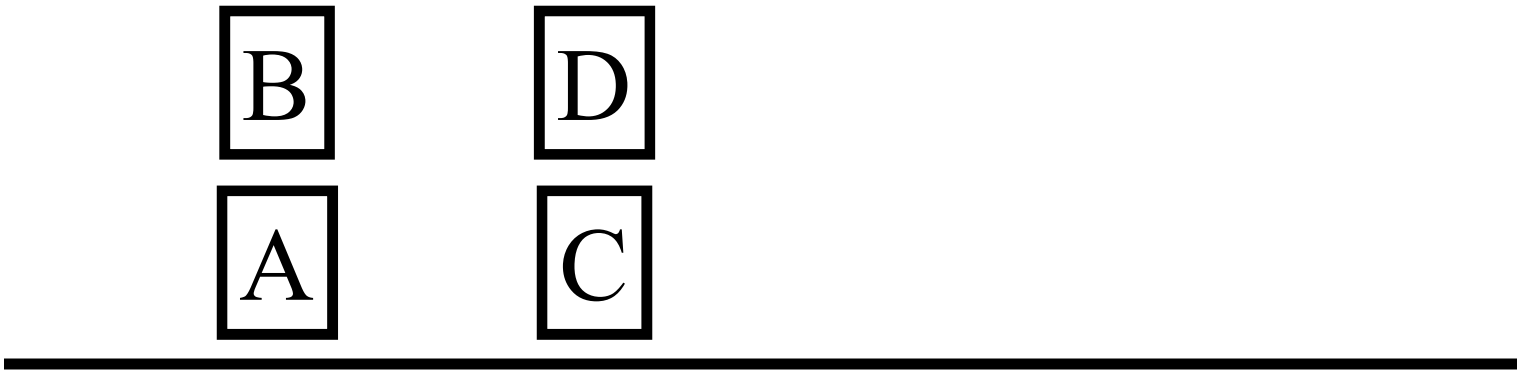


Blocks World Problem



- On(B, A), On(A, Table),
- On(D, C), On(C, Table),
- Clear(B), Clear(D)

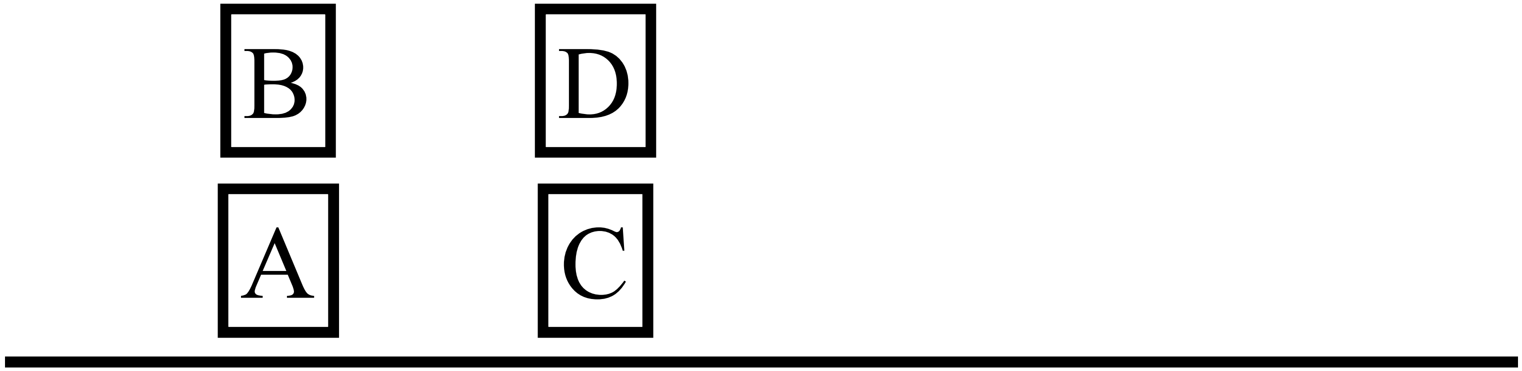
Blocks World Problem: Move Action



- Move(x,y,z)
- Preconditions:
  - On(x,y), Clear(x), Clear(z)
- Effects:
  - On(x,z), Clear(y), NOT(On(x,y)), NOT(Clear(z))

- Move(B,A,D)
- Preconditions:
  On(B,A), Clear(B), Clear(D)
- Effects:
  On(B,D), Clear(A), NOT(On(B,A)), NOT(Clear(D))

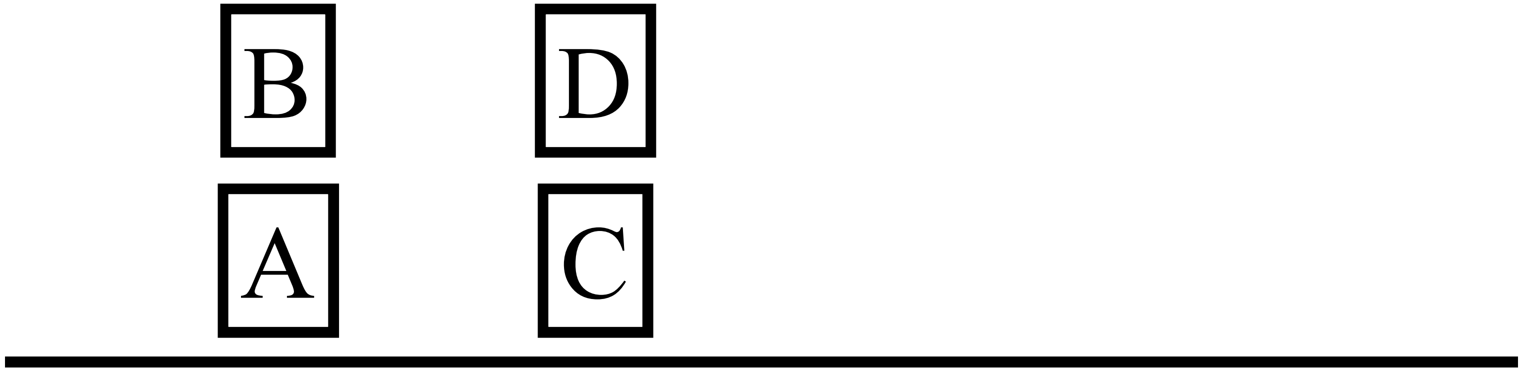
Blocks World Problem: MoveToTable Action



- MoveToTable(x,y)
- Preconditions:
  - On(x,y), Clear(x)
- Effects:
  - On(x,Table), Clear(y), NOT(On(x,y))

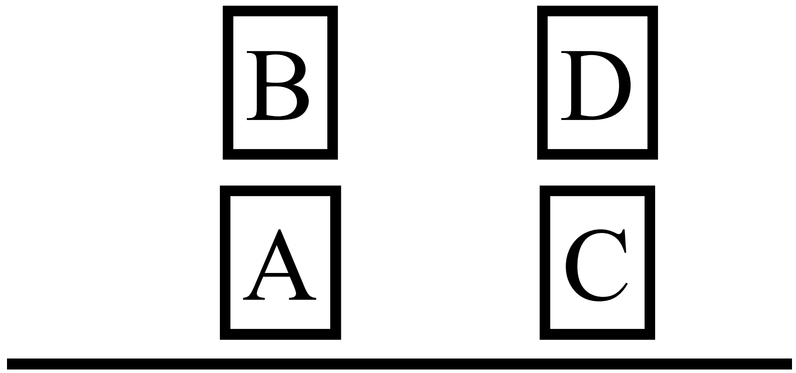
- MoveToTable(B,A)
- Preconditions:
  - On(B,A), Clear(B)
- Effects:
  - On(B,Table), Clear(A), NOT(On(B,A))

Blocks World Example

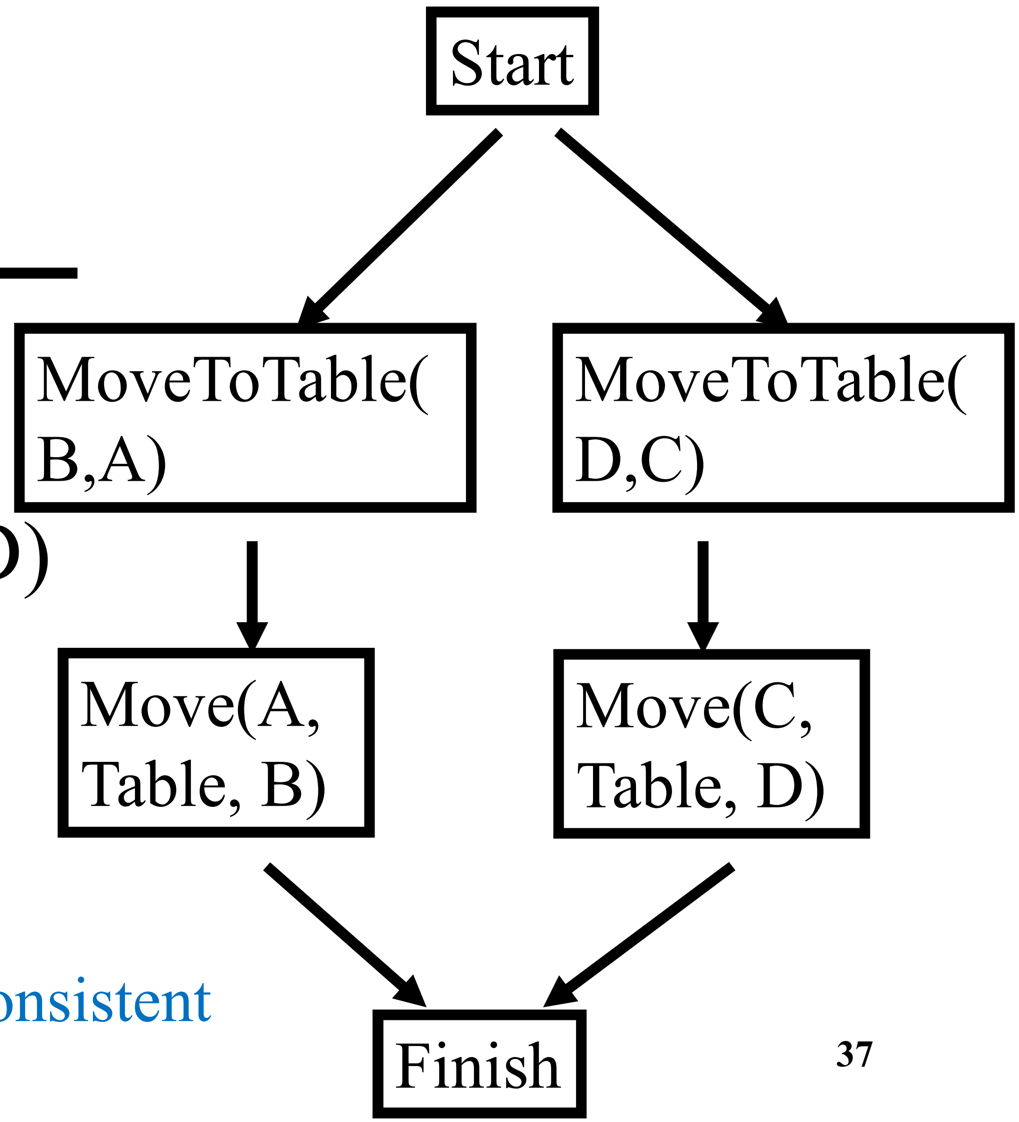


- Goal: On(A,B) AND Clear(A) AND On(C,D) AND Clear(C)
- A plan: MoveToTable(B, A), MoveToTable(D, C), Move(C, Table, D), Move(A, Table, B)
- Really two separate problems

Partial-Order Plan

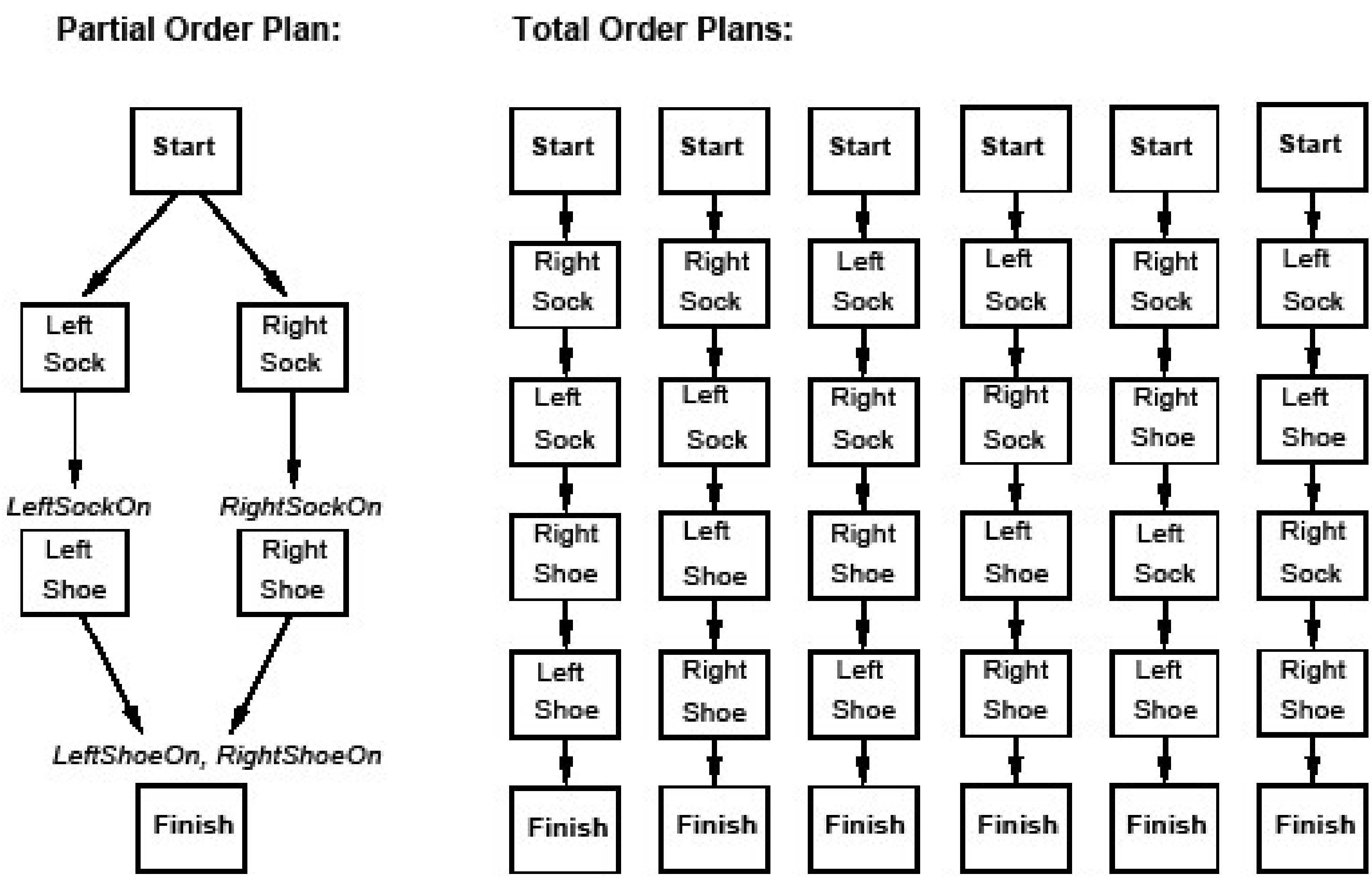


Goal: On(A,B) AND Clear(A) AND On(C,D) AND Clear(C)

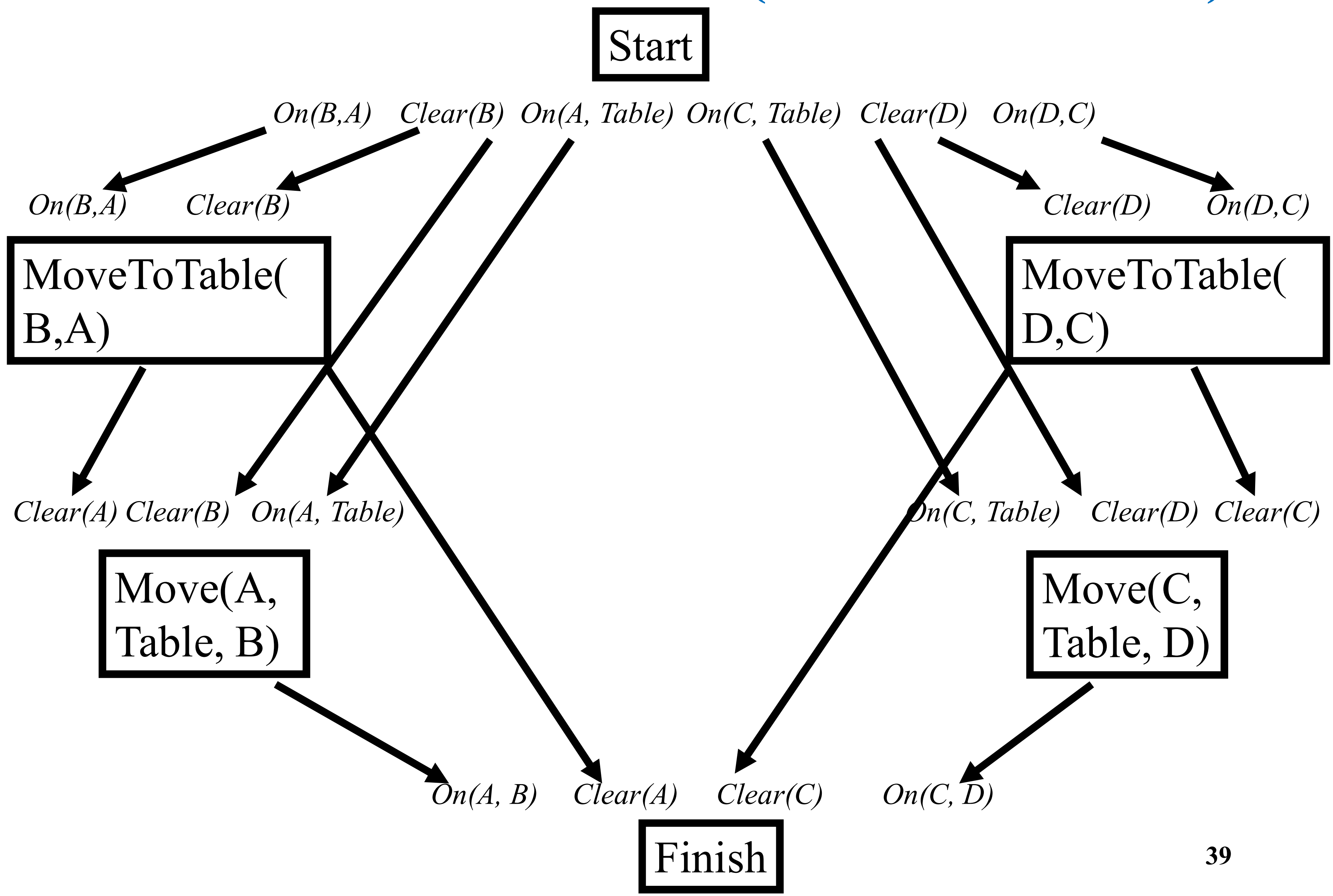


Any total order on the actions consistent with this partial order will work

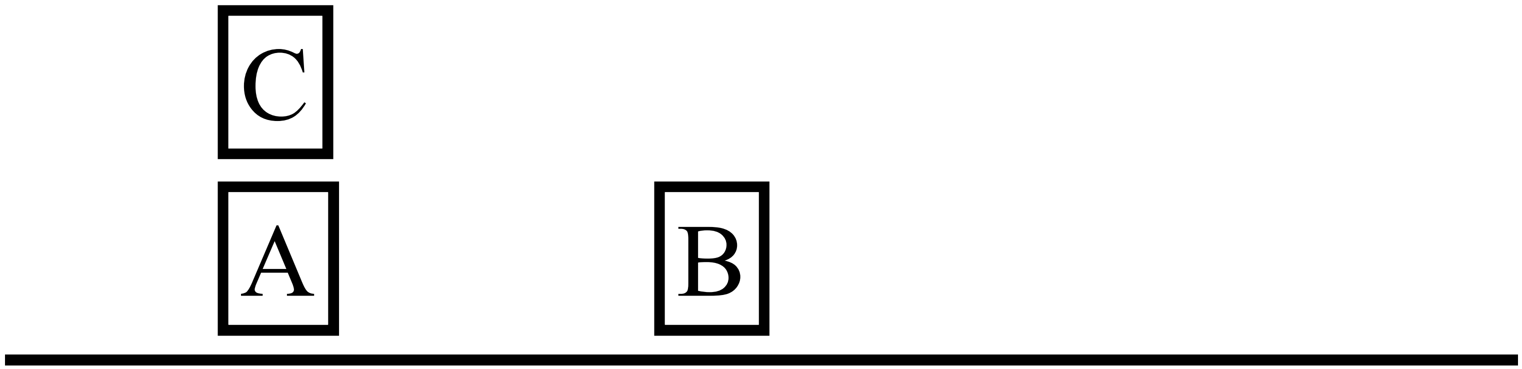
Partial-Order Plan: Shoe Example



Partial-Order Plan (with more details)

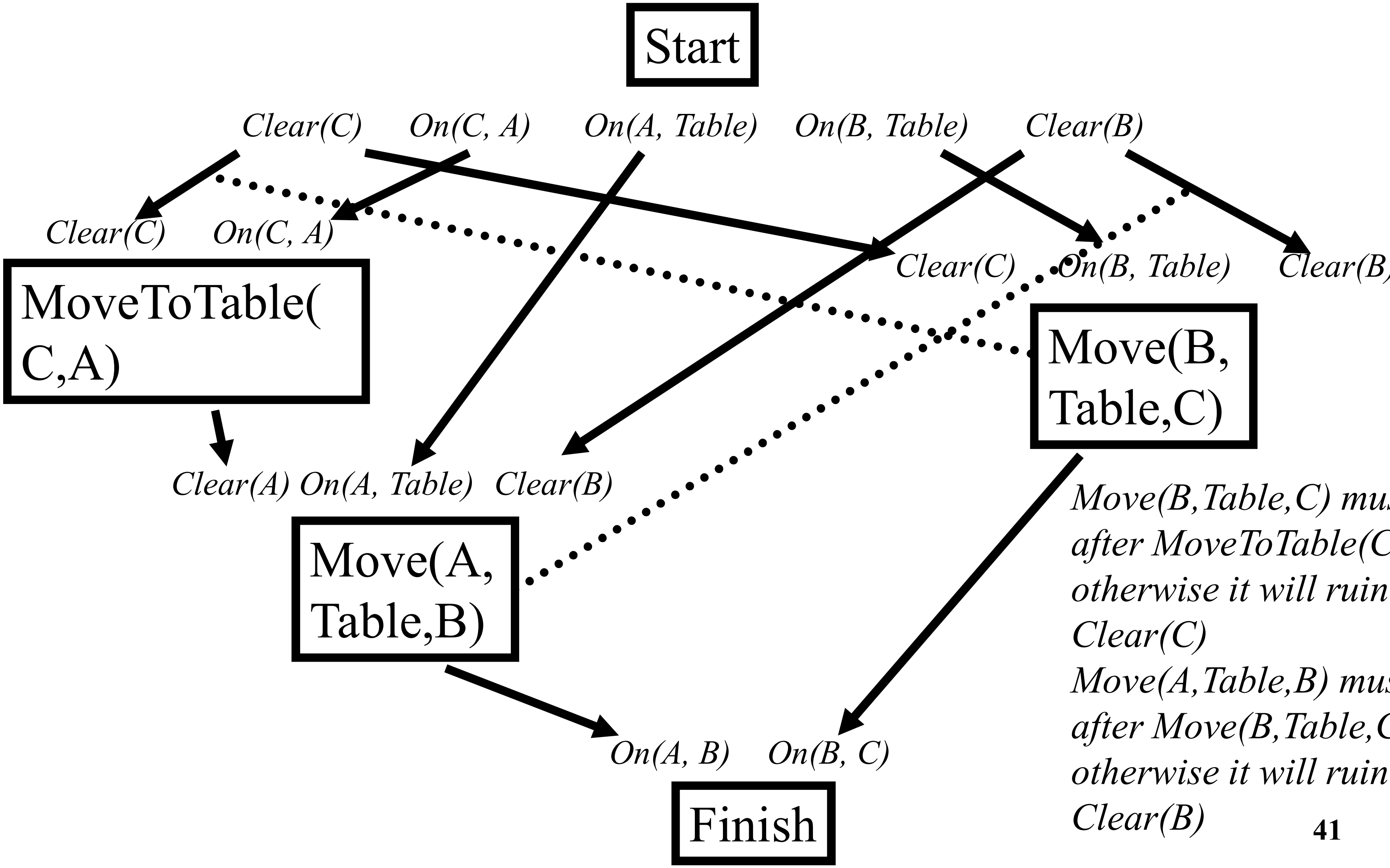


Not everything decomposes into multiple problems: Sussman Anomaly



- Goal: On(A,B) AND On(B,C)
- Focusing on one of these two individually first does not work
- Optimal plan: MoveToTable(C,A), Move(B,Table,C), Move(A,Table,B)

An **incorrect** partial order plan for the **Sussman Anomaly**



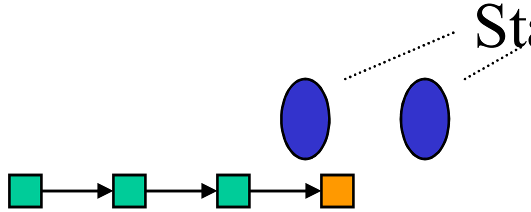
Move(B,Table,C) must be after MoveToTable(C,A), otherwise it will ruin Clear(C)

Move(A,Table,B) must be after Move(B,Table,C), otherwise it will ruin Clear(B)

State- and Plan-Space Planning

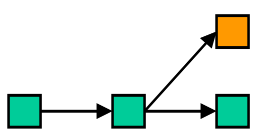
- **State-Space Planners** transform the state of the world. These planners search for a **sequence of transformations** linking the initial state and a goal state.

State of the world



(Total Order Planning)

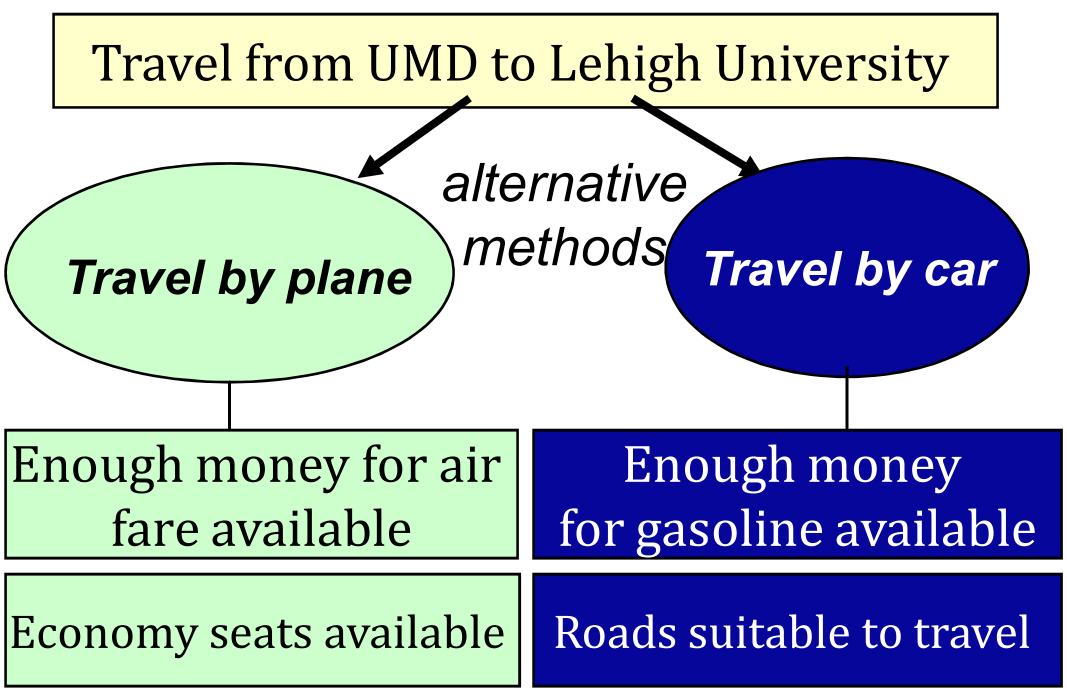
• **Plan-Space Planners** transform the plans. These planners search for a plan satisfying certain conditions.

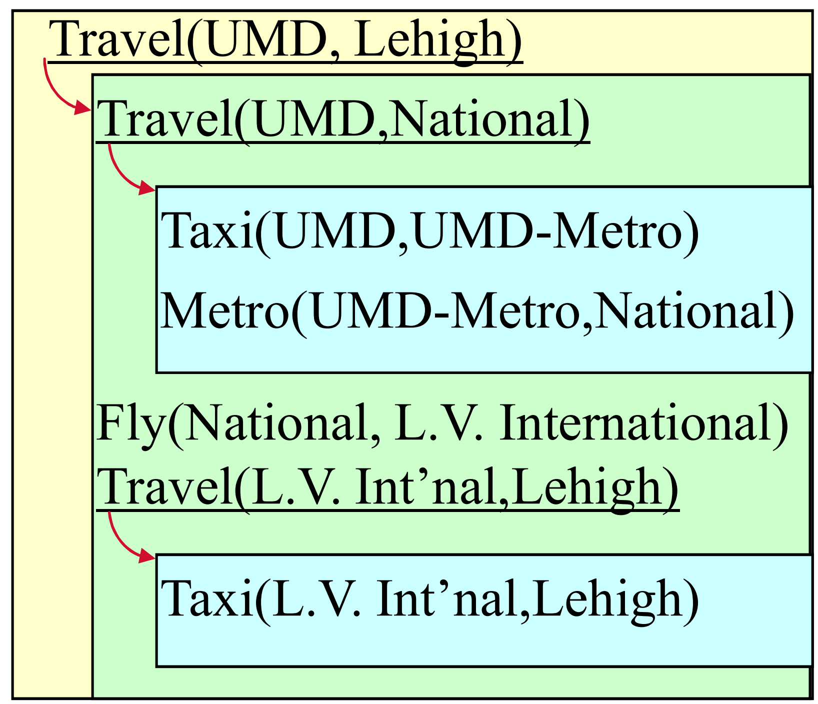


(Partial-Order Planning)

Hierarchical Planning

Principle: Complex tasks are decomposed into simpler tasks. The goal is to decompose all the tasks into *primitive* tasks, which define actions that change the world.





Thank You !