State, Action, Goal Representation; Classical Planning

Manuela Veloso

Carnegie Mellon University
School of Computer Science

15-887 Planning, Execution, and Learning – Fall 2016

Planning – Problem Solving

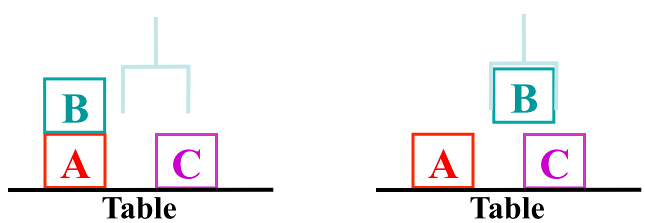
Newell and Simon 1956

- Given the *actions* available in a task domain.
- Given a problem specified as:
  - an initial *state* of the world,
  - goal statement - a set of *goals* to be achieved.
- Find a *solution* to the problem, i.e., a *way* to transform the initial state into a new state of the world where the goal statement is true.
- Planning is *"thinking…"*

Outline

- What is a ***State and Goal***
- What is an ***Action***
- What is a ***Plan***
- ***Finding*** a Plan

The Blocks World



- Blocks are on the Table, or on top of each other.
- There is an Arm – the Arm can be empty or holding one block.
- The table is always clear.

The Blocks World - States

- Objects
  - Blocks: *A, B, C*
  - Table: *Table*
- Predicates
  - *On(A, B), On(C, Table), Clear(B), Handempty, Holding(C)*
  - *On-table(A), On(A,B), Top(B),…*
  - *Tower(A,B,C,…)*
- States – Conjunctive
  - On(A,B) and On(B,C) and Clear(A) and Handempty

Classical Deterministic Planning

- Single initial state, complete, deterministic
  - CWA – closed world assumption:

  What is not true in the state, is false.
- Queries on state
  - On(A,B) – returns true or false
  - On(A,x) – returns x=Table or x=B
  - On-table (x) – returns x=A and x=B and x=C….

Blocks World State Description

A-on-B	Holding-A	*¬A-on-B ∧ ¬A-on-Table*
A-on-Table	Holding-B	*¬B-on-A ∧ ¬B-on-Table*
B-on-A	Handempty	*¬Holding-A ∧ ¬Holding-B*
B-on-Table	Clear-A	*¬B-on-A*
	Clear-B	*¬A-on-B*

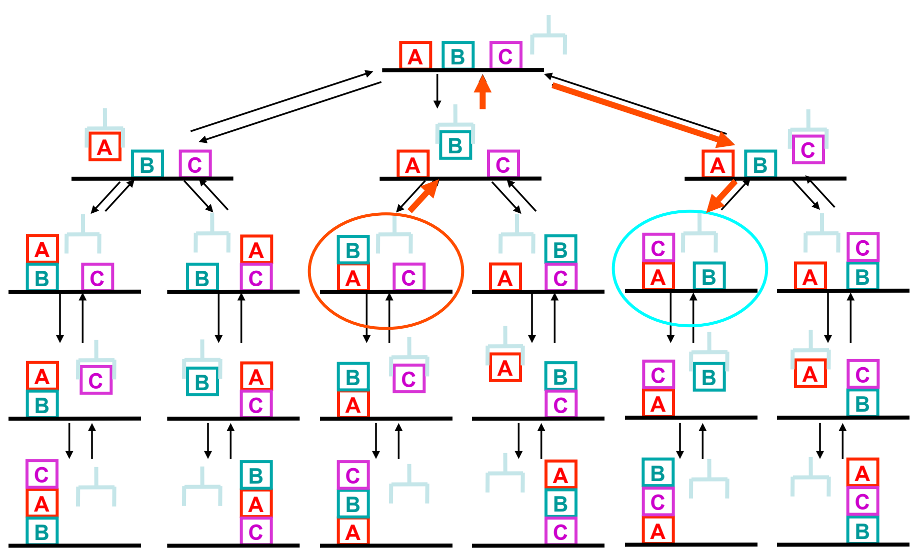
2^4 possible states

A-on-x {∅, table, B}

B-on-x {∅, table, A}

3^2 possible states

State-Space Search I



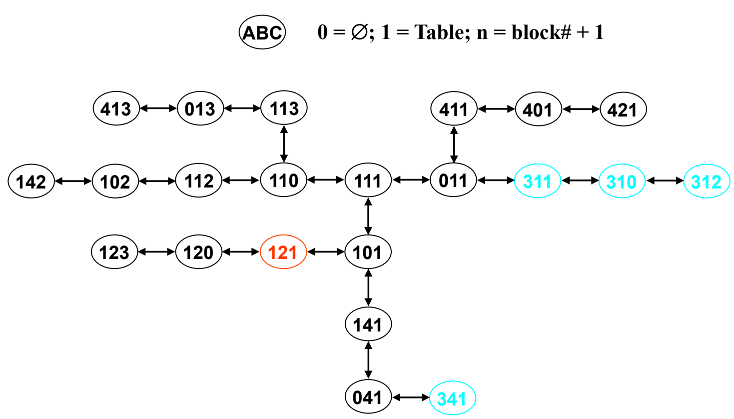
What is a Goal?

- Goal ***State***
  - Completely specified
- Goal ***Statement***
  - **Partially specified state**
- ***Preference Model***
  - Objective function
  - Defines "good" or "optimal" plan

Increasing Generality ↓

State-Space Search II

- Initial: A-on-x = Table; B-on-x = A; C-on-x = Table
- Goal: A-on-x = B



Models of World State

- Atomic identification (s1, s2,...)
- Symbolic – factored
  - Features
  - Predicates
- Conjunctive, observable
- Probabilistic, approximate
- Incremental, on-demand
- Temporal, dynamic

Predicates, conjunctive, complete, correct, deterministic

Outline

- What is a ***State and Goal***
- **What is an *Action***
- What is a ***Plan***
- ***Finding*** a Plan

What is an Action?

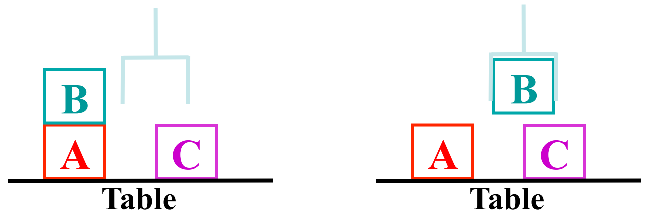
- Transition From One (Partial) State to Another
  - May be applicable only in particular states
  - Generates new state
    - Deterministic: t_{det}: S x A → S
    - Non-deterministic: $t_{non\text{-}det}$: S x A → 2^S
    - Probabilistic: t_{prob}: S x A → <2^S, r>

Explicit Action Representation

- (Grounded) Transition Matrix

	011	013	041	101	102	110	111	112	113	120	121	123	141	142	310	311	312	341	401	411	413	421
011							1									1				1		
013									1												1	
041													1					1				
101							1				1		1									
102								1						1								
110							1	1	1													
111	1			1		1																
112					1	1																
113		1				1																
120											1	1										
121				1						1												
123										1												
141			1	1																		
142					1																	
310																1	1					
311	1														1							
312															1							
341			1																			
401																				1		1
411	1																		1			
413		1																				
421																			1			

The Blocks World Dynamics – Actions

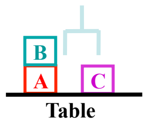


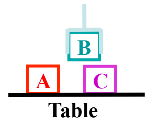
- Blocks are on the Table, or on top of each other.
- Blocks are picked up and put down by the arm.
- A block can be picked up only if it is clear, i.e., without a block on top.
- The arm can pick up a block only if the arm is empty, i.e., if it is not holding another block, i.e., the arm can pick up only one block at a time.
- The arm can put down blocks on blocks or on the table.
- The table is always clear.

STRIPS Action Representation

- Explicit action representation
  - $\{\text{preconds}(a), \text{effects}^-(a), \text{effects}^+(a)\}$
  - $\text{effects}^-(a) \cap \text{effects}^+(a) = \varnothing$
  - $\tau(S, a) = \{S - \text{effects}^-(a) \cup \text{effects}^+(a)\}$, where $S \in 2^s$

STRIPS – The Blocks World





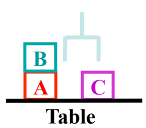
Pickup_from_table(?b)

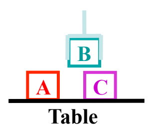
Pre:

Add:

Delete:

STRIPS – The Blocks World





Pickup_from_table(b)

Pre: Block(b), Armempty
Clear(b), On(b, Table)

Add: Holding(b)

Delete: Armempty,
On(b, Table)
clear(b)

Putdown_on_table(b)

Pre: Block(b), Holding(b)

Add: Armempty,
On(b, Table)

Delete: Holding(b)

Pickup_from_block(b1, b2)

Pre: Block(b1),Block(b2), Armempty
Clear(b1), On(b1,b2)

Add: Holding(b1), Clear(b2)

Delete: Armempty,
On(b1,b2)
Clear (b1)

Putdown_on_block(b1, b2)

Pre: Block(b1), Holding(b1)
Block(b2), Clear(b2), b1 $\neq$ b2

Add: Armempty, On(b1, b2)

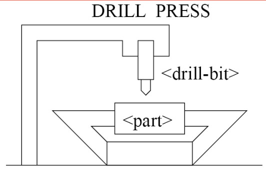
Delete: Holding(b1), Clear(b2)

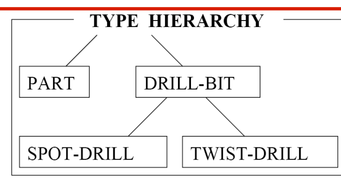
Actions

- An action *a* is **applicable** in *s* if all the preconditions of action *a are satisfied by s*.
- RESULT*(s,a)* = (*s* – Del (*a*)) U Add (*a*)
- No explicit mention of *time*
  - The precondition always refers to time *t*
  - The effect always refers to time *t+1*

Example – Action Model

DRILL PRESS





drill-spot (<part>, <drill-bit>)
<part>: type PART
<drill-bit>: type SPOT-DRILL
Pre: (holding-tool <drill-bit>)
(holding-part <part>)
Add: (has-spot <part>)

put-drill-bit (<drill-bit>)
<drill-bit>: type DRILL-BIT
Pre: tool-holder-empty
Add: (holding-tool <drill-bit>)
Del: tool-holder-empty

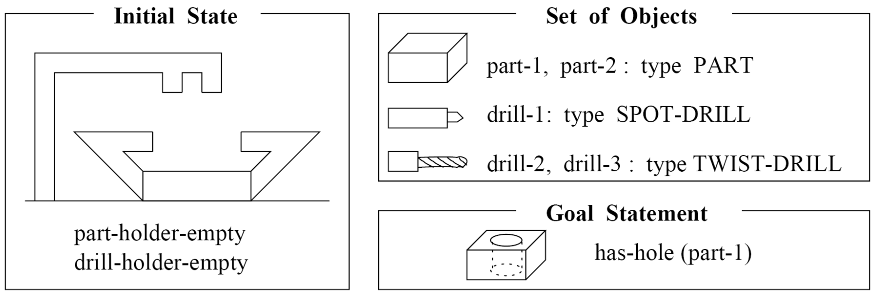
put-part(<part>)
<part>: type PART
Pre: part-holder-empty
Add: (holding-part <drill-bit>)
Del: part-holder-empty

drill-hole(<part>, <drill-bit>)
<part>: type PART
<drill-bit>: type TWIST-DRILL
Pre: (has-spot <part>)
(holding-tool <drill-bit>)
(holding-part <part>)
Add: (has-hole <part>)

remove-drill-bit(<drill-bit>)
<drill-bit>: type DRILL-BIT
Pre: (holding-tool <drill-bit>)
Add: tool-holder-empty
Del: (holding-tool <drill-bit>)

remove-part(<part>)
<part>: type PART
Pre: (holding-part <drill-bit>)
Add: part-holder-empty
Del: (holding-part <drill-bit>)

Example – Problem and Plan




```
put-part(part-1)
put-drill=bit(drill-1)
drill-spot(part-1, drill-1)
remove-drill-bit(drill-1)
put-drill-bit(drill-2)
drill-hole(part-1, drill-2)
```

PDDL Representation
Initial State, Goal, Actions Example-1

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

Figure 10.1 A PDDL description of an air cargo transportation planning problem.

Domain and Actions

- A *domain* can be represented by *many* possible choices of literals, variables, actions, preconditions, effects.

- Choice of domain
  - Granularity of representation
  - Detail of reasoning
  - Effectiveness of search

Initial State, Goal, Actions Example-2

$Init(On(A, Table) \wedge On(B, Table) \wedge On(C, A)$
$\quad \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C))$
$Goal(On(A, B) \wedge On(B, C))$
$Action(Move(b, x, y),$
$\quad$ PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y) \wedge$
$\qquad (b{\neq}x) \wedge (b{\neq}y) \wedge (x{\neq}y),$
$\quad$ EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$
$Action(MoveToTable(b, x),$
$\quad$ PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b{\neq}x),$
$\quad$ EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$

Figure 10.3 A planning problem in the blocks world: building a three-block tower. One solution is the sequence $[MoveToTable(C, A), Move(B, Table, C), Move(A, Table, B)]$.

One-Action Domain Representation – Blocksworld

```
(OPERATOR MOVE
 :preconds
   ?block BLOCK
   ?from OBJECT
   ?to OBJECT
  (and (clear ?block)
       (clear ?to)
       (on ?block ?from)
 :effects
  add (on ?block ?to)
  del (on ?block ?from)
  (if (block-p ?from)
      add (clear ?from))
  (if (block-p ?to)
      del (clear ?to)))
```

The Art of Planning

Initial: Consumed(A, Fish), Vigorous(Fish), Vigorous(Tea), Zen(A), Zen(Tea), Satisfied
Goal: Vigorous(A) , Consumed(Fish, Tea)

Eat(person, thing)
- Pre: Enlightened(person), Zen(thing), person ≠ thing
- Add: Satisfied, Consumed(person, thing)
- Delete: Enlightened(person), Zen(thing)

Man(person)
- Pre: Zen(person), Satisfied, Vigorous(person)
- Add: Enlightened(person)
- Delete: Vigorous(person), Satisfied

Drink(person, thing)
- Pre: Zen(person), Satisfied, Consumed(person, thing)
- Add: Enlightened(person), Zen(thing)
- Delete: Consumed(person, thing), Satisfied

Woman(person)
- Pre: Enlightened(person)
- Add: Vigorous(person), Satisfied
- Delete: Enlightened(person)

More Realistic Action Representations I

- Conditional Effects

  Pickup (b)
  Pre: Block(b), Handempty, Clear(b), On(b, x)
  Add: Holding(b)
  if (Block(x)) then Clear(x)
  Delete: Handempty, On(b, x)

- Quantified Effects

  Move (o, x)
  Pre: At(o, y), At(Robot, y)
  Add: At(o, x), At(Robot, x)
  forall (Object(u)) [if (In(u, o)) then At(u, y)]
  Delete: At(o, y), At(Robot, y), forall (Object(u)) [if (In(u, o)) then At(u, y)]

- Disjunctive and Negated Preconditions

  Holding(x) Or Not[Lighter_Than_Air(x)]

All these extensions can be emulated by adding actions

More Realistic Action Representations II

- ***These extensions make the planning problem significantly harder***
- Inference Operators / Axioms

  Clear(x) iff forall(Block(y))[Not[On(y, x)]]

- Functional Effects

  Move (o, x)
  Pre: At(o, y), At(Robot, y), Fuel(f), f ≥ Fuel_Needed(y, x)
  Add: At(o, x), At(robot, x), Fuel(f – Fuel_Needed(y, x)),
  forall (Object(u)) [if (In(u, o)) then At(u, y)]
  Delete: At(o, y), At(Robot, y), Fuel(f),
  forall (Object(u)) [if (In(u, o)) then At(u, y)]

More Realistic Action Representations III

- ***These extensions make the problem even harder still***
- Disjunctive Effects

  Pickup_from_block(b)

  Pre: Block(b), Handempty, Clear(b), On(b, c), Block(c)

  C1: Add: Clear(c), Holding(b); Delete: On(b, c), Handempty

  C2: Add: Clear(c), On(b, Table); Delete: On(b, c)

  C3: Add: ; Delete:
- Probabilistic Effects
  - Add probabilities to contexts of disjunctive effects
- Other Extensions
  - External events
  - Concurrent events
  - Sensing actions
  - Actions with duration

Outline

- What is a ***State and Goal***
- What is an ***Action***
- **What is a *Plan***
- ***Finding*** a Plan