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 \neg A\text{-}on\text{-}B \land \neg A\text{-}on\text{-}Table
Holding-B \neg B\text{-}on\text{-}A \land \neg B\text{-}on\text{-}Table
A-on-Table
Handempty \neg Holding\text{-}A \land \neg Holding\text{-}B
B-on-A
Clear-A \neg B\text{-}on\text{-}A
B-on-Table
Clear-B \neg A\text{-}on\text{-}B

2^4 possible states

A-on-x \{\emptyset, \text{table, } B\}
B-on-x \{\emptyset, \text{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

State-Space Search II

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

\[ 0 = \emptyset; 1 = \text{Table}; n = \text{block#} + 1 \]
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: $t_{det}: S \times A \rightarrow S$
    - Non-deterministic: $t_{non-det}: S \times A \rightarrow 2^S$
    - Probabilistic: $t_{prob}: S \times A \rightarrow <2^S, r>$

Explicit Action Representation

- (Grounded) Transition Matrix
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) = \emptyset
  - \tau(S, a) = \{S - \text{effects}^-(a) \cup \text{effects}^+(a)\}, \text{ where } S \in 2^s
STRIPS – The Blocks World

Pickup_from_table(?b)
Pre:
Add:
Delete:

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)
Delete: Holding(b1), Clear(b2)
Actions

- An action \( a \) is **applicable** in \( s \) if all the preconditions of action \( a \) are satisfied by \( s \).

- \( \text{RESULT}(s,a) = (s - \text{Del}(a)) \cup \text{Add}(a) \)

- No explicit mention of **time**
  - The precondition always refers to time \( t \)
  - The effect always refers to time \( t+1 \)

---

**Example – Action Model**

```plaintext

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

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>)

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

remove-part(<part>)
<part>: type PART
Pre: (holding-part <drill-bit>)
Add: part-holder-empty
Del: (holding-part <drill-bit>)
```
**Example – Problem and Plan**

**Initial State**
- part-holder-empty
- drill-holder-empty

**Set of Objects**
- part-1, part-2 : type PART
- drill-1: type SPOT-DRILL
- drill-2, drill-3: type TWIST-DRILL

**Goal Statement**
- has-hole (part-1)

- 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**

```
Inst(At(C1, SFO) ∧ At(C2, JFK) ∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
Action(LOAD(c, p, a),
   PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
   EFFECT: −At(c, a) ∧ In(c, p))
Action(UNLOAD(c, p, a),
   PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
   EFFECT: At(c, a) ∧ −In(c, p))
Action(FLY(p, from, to),
   PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
   EFFECT: −At(p, from) ∧ 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) \land On(B, Table) \land On(C, A) \land Block(A) \land Block(B) \land Block(C) \land Clear(B) \land Clear(C))

Goal(On(A, B) \land On(B, C))

Action(Move(b, x, y),
  PRECOND: On(b, x) \land Clear(b) \land Clear(y) \land Block(b) \land Block(y) \land (b \neq x) \land (b \neq y) \land (x \neq y),
  EFFECT: On(b, y) \land Clear(x) \land \neg On(b, x) \land \neg Clear(y))

Action(MoveToTable(b, x),
  PRECOND: On(b, x) \land Clear(b) \land Block(b) \land (b \neq x),
  EFFECT: On(b, Table) \land Clear(x) \land \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, Clean(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 — Sensing actions
  - Concurrent events — Actions with duration

Outline

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