Planning - I

Manuela Veloso
CMU 15-381, Fall 2001

- Planning domain representation
- General problem solving
- Means-ends analysis
- State-space planning

These lecture slides should be complemented with the examples discussed and worked out on the blackboard during the lecture.

Artificial Intelligence

Computer Science:
"The study of computers and the phenomena that surround them." (Perlin, Newell, Simon)

Artificial Intelligence - ambitious goals:

- What is the nature of human intelligence?
- How does the brain work?
- How to solve problems effectively?
- How do we learn?
- How to build intelligent artifacts?

Cognition, Perception, and Action

Selecting ACTIONS

- Planning algorithms
- Comparison of planning algorithms
- Learning in planning
- Planning, execution, and learning
- Behavior-based (reactive) planning
- Action selection in multiagent systems

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,
  - ▷ 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.

Action Model, State, Goals

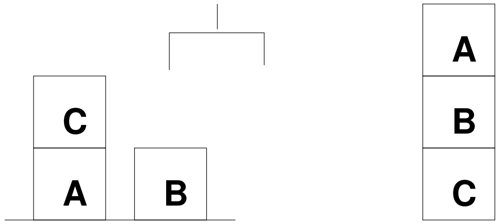
Classical Deterministic Planning

- Action Model: complete, deterministic, correct, rich representation
- State: single initial state, fully known
- Goals: complete satisfaction

Several different planning algorithms

The Blocksworld

Winograd 76?



Sussman's anomaly

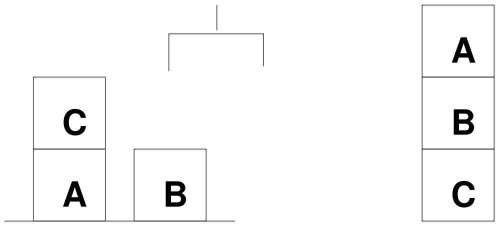
Domain Representation

- Operators – rules – with:
  - Precondition expression – must be satisfied before the operator is applied.
  - Set of effects – describe how the application of the operator changes the state.
- Precondition expression: propositional, typed first order predicate logic, negation, conjunction, disjunction, existential and universal quantification, and functions.
- Effects: add-list and delete-list.
- Conditional effects – dependent on condition on the state.

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)))
```

Plan for the Sussman's Anomaly




```
MOVE C A table
MOVE B table C
MOVE A table B
```

Different Representations

```
(OPERATOR PICK-UP
 :preconds
   ?ob BLOCK
  (and (clear ?ob)
       (on-
table ?ob)
       (arm-empty))
 :effects
  del (on-
table ?ob)
  del (clear ?ob)
  del (arm-empty)
  add (hold-
ing ?ob))
```

```
(OPERATOR PUT-
DOWN
  :preconds
   ?ob BLOCK
  (hold-
ing ?ob)
  :effects
   del (hold-
ing ?ob)
   add (clear ?ob)
   add (arm-
empty)
   add (on-
table ?ob))
```

Different Representations

```
(OPERATOR STACK
  :preconds
   ?ob BLOCK
   ?uob BLOCK
  (and (clear ?uob)
       (hold-
ing ?ob))
  :effects
   del (hold-
ing ?ob)
   del (clear ?uob)
   add (clear ?ob )
   add (arm-empty)
   add (on ?ob ?uob))
```

```
(OPERATOR UNSTACK
  :preconds
   ?ob BLOCK
   ?uob BLOCK
  (and (on ?ob ?uob)
       (clear ?ob)
       (arm-
empty))
  :effects
   del (on ?ob ?uob)
   del (clear ?ob)
   del (arm-
empty)
   add (hold-
ing ?ob)
   add (clear ?uob))
```

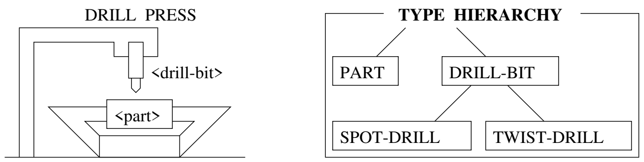
Many Issues To Resolve...

- What knowledge *defines* the task domain?
- How to *represent the action model*?
- What is a *sufficient initial state* of the world?
- What are *prioritized goals*?
- How *to acquire* domain knowledge efficiently from expert human planners?
- Which *algorithm to generate* a solution plan?
- How to create plans of *good* quality?
- How *to scale up* to real-world problems?

Many Planning "Domains"

- Web management agents
- Robot planning
- Manufacturing planning
- Image processing management
- Logistics transportation
- Crisis management
- Bank risk management
- Blocks world
- Puzzles
- Artificial domains

Example - Action Model



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>)

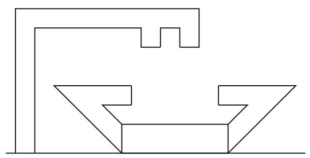
remove-drill-bit(<drill-bit>)
<drill-bit>: type DRILL-BIT
Pre: (holding-tool <drill-bit>)
Add: tool-holder-empty
Del: (holding-tool <drill-bit>)

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

Actions with preconditions and effects as state changes

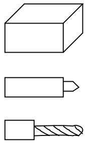
Example - Problem and Plan

Initial State



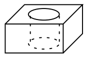
part-holder-empty
drill-holder-empty

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)
```

Generating a Solution Plan

- Linear planning – Planning with a goal **stack**.
- Nonlinear planning – Interleaving of goals
  - ▷ State-space search
  - ▷ Plan-space search
  - ▷ Graph-based search
  - ▷ Sat-based search
  - ▷ OBDD-based search
- Hierarchical planning
  - ▷ Emphasis on action decomposition/refinement
  - ▷ Knowledge engineering/acquisition
  - ▷ Very little search

Generating a Solution Plan

A complex process:

- **Alternative operators to achieve a goal.**
- **Multiple goals that interact.**
- **Efficiency, quality, and accuracy – hard.**

Means-ends Analysis

[*Newell and Simon 60s*] [*Ernst and Newell 69*]

GPS Algorithm (*initial-state, goals*)

- If *goals* $\subseteq$ *initial-state*, then return *True*
- Choose a difference $d \in$ *goals* between *initial-state* and *goals*
- Choose an operator o to reduce the difference d
- If no more operators, then return *False*
- *State*=**GPS**(*initial-state*, preconditions(o))
- If *State*, then return **GPS**(apply(o,*initial-state*), *goals*)

Example: Irreversible Actions

One-Way Rocket Domain (Veloso 89)

```
(OPERATOR LOAD-ROCKET
 :preconds
  ?roc ROCKET
  ?obj OBJECT
  ?loc LOCATION
 (and (at ?obj ?loc)
      (at ?roc ?loc))
 :effects
  add (inside ?obj ?roc)
  del (at ?obj ?loc))
```

```
(OPERATOR UNLOAD-ROCKET
 :preconds
  ?roc ROCKET
  ?obj OBJECT
  ?loc LOCATION
 (and (inside ?obj ?roc)
      (at ?roc ?loc))
 :effects
  add (at ?obj ?loc)
  del (inside ?obj ?roc))
```

```
(OPERATOR MOVE-ROCKET
 :preconds
  ?roc ROCKET
  ?from-l LOCATION
  ?to-l LOCATION
 (and (at ?roc ?from-l)
      (has-fuel ?roc))
 :effects
  add (at ?roc ?to-l)
  del (at ?roc ?from-l)
  del (has-fuel ?roc))
```

Incompleteness of Linear Planning

Initial state:

(at obj1 locA)
(at obj2 locA)
(at ROCKET locA)
(has-fuel ROCKET)

Goal statement:

(and
(at obj1 locB)
(at obj2 locB))

Goal	*Plan*
(at obj1 locB)	(LOAD-ROCKET obj1 locA) (MOVE-ROCKET) (UNLOAD-ROCKET obj1 locB)
(at obj2 locB)	*failure*

State-Space Nonlinear Planning

Extend linear planning:

- From **stack** to **set** of goals.

- Include in the search space all possible interleaving of goals.

State-space nonlinear planning is **complete**.

Goal	*Plan*
(at obj1 locB)	(LOAD-ROCKET obj1 locA)
(at obj2 locB)	(LOAD-ROCKET obj2 locA)
	(MOVE-ROCKET)
(at obj1 locB)	(UNLOAD-ROCKET obj1 locB)
(at obj2 locB)	(UNLOAD-ROCKET obj2 locB)

Prodigy4.0 Planning Algorithm

Veloso et al. 90

1. Terminate if the goal statement is satisfied in the current state.

2. Compute the **SET** of *pending goals* $\mathcal{G}$, and the **set** of *applicable operators* $\mathcal{A}$.
   - A goal is pending if it is a precondition, not satisfied in the current state, of an operator already in the plan.
   - An operator is applicable when all its preconditions are satisfied in the state.

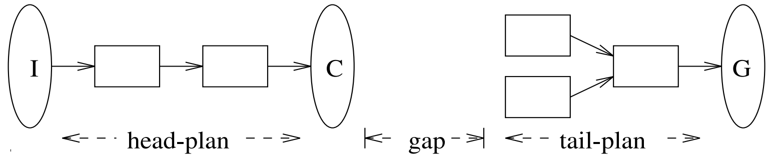
3. Choose a goal G in $\mathcal{G}$ or choose an operator A in $\mathcal{A}$.

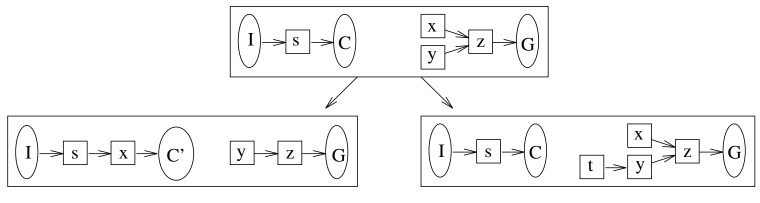
Prodigy4.0 Planning Algorithm

4. If G has been chosen, then
   - *Expand goal* G, i.e., get the set $\mathcal{O}$ of *relevant instantiated operators* that could achieve the goal G,
   - Choose an operator O from $\mathcal{O}$,
   - Go to step 1.

5. If an operator A has been selected as directly applicable, then
   - *Apply* A,
   - Go to step 1.

Prodigy4.0 - Search Representation

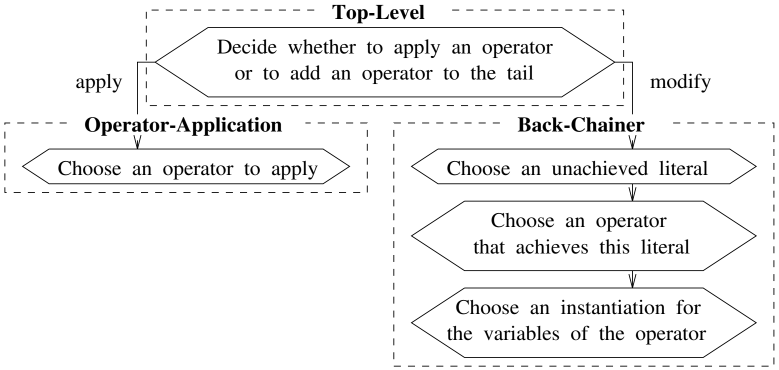




Applying an operator (moving it to the head)

Adding an operator to the tail-plan

Planning Choices



- Planning as search, i.e., a decision-making process – learn search heuristics
- Planning representation – learn efficient, complete, correct domain specifications

Control Rules - Heuristic to Guide Search

Select Rule

If (has-spot <part>) is the current goal
and drill-spot (<part>, <drill>) is the current operator
and (holding-drill-bit <drill-1>) holds in the current state
and <drill-1> is of the type SPOT-DRILL
Then select instantianting <drill> with <drill-1>

Prefer Rule

If (has-hole <part-1>) is a candidate goal
and (has-hole <part-2>) is a candidate goal
and (holding-part <part-1>) holds in the current state
Then prefer the goal (has-hole <part-1>) to (has-hole <part-2>)

Summary - Planning - Lecture 1

- Planning: selecting one sequence of actions (operators) that transform (apply to) an initial state to a final state where the goal statement is true.

- Means-ends analysis: identify and reduce, as soon as possible, *differences* between state and goals.

- Linear planning: backward chaining with means-ends analysis using a stack of goals - potentially efficient, possibly unoptimal, incomplete; GPS.

- Nonlinear planning with means-ends analysis: backward chaining using a set of goals; reason

Summary - Planning - Lecture 1

about *when* "to reduce the differences;" Prodigy4.0.

- Planning as search: control rules to capture heuristics for direct search; learning opportunities.