Planning

Philipp Koehn

30 March 2017
Outline

- Search vs. planning
- STRIPS operators
- Partial-order planning
- The real world
- Conditional planning
- Monitoring and replanning
search vs. planning
Search vs. Planning

- Consider the task *get milk, bananas, and a cordless drill*

- Standard search algorithms seem to fail miserably:

- After-the-fact heuristic/goal test inadequate
Search vs. Planning

• Planning systems do the following

  1. improve action and goal representation to allow selection
  2. divide-and-conquer by **subgoaling**
  3. relax requirement for sequential construction of solutions

• Differences

<table>
<thead>
<tr>
<th></th>
<th>Search</th>
<th>Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>States</strong></td>
<td>Data structures</td>
<td>Logical sentences</td>
</tr>
<tr>
<td><strong>Actions</strong></td>
<td>Program code</td>
<td>Preconditions/outcomes</td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td>Program code</td>
<td>Logical sentence (conjunction)</td>
</tr>
<tr>
<td><strong>Plan</strong></td>
<td>Sequence from $S_0$</td>
<td>Constraints on actions</td>
</tr>
</tbody>
</table>
strips operators
STRIPS Operators

- Tidily arranged actions descriptions, restricted language

\[
\begin{align*}
\text{At}(p) & \land \text{Sells}(p, x) \\
\text{Buy}(x) & \\
\text{Have}(x)
\end{align*}
\]

- **ACTION**: \(\text{Buy}(x)\)
  - **PRECONDITION**: \(\text{At}(p), \text{Sells}(p, x)\)
  - **EFFECT**: \(\text{Have}(x)\)

- Note: this abstracts away many important details!

- Restricted language \(\implies\) efficient algorithm
  - Precondition: conjunction of positive literals
  - Effect: conjunction of literals
partial-order planning
Partially Ordered Plans

- Partially ordered collection of steps with
  - Start step has the initial state description as its effect
  - Finish step has the goal description as its precondition
  - causal links from outcome of one step to precondition of another
  - temporal ordering between pairs of steps

- Open condition = precondition of a step not yet causally linked

- A plan is complete iff every precondition is achieved

- A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it
Example

Start

At(Home)  Sells(HWS,Drill)  Sells(SM,Milk)  Sells(SM,Ban.)

Have(Milk)  At(Home)  Have(Ban.)  Have(Drill)

Finish
Example

\[
\begin{align*}
\text{Start} & \quad \text{At(Home)} \quad \text{Sells(HWS,Drill)} \quad \text{Sells(SM,Milk)} \quad \text{Sells(SM,Ban.)} \\
\text{At(SM)} & \quad \text{Sells(SM,Milk)} \\
\text{Buy(Milk)} & \\
\text{Have(Milk)} & \quad \text{At(Home)} \quad \text{Have(Ban.)} \quad \text{Have(Drill)} \\
\text{Finish} &
\end{align*}
\]
Example

```
Start

At(Home)  Sells(HW, Drill)  Sells(SM, Milk)  Sells(SM, Ban.)
```

```
At(x)

Go(SM)

At(SM)  Sells(SM, Milk)

Buy(Milk)

Have(Milk)  At(Home)  Have(Ban.)  Have(Drill)

Finish
```
Example

Diagram showing the planning process:

1. Start
   - At(Home)
   - Sells(HWS, Drill)
   - Sells(SM, Milk)
   - Sells(SM, Ban.)

2. At(HWS)
   - Sells (HWS, Drill)
   - Buy( Drill)

3. At(x)
   - Go(SM)

4. At(SM)
   - Sells(SM, Milk)
   - Buy(Milk)

5. Have(Milk)
   - At(Home)
   - Have(Ban.)
   - Have( Drill)

6. Finish
Planning Process

- Operators on partial plans
  - add a link from an existing action to an open condition
  - add a step to fulfill an open condition
  - order one step wrt another to remove possible conflicts

- Gradually move from incomplete/vague plans to complete, correct plans

- Backtrack if an open condition is unachievable or if a conflict is unresolvable
**Partially Ordered Plans Algorithm**

**function** POP\((initial, goal, operators)\) **returns** plan

\[
\text{plan} \leftarrow \text{MAKE-MINIMAL-PLAN}(initial, goal)
\]

**loop do**

**if** SOLUTION\(? (plan) \text{ then return plan} \]

\[
S_{\text{need}}, c \leftarrow \text{SELECT-SUBGOAL}(plan)
\]

**CHOOSE-OPERATOR\((plan, operators, S_{\text{need}}, c)\)**

**RESOLVE-THREATS\((plan)\)**

**end**

**function** SELECT-SUBGOAL\((plan)\) **returns** \(S_{\text{need}}, c\)

pick a plan step \(S_{\text{need}}\) from STEPS\((plan)\)

with a precondition \(c\) that has not been achieved

**return** \(S_{\text{need}}, c\)
**Partially Ordered Plans Algorithm**

**procedure** `CHOOSE-OPERATOR(plan, operators, S_{need}, c)`

choose a step $S_{add}$ from `operators` or `STEPS(plan)` that has $c$ as an effect
if there is no such step **then fail**
add the causal link $S_{add} \xrightarrow{c} S_{need}$ to `LINKS(plan)`
add the ordering constraint $S_{add} < S_{need}$ to `ORDERINGS(plan)`
if $S_{add}$ is a newly added step from `operators` **then**
  add $S_{add}$ to `STEPS(plan)`
  add $Start < S_{add} < Finish$ to `ORDERINGS(plan)`

**procedure** `RESOLVE-THREATS(plan)`

for each $S_{threat}$ that threatens a link $S_i \xrightarrow{c} S_j$ in `LINKS(plan)` do
choose either
   
   **Demotion:** Add $S_{threat} < S_i$ to `ORDERINGS(plan)`
   
   **Promotion:** Add $S_j < S_{threat}$ to `ORDERINGS(plan)`

if not `CONSISTENT(plan)` **then fail**
end
A clobberer is a potentially intervening step that destroys the condition achieved by a causal link. E.g., \( Go(\text{Home}) \) clobbers \( At(\text{Supermarket}) \):

- **Demotion**: put before \( Go(\text{Supermarket}) \)
- **Promotion**: put after \( Buy(\text{Milk}) \)
Properties of Partially Ordered Plans

- Nondeterministic algorithm: backtracks at choice points on failure
  - choice of $S_{add}$ to achieve $S_{need}$
  - choice of demotion or promotion for clobberer
  - selection of $S_{need}$ is irrevocable

- Partially Ordered Plans is sound, complete, and systematic (no repetition)

- Extensions for disjunction, universals, negation, conditionals

- Can be made efficient with good heuristics derived from problem description

- Particularly good for problems with many loosely related subgoals
Example: Blocks World

"Sussman anomaly" problem

Start State

Goal State

\[ \begin{align*}
\text{Clear}(x) & \quad \text{On}(x,z) \quad \text{Clear}(y) \\
\text{PutOn}(x,y) & \\
\sim\text{On}(x,z) & \quad \sim\text{Clear}(y) \\
\text{Clear}(z) & \quad \text{On}(x,y) \\
\end{align*} \]

\[ \begin{align*}
\text{Clear}(x) & \quad \text{On}(x,z) \\
\text{PutOnTable}(x) & \\
\sim\text{On}(x,z) & \quad \text{Clear}(z) \quad \text{On}(x,Table) \\
\end{align*} \]

+ several inequality constraints
Example

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

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

Finish
Example
Example

```
START
On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)

PutOn(A,B) clobbers Cl(B) => order after PutOn(B,C)

PutOn(B,C)
On(B,C) On(A,Table) Cl(B)

FINISH
On(A,B) On(A,Table) Cl(B)
```

Philipp Koehn  Artificial Intelligence: Planning  30 March 2017
Example
the real world
The Real World

START

~Flat(Spare) Intact(Spare) Off(Spare)
On(Tire1) Flat(Tire1)

On(x) ~Flat(x)

FINISH

On(x)
Remove(x)
Off(x) ClearHub

Off(x) ClearHub
Puton(x)

Intact(x) Flat(x)
Inflate(x)
~Flat(x)

On(x) ~ClearHub

Off(x) ClearHub
Things Go Wrong

- **Incomplete information**
  - Unknown preconditions, e.g., $\text{Intact}(\text{Spare})$?
  - Disjunctive effects, e.g., $\text{Inflate}(x)$ causes $\text{Inflated}(x) \vee \text{SlowHiss}(x) \vee \text{Burst}(x) \vee \text{BrokenPump} \vee \ldots$

- **Incorrect information**
  - Current state incorrect, e.g., spare NOT intact
  - Missing/incorrect postconditions in operators

- **Qualification problem** can never finish listing all
  - required preconditions of actions
  - possible conditional outcomes of actions
Solutions

- **Conformant or sensorless planning**
  Devise a plan that works regardless of state or outcome
  Such plans may not exist

- **Conditional planning**
  Plan to obtain information (observation actions)
  Subplan for each contingency, e.g.,
  \[\text{Check(Tire1), if Intact(Tire1) then Inflate(Tire1) else CallAAA}\]
  Expensive because it plans for many unlikely cases

- **Monitoring/Replanning**
  Assume normal states, outcomes
  Check progress during execution, replan if necessary
  Unanticipated outcomes may lead to failure (e.g., no AAA card)

⇒ Really need a combination; plan for likely/serious eventualities,
deal with others when they arise, as they must eventually.
Conformant Planning

- Search in space of belief states (sets of possible actual states)
conditional planning
Conditional Planning

- If the world is nondeterministic or partially observable then percepts usually provide information, i.e., split up the belief state
Conditional Planning

- Conditional plans check (any consequence of KB +) percept

- \([\ldots, \text{if } C \text{ then } Plan_A \text{ else } Plan_B, \ldots]\)

- Execution: check \(C\) against current KB, execute “then” or “else”

- Need *some* plan for *every* possible percept
  - game playing: *some* response for *every* opponent move
  - backward chaining: *some* rule such that *every* premise satisfied

- AND–OR tree search (very similar to backward chaining algorithm)
Example

- Double Murphy: sucking or arriving may dirty a clean square
monitoring and replanning
Execution Monitoring

- Plan with Partially Ordered Plans algorithms
- Process plan, one step at a time
- Validate planned conditions against perceived reality
- “Failure” = preconditions of remaining plan not met
- Preconditions of remaining plan
  - all preconditions of remaining steps not achieved by remaining steps
  - all causal links crossing current time point
Responding to Failure

- Run Partially Ordered Plans algorithms again
- Resume Partially Ordered Plans to achieve open conditions from current state
- IPEM (Integrated Planning, Execution, and Monitoring)
  - keep updating \textit{Start} to match current state
  - links from actions replaced by links from \textit{Start} when done
Example
Example
Example

Start
  ↓
At(Home)
  ↓
Go(HWS)
  ↓
At(HWS)  Sells(HWS,Drill)
  ↓
Buy(Drill)
  ↓
At(HWS)
  ↓
Go(SM)
  ↓
At(SM)  Sells(SM,Milk)
  ↓
Buy(Milk)
  ↓
At(SM)  Sells(SM,Ban.)
  ↓
Buy(Ban.)
  ↓
At(SM)
  ↓
Go(Home)
  ↓
Have(Milk)  At(Home)  Have(Ban.)  Have(Quit)
  ↓
Finish

At(HWS)
  ↓
Have(Drill)
  ↓
Sells(SM,Ban.)
  ↓
Sells(SM,Milk)
Example

A flowchart showing a planning sequence:

- **Start**
  - At(Home)
  - Go(HWS)
    - At(HWS)
      - Sells(HWS, Drill)
      - Buy(Drill)
    - Go(SM)
      - At(SM)
      - Sells(SM, Milk)
      - Buy(Milk)
    - Go(Home)
      - At(Home)
      - Have(Home)
      - Have(Ban.)
      - Have(Drill)
      - Sells(SM, Ban.)
      - Sells(SM, Milk)
  - At(SM)
    - Buy(Ban.)
  - Finish

The sequence involves moving from home to the hardware store (HWS) to buy a drill, then to the supermarket (SM) to buy milk and bananas, and finally back home to complete the task.
Example
Example
Emergent Behavior

PRECONDITIONS

START

\( \text{Color}(Chair, \text{Blue}) \) \quad \lnot \text{Have}(\text{Red}) \)

Get(\text{Red})

\text{Have}(\text{Red})

Paint(\text{Red})

\( \text{Color}(Chair, \text{Red}) \)

FINISH

FAILURE RESPONSE

Have(\text{Red})

Fetch more red
Emergent Behavior

```
<table>
<thead>
<tr>
<th>PRECONDITIONS</th>
<th>FAILURE RESPONSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>Extra coat of paint</td>
</tr>
<tr>
<td>Color(Chair,Blue) ~ Have(Red)</td>
<td>Color(Chair,Red)</td>
</tr>
<tr>
<td>Get(Red)</td>
<td></td>
</tr>
<tr>
<td>Have(Red)</td>
<td></td>
</tr>
<tr>
<td>Paint(Red)</td>
<td></td>
</tr>
<tr>
<td>Color(Chair,Red)</td>
<td></td>
</tr>
<tr>
<td>FINISH</td>
<td></td>
</tr>
</tbody>
</table>
```
Emergent Behavior

"Loop until success" behavior emerges from interaction between monitor/replan agent design and uncooperative environment.
Summary

- Planning
  - break down problem into subgoals
  - search for plans for subgoals
  - merge sub-plans

- Defined actions in terms of preconditions and effects

- Partially Ordered Plans algorithm

- Clobbering: need to deal with steps that destroy clausal link in plan

- Real world: incomplete and incorrect information

⇒ conformant or conditional planning, monitoring and replanning