Course on Automated Planning: Transformations

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain
AI Planning: Status

• The good news: classical planning works!
  ▶ Large problems solved very fast (non-optimally)

• Model simple but useful
  ▶ Operators not primitive; can be policies themselves
  ▶ Fast closed-loop replanning able to cope with uncertainty sometimes

• Not so good; limitations:
  ▶ Does not model Uncertainty (no probabilities)
  ▶ Does not deal with Incomplete Information (no sensing)
  ▶ Does not accommodate Preferences (simple cost structure)
  ▶ . . .
Beyond Classical Planning: Two Strategies

• **Top-down:** Develop solver for *more general class of models*; e.g., Markov Decision Processes (MDPs), Partial Observable MDPs (POMDPs), . . .
  
  +: generality  
  -: complexity

• **Bottom-up:** Extend the scope of *current 'classical' solvers*

  +: efficiency  
  -: generality

• We’ll do both, starting with **transformations** for

  ▶ compiling **soft goals** away (planning with preferences)  
  ▶ compiling **uncertainty** away (conformant planning)  
  ▶ compiling **sensing** away (planning with sensing)  
  ▶ doing **plan recognition** (as opposed to plan generation)
Compilation of Soft Goals

- Planning with **soft goals** aimed at plans $\pi$ that maximize **utility**

\[ u(\pi) = \sum_{p \in do(\pi,s_0)} u(p) - \sum_{a \in \pi} c(a) \]

- Actions have **cost** $c(a)$, and soft goals **utility** $u(p)$

- Best plans achieve best **tradeoff** between **action costs** and **utilities**

- Model used in recent planning competitions; **net-benefit track** 2008 IPC

- Yet it turns that soft goals **do not** add expressive power, and can be **compiled away**
Compilation of Soft Goals (cont’d)

- For each soft goal \( p \), create **new hard goal** \( p' \) initially false, and **two new actions**:

  - \( collect(p) \) with precondition \( p \), effect \( p' \) and **cost** 0, and
  - \( forgo(p) \) with an empty precondition, effect \( p' \) and **cost** \( u(p) \)

- Plans \( \pi \) maximize \( u(\pi) \) iff minimize \( c(\pi) = \sum_{a \in \pi} c(a) \) in resulting problem

- Compilation yields better results that native soft goal planners in recent IPC (Keyder & G. 07,09)

<table>
<thead>
<tr>
<th>Domain</th>
<th>IPC6 Net-Benefit Track</th>
<th>Compiled Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gamer</td>
<td>HSP*</td>
</tr>
<tr>
<td>crewplanning(30)</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>elevators (30)</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>openstacks (30)</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>pegsol (30)</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>transport (30)</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>woodworking (30)</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>total</td>
<td>71</td>
<td>49</td>
</tr>
</tbody>
</table>
**Problem:** A robot must move from an **uncertain** $I$ into $G$ with **certainty**, one cell at a time, in a grid $n \times n$

- Problem very much like a classical planning problem except for **uncertain** $I$
- Plans, however, quite different: best **conformant plan must move the robot to a corner first** (localization)
Conformant Planning: Belief State Formulation

- call a set of possible states, a **belief state**
- actions then map a belief state $b$ into a belief state $b_a = \{s' | s' \in F(a, s) \& s \in b\}$
- **conformant problem** becomes a path-finding problem in **belief space**

**Problem:** number of belief state is **doubly exponential** in number of variables.

- **effective representation** of belief states $b$
- **effective heuristic** $h(b)$ for estimating cost in belief space

**Recent alternative:** translate into classical planning . . .
Basic Translation: Move to the 'Knowledge Level'

Given **conformant problem** $P = \langle F, O, I, G \rangle$

- $F$ stands for the fluents in $P$
- $O$ for the operators with effects $C \rightarrow L$
- $I$ for the initial situation (**clauses** over $F$-literals)
- $G$ for the goal situation (set of $F$-literals)

Define **classical problem** $K_0(P) = \langle F', O', I', G' \rangle$ as

- $F' = \{ KL, K\neg L \mid L \in F \}$
- $I' = \{ KL \mid \text{clause } L \in I \}$
- $G' = \{ KL \mid L \in G \}$
- $O' = O$ but preconds $L$ replaced by $KL$, and effects $C \rightarrow L$ replaced by $KC \rightarrow KL$ (**supports**) and $\neg K\neg C \rightarrow \neg K\neg L$ (**cancellation**)

$K_0(P)$ is **sound** but **incomplete**: every classical plan that solves $K_0(P)$ is a conformant plan for $P$, but not vice versa.
Key elements in Complete Translation $K_{T,M}(P)$

• A set $T$ of **tags** $t$: consistent sets of **assumptions** (literals) about the **initial situation** $I$

$$I \not\models \neg t$$

• A set $M$ of **merges** $m$: **valid subsets of tags** ($= \text{DNF}$)

$$I \models \bigvee_{t \in m} t$$

• New (tagged) literals $KL/t$ meaning that $L$ is true if $t$ true initially
A More General Translation $K_{T,M}(P)$

Given **conformant problem** $P = \langle F, O, I, G \rangle$

- $F$ stands for the fluents in $P$
- $O$ for the operators with effects $C \rightarrow L$
- $I$ for the initial situation (**clauses** over $F$-literals)
- $G$ for the goal situation (set of $F$-literals)

define **classical problem** $K_{T,M}(P) = \langle F', O', I', G' \rangle$ as

- $F' = \{ KL/t , K\neg L/t \mid L \in F \text{ and } t \in T \}$
- $I' = \{ KL/t \mid \text{if } I \models t \supset L \}$
- $G' = \{ KL \mid L \in G \}$
- $O' = O$ but preconds $L$ replaced by $KL$, and effects $C \rightarrow L$ replaced by $KC/t \rightarrow KL/t$ (**supports**) and $\neg K\neg C/t \rightarrow \neg K\neg L/t$ (**cancellation**), and **new merge actions**

$$\bigwedge_{t \in m, m \in M} KL/t \rightarrow KL$$

The two **parameters** $T$ and $M$ are the set of **tags** (assumptions) and the set of **merges** (valid sets of assumptions) . . .
Compiling Uncertainty Away: Properties

- General translation scheme $K_{T,M}(P)$ is always sound, and for suitable choice of the sets of tags and merges, it is complete.

- $K_{S0}(P)$ is complete instance of $K_{T,M}(P)$ obtained by setting $T$ to the set of possible initial states of $P$.

- $K_i(P)$ is a polynomial instance of $K_{T,M}(P)$ that is complete for problems with width bounded by $i$.
  
  ▶ Merges for each $L$ in $K_i(P)$ chosen to satisfy $i$ clauses in $I$ relevant to $L$.

- The width of most benchmarks bounded and equal 1!

- This means that such problems can be solved with a classical planner after a polynomial translation (Palacios & G. 07, 09).
Problem: Starting in one of two rightmost cells, get to \( B \); \( A \) & \( B \) observable

\[
\begin{array}{ccc}
A & & B \\
\end{array}
\]

- Contingent Planning
  - A contingent plan is a tree of possible executions, all leading to the goal
  - A contingent plan for the problem: \( \text{R(right), R, R, if } \neg B \text{ then } R \)

- POMDP planning
  - A POMDP policy is mapping of belief states to actions, leading to goal
  - A POMDP policy for problem: If \( Bel \neq B \), then \( R \) \( (2^5 - 1 \text{ Bel’s}) \)

I’ll focus on different solution form: finite state controllers
Finite State Controllers: Example 1

- Starting in $A$, move to $B$ and back to $A$; marks $A$ and $B$ observable.

- This finite-state controller solves the problem

- FSC is compact and general: can add noise, vary distance, etc.

- Heavily used in practice, e.g. video-games and robotics, but written by hand

- The Challenge: How to get these controllers automatically
Finite State Controllers: Example 2

- **Problem** $P$: find **green block** using visual-marker (circle) that can move around one cell at a time (à la Chapman and Ballard)

- **Observables**: Whether cell marked contains a green block (G), non-green block (B), or neither (C); and whether on table (T) or not (–)

- Controller on the right **solves** the problem, and not only that, it’s **compact** and **general**: it applies to **any number of blocks** and **any configuration**!

- Controller obtained by running a **classical planner** over **transformed problem** (Bonet, Palacios, G. 2009)
Some notation: Problem and Finite State Controllers

• **Target problem** $P$ is like a classical problem with **incomplete initial situation** $I$ and some **observable fluents**

• **Finite State Controller** $C$ is a set of tuples $t = \langle q, o, a, q' \rangle$

  $$t = \langle q, o, a, q' \rangle,$$ depicted $q \xrightarrow{o/a} q'$, tells to do action $a$
  when $o$ is observed in controller state $q$ and then to switch to $q'$

• **Finite State Controller** $C$ **solves** $P$ if all state trajectories compatible with $P$ and $C$ reach the goal

  **Question:** how to derive FSC for solving $P$?
Idea: Finite State Controllers as Conformant Plans

- Consider set of possible tuples \( t = \langle q, o, a, q' \rangle \)

- Let \( P' \) be a problem that is like \( P \) but with
  1. no observable fluents
  2. new fluents \( o \) and \( q \) representing possible joint observations \( o \) and \( q \)'s
  3. actions \( b(t) \) replacing the actions \( a \) in \( P \), where for \( t = \langle q, o, a, q' \rangle \),
     \( b(t) \) is like \( a \) but \textbf{conditional on} both \( q \) and \( o \) being true, and resulting in \( q' \).

\[
\textbf{Theorem:} \text{ The finite state controller } C \text{ solves } P \text{ iff } C \text{ is the set of tuples } t \text{ in the actions } b(t) \text{ of a stationary conformant plan for } P' \\
\]

- Corollary: The finite state controller for \( P \) can be obtained with \textbf{classical planner} from further transformation of \( P' \).

- Plan \( \pi \) is \textbf{stationary} when for \( b(t) \) and \( b(t') \) in \( \pi \) for \( t = \langle q, o, a, q' \rangle \) and \( t' = \langle q, o, a', q'' \rangle \), then \( a = a' \) and \( q' = q'' \)
Intuition: Memoryless Controllers

- For simplicity, consider **memoryless** controllers where tuples are $t = \langle o, a \rangle$, meaning to do $a$ when $o$ observed

- In transformed problem $P'$ the actions $a$ in $P$ replaced by $a(o)$ where

  $$a(o) \text{ is like } a \text{ when } o \text{ is true, else is a NO-OP}$$

**Claim:** If the memoryless controller $C = \{ \langle o_i, a_i \rangle \mid i = 1, n \}$ solves $P$ in $m$ steps, the sequence $a_1[o_1], \ldots, a_n[o_n]$ repeated $m$ times is a conformant **plan** for $P'$
Example: FSC for Visual Marker Problem

- **Problem P**: find **green block** using visual-marker (circle) that can move around one cell at a time (à la Chapman or Ballard)

- **Observables**: Whether cell marked contains a green block (G), non-green block (B), or neither (C); and whether on table (T) or not (–)

- **Controller** obtained using a **classical planner** from translation that assumes 2 controller states.

- Controller is **compact** and **general**: it applies to **any number of blocks** and **any configuration**
• Agent can **move** one unit in the four directions

• Possible **targets** are A, B, C, . . .

• Starting in S, he is **observed** to move up twice

• **Where** is he going?
A plan recognition problem defined by triplet $T = \langle \mathcal{G}, \Pi, O \rangle$ where

- $\mathcal{G}$ is the set of possible goals $G$,
- $\Pi(G)$ is the set of possible plans $\pi$ for $G$, $G \subseteq \mathcal{G}$,
- $O$ is an observation sequence $a_1, \ldots, a_n$ where each $a_i$ is an action.

A possible goal $G \in \mathcal{G}$ is plausible if $\exists$ plan $\pi$ in $\Pi(G)$ that satisfies $O$.

An action sequence $\pi$ satisfies $O$ if $O$ is a subsequence of $\pi$. 
(Classical) Plan Recognition over Action Theories

PR over **action theories** similar but with set of plans Π(𝐺) defined **implicitly**:

- A **plan recognition problem** is a triplet \( T = \langle P, G, O \rangle \) where
  - \( P = \langle F, A, I \rangle \) is **planning domain**: fluents \( F \), actions \( A \), init \( I \), **no goal**
  - \( G \) is a set of **possible goals** \( G, G \subseteq F \)
  - \( O \) is the **observation sequence** \( a_1, \ldots, a_n \), all \( a_i \) in \( A \)

If \( \Pi(G) \) stands for '**good plans**' for \( G \) in \( P \) (to be defined), then as before:

- A possible goal \( G \in G \) is **plausible** if there is a plan \( \pi \) in \( \Pi(G) \) that **satisfies** \( O \)
- An action sequence \( \pi \) **satisfies** \( O \) if \( O \) is a subsequence of \( \pi \)

**Our goal:** define the **good plans** and solve the problem with a **classical planner**
Compiling Observations Away

We get rid of obs. $O$ by transforming $P = \langle F, I, A \rangle$ into $P' = \langle F', I', A \rangle$ so that

$\pi$ is a plan for $G$ in $P$ that satisfies $O$ iff $\pi$ is a plan for $G + O$ in $P'$

and

$\pi$ is a plan for $G$ in $P$ that doesn't satisfy $O$ iff $\pi$ is a plan for $G + \overline{O}$ in $P'$

The transformation from $P$ into $P'$ is actually very simple . . .
Compiling Observations Away (cont’d)

• Given \( P = \langle F, I, A \rangle \), the transformed problem is \( P' = \langle F', I', A' \rangle \):

  \( F' = F \cup \{ p_a \mid a \in O \} \),
  \( I' = I \)
  \( A' = A \)

where \( p_a \) is new fluent for the observed action \( a \) in \( A' \) with extra effect:

  \( p_a \), if \( a \) is the first observation in \( O \), and
  \( p_b \rightarrow p_a \), if \( b \) is the action that immediately precedes \( a \) in \( O \).

• The ‘goals’ \( O \) and \( O' \) in \( P' \) are \( p_a \) and \( \neg p_a \) for the last action \( a \) in \( O \)

• The plans \( \pi \) for \( G \) in \( P \) that satisfy/don’t satisfy \( O \) are the plans in \( P' \) for \( G + O/G + O' \) respectively.
Define the set $\Pi(G)$ of ‘good plans’ for $G$ in $P$, as the \textbf{optimal plans} for $G$ in $P$.

- Then $G \in G$ is a \textbf{plausible goal} given observations $O$

  iff there is an \textbf{optimal plan} $\pi$ for $G$ in $P$ that satisfies $O$;
  iff there is an \textbf{optimal plan} $\pi$ for $G$ in $P$ that is a plan for $G + O$ in $P'$;
  iff \textbf{cost} of $G$ in $P$ equal to \textbf{cost} of $G + O$ in $P'$ abbreviated

\[ c_{P'}(G + O) = c_P(G) \]

- It follows that \textbf{plausibility} of $G$ can be \textbf{computed exactly} by calling an \textbf{optimal planner} twice: one for computing $c_{P'}(G + O)$, one for computing $c_P(G)$.

- In turn, this can be \textbf{approximated} by calling \textbf{suboptimal planner} just once (Ramirez & G. 2009). We pursue a \textbf{more general} approach here . . .
Plan Recognition as Planning: A More General Formulation

- Don’t **filter** goals $G$ as **plausible/implausible**,

- Rather **rank** them with a **probability distribution** $P(G|O)$, $G \in \mathcal{G}$

- From **Bayes Rule** $P(G|O) = \alpha P(O|G) P(G)$, where
  - $\alpha$ is a normalizing constant
  - $P(G)$ assumed to be **given** in problem specification
  - $P(O|G)$ defined in terms of **extra cost** to pay for not complying with the observations $O$:

  \[
P(O|G) = \text{function}(c(G + \overline{O}) - c(G + O))\]
Example: Navigation in a Grid Revisited

If $\Delta(G, O) \overset{\text{def}}{=} c(G + \overline{O}) - c(G + O)$:

- For $G = B$, $c(B + O) = c(B) = 4$; $c(B + \overline{O}) = 6$; thus $\Delta(B, O) = 2$
- For $G = C$ or $A$, $c(C + O) = c(C + \overline{O}) = c(C) = 8$; thus $\Delta(C, O) = 0$
- For all others $G$, $c(G + O) = 8$; $c(G + \overline{O}) = c(G) = 4$; thus $\Delta(G, O) = -4$

If $P(O|G')$ is a monotonic function of $\Delta(G, O)$, then

$$P(O|B) > P(O|C) = P(O|A) > P(G), \text{ for } G \not\in \{A, B, C\}$$
Defining the Likelihoods $P(O|G)$

- Assuming Boltzmann distribution and writing $\exp\{x\}$ for $e^x$, likelihoods become

$$P(O|G) \overset{\text{def}}{=} \alpha \exp\{-\beta c(G + O)\}$$

$$P(\overline{O}|G) \overset{\text{def}}{=} \alpha \exp\{-\beta c(G + \overline{O})\}$$

where $\alpha$ is a normalizing constant, and $\beta$ is a positive constant.

- Taking ratio of two equations, it follows that

$$P(O|G)/P(\overline{O}|G) = \exp\{\beta \Delta(G, O)\}$$

and hence

$$P(O|G) = 1/(1 + \exp\{-\beta \Delta(G, O)\}) = \text{sigmoid} (\beta \Delta(G, O))$$
Defining Likelihoods $P(O|G)$ (cont’d)

$$P(O|G) = \text{sigmoid}(\beta \Delta(G, O))$$

$$\Delta(G, O) = c(G + \overline{O}) - c(G + O)$$

E.g.,

$$P(O|G) < P(\overline{O}|G) \quad \text{if} \quad c(G + \overline{O}) < c(G + O)$$

$$P(O|G) = 1 \quad \text{if} \quad c(G + O) < c(G + \overline{O}) = \infty$$
Probabilistic Plan Recognition as Planning: Summary

- A **plan recognition problem** is a tuple $T = \langle P, G, O, Prob \rangle$ where
  - $P$ is a **planning domain** $P = \langle F, I, A \rangle$
  - $G$ is a set of **possible goals** $G$, $G \subseteq F$
  - $O$ is the **observation sequence** $a_1, \ldots, a_n$, $a_i \in O$
  - $Prob$ is **prior distribution** over $G$

- **Posterior distribution** $P(G|O)$ obtained from
  - **Bayes Rule** $P(G|O) = \alpha P(O|G) \text{Prob}(G)$ and
  - **Likelihood** $P(O|G) = \text{sigmoid}\{\beta [c(G + \overline{O}) - c(G + O)]\}$

- Distribution $P(G|O)$ **computed** exactly or approximately:
  - exactly using **optimal planner** for determining $c(G + O)$ and $c(G + \overline{O})$,
  - approximately using **suboptimal planner** for $c(G + O)$ and $c(G + \overline{O})$

- In either case, $2 \cdot |G|$ planner calls are needed.
Example: Noisy Walk

Graph on the left shows ‘noisy walk’ and possible targets; curves on the right show posterior $P(G|O)$ of each possible target $G$ as a function of time.
Summary: Transformations

- **Classical Planning** solved as **path-finding** in state state
  - Most used techniques are **heuristic search** and **SAT**

- **Beyond classical planning**: two approaches
  - **Top-down**: solvers for richer models like MDPs and POMDPs
  - **Bottom-up**: compile non-classical features away

- We have follow second approach with **transformations** to eliminate
  - **soft goals** when planning with preferences
  - **uncertainty** in conformant planning
  - **sensing** for deriving finite-state controllers
  - **observations** for plan recognition

- Other transformations used for **LTL plan constraints**, **control knowledge**, etc.