Planning in propositional logic

Represent transition relations (adjacency matrices) as propositional formulae, and use these formulas in planning algorithms.

1. Find plans with theorem-provers for propositional logic.
2. Use breadth-first search for computing the set of all reachable states (these sets are represented as propositional formulae), and extract plans from the information you have gathered.

Jussi Rintanen

Actions as propositional formulae

$P = \{p_1, \ldots, p_n\}$ = state variables in the current state
$P' = \{p'_1, \ldots, p'_n\}$ = state variables in the successor state

A formula $\phi$ over $P \cup P'$ can be viewed as representing an action, because it can be viewed as a relation over sets of states.

For $n$ state variables a formula (over $2n$ variables) represents an adjacency matrix of size $2^n \times 2^n$.

For $n = 20$, matrix size is $2^{20} \times 2^{20} = 1048576 \times 1048576 \sim 10^{12}$ elements

Jussi Rintanen

Actions as propositional formulae: example

Formula $(p_1 \iff p'_2) \land (p_2 \iff p'_3) \land (p_3 \iff p'_4)$ represents matrix

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Jussi Rintanen

Translating operators into formulae

- Any operator can be translated into a propositional formula.
- Translation is polynomial time, formula has polynomial size.
- Use in planning algorithms. Two main approaches are
  1. Translate problem instance into a formula $\phi$, find a satisfying assignment $v$, read the plan from the assignment $v$. = Planning as Satisfiability
  2. Use formulae as a data structure for representing sets of states, algorithm manipulates these data structures. e.g. BDD-based planning algorithms, (regression)
Translating operators into formulae

1. For operator $o = (z, e)$, $\tau_o$ is the conjunction of $z$ and for every state variable $p \in P$

$$((EPC_p(e) \lor (p \land \neg EPC_{\neg p}(e))) \leftrightarrow p') \land \neg((EPC_p(e) \land EPC_{\neg p}(e)))$$

Planning as satisfiability

1. Encode operator sequences of length 0, 1, 2, ... as formulae $\phi_0$, $\phi_1$, $\phi_2$, ... (see next slide...)
2. Test satisfiability of $\phi_0$, $\phi_1$, $\phi_2$, ...
3. Satisfiable formula corresponds to a plan.

There are very good algorithms for testing satisfiability, and planning this way is often very efficient.

This is also applied in microprocessor verification / intelligent debugging: Intel, IBM, Infineon, Motorola, NEC, ... (Hot topic in model-checking in CAV ⇒ Bounded Model-Checking.)

Translating operators into formulae: example

Consider operator $(A \lor B, ((B \lor C) \land A) \land (\neg C \land \neg A) \land (A \lor B))$.

The corresponding propositional formula is

$$(A \lor B) \land (((B \lor C) \lor (A \land \neg C)) \leftrightarrow A') \land \neg((B \lor C) \land \neg C) \land ((A \lor (B \land \neg C)) \leftrightarrow B') \land \neg(A \land C) \land ((1 \lor C \land \neg C) \leftrightarrow C') \land \neg(1 \land C)$$

$$(A \lor B) \land (((B \lor C) \lor (A \land C)) \leftrightarrow A') \land \neg((B \lor C) \land \neg C) \land ((A \lor B) \leftrightarrow B') \land (C \leftrightarrow C')$$

Planning as satisfiability: encoding 1

Let $\langle P, I, O, G \rangle$ be a problem instance.

Let $R_1(P^0, P^1)$ denote $\bigvee_{o \in O} \tau_o$ where $P = \{p_1, \ldots, p_n\}$ and $P' = \{p'_1, \ldots, p'_n\}$ are respectively replaced by $P^0 = \{p^0_1, \ldots, p^0_n\}$ and $P^1 = \{p^1_1, \ldots, p^1_n\}$.

Finding plans of length $t$ is encoded as

$$l^0 \land R_1(P^0, P^1) \land R_1(P^1, P^2) \land \cdots \land R_1(P^{t-1}, P^t) \land G^t$$

Here $l^0 = \wedge \{p^0 | p \in P, I(p) = 1\} \cup \{p^0 | p \in P, I(p) = 0\}$ and $G^t$ is $G$ with propositions $p$ replaced by $p^t$. 
Planning as satisfiability: encoding 1, example

$I \models A \land B$, \quad $G = (A \land \neg B) \lor (\neg A \land B)$,

$a_1 = \langle \top, (A \rightarrow \neg A) \land (\neg A \rightarrow A) \rangle$, \quad $a_2 = \langle \top, (B \rightarrow \neg B) \land (\neg B \rightarrow B) \rangle$,

plan length 3

\[(A^0 \land B^0) \land ((A^0 \leftrightarrow A^1) \land (B^0 \leftrightarrow \neg B^1)) \lor ((A^1 \leftrightarrow \neg A^1) \land (B^0 \leftrightarrow B^1)) \land ((A^1 \leftrightarrow A^2) \land (B^1 \leftrightarrow \neg B^2)) \lor ((A^1 \leftrightarrow \neg A^1) \land (B^1 \leftrightarrow B^2)) \land ((A^2 \leftrightarrow \neg A^2) \land (B^2 \leftrightarrow B^3)) \land ((A^3 \land \neg B^3) \lor (\neg A^3 \land B^3))\]

One valuation that satisfies the formula:

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>$A^i$</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>$B^i$</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

1. There are several valuations/plans
2. Also plans of length 1 exists (just ignore time points 2 and 3!)
3. Plans of length 2 do not exist!

Q: Satisfiability in propositional logic is NP-complete, but testing existence of plans is PSPACE-complete. How is it possible to do this translation from planning to satisfiability?

A: The translation is polynomial time in the size of the problem instance and in the plan length. For exponentially long plans the translation takes exponential time.

However, in practice plans are often short.

Planning as satisfiability: encoding 2, explanatory frame axioms in $\mathcal{R}_2(P, P')$

Let $p \in P$ be one of the state variables.

\[
(\neg p \land p') \rightarrow ((a_1 \land EPC_p(e_1)) \lor \cdots \lor (a_n \land EPC_p(e_n)))
\]

\[
(p \land \neg p') \rightarrow ((a_1 \land EPC_{\neg p}(e_1)) \lor \cdots \lor (a_n \land EPC_{\neg p}(e_n)))
\]
Planning as satisfiability: $\mathcal{R}_2(P, P')$, effect axioms

$\alpha_i = (z, e)$ may affect the state variables as follows.

\[
\begin{align*}
(\alpha_i \land EPC_{p_1}(e)) & \rightarrow p'_1 \\
(\alpha_i \land EPC_{p_1}(e)) & \rightarrow \neg p'_1 \\
& \vdots \\
(\alpha_i \land EPC_{p_n}(e)) & \rightarrow p'_n \\
(\alpha_i \land EPC_{p_n}(e)) & \rightarrow \neg p'_n
\end{align*}
\]

Also, the precondition of the operator has to be true:

\[
\alpha_i \rightarrow z
\]

Planning as satisfiability: $\mathcal{R}_2(P, P')$, example

$\alpha_1 = \langle \neg LAMP_1, LAMP_1 \rangle$, $\alpha_2 = \langle \neg LAMP_2, LAMP_2 \rangle$

\[
\begin{align*}
(\neg LAMP_1 \land LAMP_1') & \rightarrow \alpha_1 \\
(LAMP_1 \land \neg LAMP_1') & \rightarrow \bot \\
(\neg LAMP_2 \land LAMP_2') & \rightarrow \alpha_2 \\
(LAMP_2 \land \neg LAMP_2') & \rightarrow \bot \\
\alpha_1 & \rightarrow LAMP_1' \\
\alpha_1 & \rightarrow \neg LAMP_1 \\
\alpha_2 & \rightarrow LAMP_2' \\
\alpha_2 & \rightarrow \neg LAMP_2
\end{align*}
\]

Planning as satisfiability: encoding 2

To obtain valid plans only one operator may be applied at a time: for every $\alpha_i, \alpha_j \in O$ such that $i \neq j$, we have

\[
\neg (\alpha_i \land \alpha_j)
\]

in $\mathcal{R}_2(P, P')$.

Planning as satisfiability: encoding 2

Plans of length $t$ are encoded exactly like with $\mathcal{R}_1(P, P')$:

\[
i^0 \land \mathcal{R}_2(P^0, P^1) \land \mathcal{R}_2(P^1, P^2) \land \cdots \land \mathcal{R}_2(P^{t-1}, P^t) \land G^t
\]

Reading the plan from a satisfying assignment $v$:

$\alpha_i$ is the operator at time point $t$ if and only if $v(\alpha_i^t) = 1$. 