Lecture 7: Planning in propositional logic II

- Representation of matrix operations in propositional logic
- Binary decision diagrams BDDs
- Algorithms based on BDDs
## Normal forms for propositional formulae

<table>
<thead>
<tr>
<th></th>
<th>$\lor$</th>
<th>$\land$</th>
<th>$\neg$</th>
<th>$\phi \in \text{TAUT}?$</th>
<th>$\phi \in \text{SAT}?$</th>
<th>$\phi \equiv \phi'$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>circuits</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>co-NP-hard</td>
<td>NP-hard</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>formulae</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>co-NP-hard</td>
<td>NP-hard</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>DNF</td>
<td>P</td>
<td>exp</td>
<td>exp</td>
<td>co-NP-hard</td>
<td>P</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>CNF</td>
<td>exp</td>
<td>P</td>
<td>exp</td>
<td>P</td>
<td>NP-hard</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>DNNF</td>
<td>P</td>
<td>exp</td>
<td>exp</td>
<td>co-NP-hard</td>
<td>P</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>BDD</td>
<td>exp</td>
<td>exp</td>
<td>P</td>
<td>P</td>
<td>NP-hard</td>
<td>P</td>
</tr>
</tbody>
</table>

Jussi Rintanen

November 6, AI Planning 2/15
BDD-based planning algorithms

- Use propositional formulae (usually: binary decision diagrams) for representing the adjacency matrices.

- Compute sets of states reachable by $i$ operators by matrix multiplication.

- Applicable to big state spaces.

- Sometimes competitive with other types of algorithms.
## BDD-based planning algorithms: matrices as formulae/BDD

<table>
<thead>
<tr>
<th>matrices</th>
<th>formulas</th>
<th>sets of states</th>
</tr>
</thead>
<tbody>
<tr>
<td>vector $V_{1\times n}$</td>
<td>formula over $P$</td>
<td>set of states</td>
</tr>
<tr>
<td>matrix $M_{n\times n}$</td>
<td>formula over $P \cup P'$</td>
<td>transition relation</td>
</tr>
<tr>
<td>$M_{n\times n} \times N_{n\times n}$</td>
<td>$\exists P'.(\phi(P, P') \land \psi(P', P''))$</td>
<td>sequential composition</td>
</tr>
<tr>
<td>$S_{1\times n} \times M_{n\times n}$</td>
<td>$\exists P.\left(\phi(P) \land \psi(P, P')\right)$</td>
<td>successor states of $S$</td>
</tr>
<tr>
<td>$S_{1\times n} + S'_{1\times n}$</td>
<td>$\phi \lor \psi$</td>
<td>set union</td>
</tr>
<tr>
<td></td>
<td>$\phi \land \psi$</td>
<td>set intersection</td>
</tr>
</tbody>
</table>
BDD-based planning algorithms: idea

$$
\iota^0 \land \mathcal{R}_1(P^0, P^1) \land \mathcal{R}_1(P^1, P^2) \land \cdots \land \mathcal{R}_1(P^{n-1}, P^n) \land G^n
$$

corresponds to the $n$-fold product of matrices $\mathcal{R}_1(P, P')$: 

$$
\iota^0 \times (\mathcal{R}_1(P^0, P^1) \times \mathcal{R}_1(P^1, P^2) \times \cdots \times \mathcal{R}_1(P^{n-1}, P^n)) \times G^n
$$

and seeing whether one state in $G$ is reachable from $I$. 

Jussi Rintanen  
November 6, AI Planning  5/15
Matrix multiplication: renaming, $\exists$-abstraction

- Renaming substitution: replace propositions $p_1, \ldots, p_n$ in $\phi$ respectively by $p'_1, \ldots, p'_n$

  \[ \phi[p'_1/p_1, p'_2/p_2, \ldots, p'_n/p_n] \]

- Existential abstraction $\exists p. \phi$ is defined as

  \[ \phi[\top/p] \lor \phi[\bot/p]. \]
Matrix multiplication: $\exists$-abstraction, example

\[\exists B. ((A \to B) \land (B \to C')) \equiv C \lor \neg A \equiv A \to C\]

\[\exists AB. (A \lor B) = \exists B. (\top \lor B) \lor (\bot \lor B) = ((\top \lor \top) \lor (\bot \lor \top)) \lor ((\top \lor \bot) \lor (\bot \lor \bot)) \equiv workday\]

$\exists$-abstraction is sometimes called *forgetting*: abstracted facts are forgotten, everything else is exactly remembered.

\[\exists \text{monday}. (\text{monday} \land (\text{monday} \to \text{workday}))) = \text{workday}\]
Matrix multiplication: definition

Let $\phi$ be a formula over $P \cup P'$ and $\psi$ be a formula over $P' \cup P''$. Now matrix product of matrices corresponding to $\phi$ and $\psi'$ is

$$\exists P'. \phi \land \psi.$$ 

($\phi \land \psi$ alone is the relational product of $\phi$ and $\psi$.)

(Q: is there a valuation of $P'$ “between” valuations of $P$ and $P''$.)
Matrix multiplication: example

\[\phi = A \leftrightarrow \neg A' \text{ (reverse truth-value of } A)\]

\[\psi = A' \leftrightarrow A'' \text{ (do nothing)}\]

The sequential composition of these actions is

\[\exists A'. \phi \land \psi = ((A \leftrightarrow \neg \top) \land (\top \leftrightarrow A'')) \lor ((A \leftrightarrow \neg \bot) \land (\bot \leftrightarrow A''))\]

\[\equiv ((A \leftrightarrow \bot) \land (\top \leftrightarrow A'')) \lor ((A \leftrightarrow \top) \land (\bot \leftrightarrow A''))\]

\[\equiv A \leftrightarrow \neg A''\]
A planning algorithm: version 1

\[ \iota = \bigwedge \{ p \mid p \in P, I(p) = 1 \} \cup \{ \neg p \mid p \in P, I(p) = 0 \} \]

Compute \( n \)-fold matrix products of the transition relation, and test whether \( G \) can be reached from \( I \) in \( n \) time steps:

\[ R := \bigwedge \{ p \leftrightarrow p' \mid p \in P \}; \]

\[ \text{old} R := \top; \]

\[ \text{WHILE} \ \text{old} R \not\equiv R \ \text{AND} \ \iota \land R \land G' \not\in \text{SAT}, \ \text{DO} \]

\[ \text{old} R := R; \]

\[ R := R \lor (\exists P'(R \land R_1(P', P''))) [p'_1/p''_1, p'_2/p''_2, \ldots, p'_n/p''_n]; \]

\[ \text{END WHILE}; \]

\[ \text{IF} \ \iota \land R \land G' \in \text{SAT} \ \text{THEN} \ \text{plan was found}; \]
A planning algorithm: version 2

Compute sets of states reachable in $n$ time steps from $I$ and test whether $G$ intersects these sets:

\[
S := \emptyset; \\
\text{old}S := \top; \\
\text{WHILE old}S \neq S \text{ AND } S \land G \not\in \text{SAT, DO} \\
\quad \text{old}S := S; \\
\quad S := S \lor ((\exists P.(S \land R_1(P, P')))[[p_1/p_1', p_2/p_2', \ldots, p_n/p_n']]); \\
\text{END WHILE;} \\
\text{IF } S \land G \in \text{SAT THEN plan was found;}
\]
Binary decision diagrams

3-place connective *if-then-else*:

\[
\text{ite}(p, \phi_1, \phi_2) = (p \land \phi_1) \lor (\neg p \land \phi_2)
\]

Shannon expansion:

\[
\phi \equiv (p \land \phi[\top/p]) \lor (\neg p \land \phi[\bot/p]) = \text{ite}(p, \phi[\top/p], \phi[\bot/p])
\]
Binary decision diagrams: example

Construct BDD with variable ordering A, B, C:

\[(A \lor B) \land (B \lor C)\]
\[\equiv \text{ite}(A, (\top \lor B) \land (B \lor C), (\bot \lor B) \land (B \lor C))\]
\[\equiv \text{ite}(A, B \lor C, B)\]
\[\equiv \text{ite}(A, \text{ite}(B, \top \lor C, \bot \lor C), \text{ite}(B, \top, \bot))\]
\[\equiv \text{ite}(A, \text{ite}(B, \top, C), \text{ite}(B, \top, \bot))\]
\[\equiv \text{ite}(A, \text{ite}(B, \top, \text{ite}(C, \top, \bot)), \text{ite}(B, \top, \bot))\]
Later in the course we will just talk about unions, intersections and differences of sets of states as well as products of adjacency matrices without discussing the implementation of these operations as propositional formulae (in DNNF, as BDDs.)
## Satisfiability planning vs. BDD-based algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Size of $\mathcal{R}_1(P, P')$</th>
<th>Runtime vs. Plan Length $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfiability planning</td>
<td>Not a problem</td>
<td>Exponential on $n$</td>
</tr>
<tr>
<td>BDDs</td>
<td>Major problem</td>
<td>Much less dependent on $n$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Critical Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfiability planning</td>
<td>Runtime</td>
</tr>
<tr>
<td>BDDs</td>
<td>Memory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Types of Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfiability planning</td>
<td>Lots of state variables, short plans</td>
</tr>
<tr>
<td>BDDs</td>
<td>Fewer state variables, long plans</td>
</tr>
</tbody>
</table>