## 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.



## 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



## Actions as propositional formulae: example

Formula $(p_1 \leftrightarrow p'_2) \wedge (p_2 \leftrightarrow p'_3) \wedge (p_3 \leftrightarrow p'_1)$ represents matrix

| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 010 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 011 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 100 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 101 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 110 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |



## 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 $o = \langle z, e\rangle$, $\tau_o$ is the conjunction of $z$ and for every state variable $p \in P$

$$((\mathit{EPC}_p(e) \vee (p \wedge \neg \mathit{EPC}_{\neg p}(e))) \leftrightarrow p') \wedge \neg(\mathit{EPC}_p(e) \wedge \mathit{EPC}_{\neg p}(e)).$$



## Translating operators into formulae: example

Consider operator $\langle A \vee B, ((B \vee C) \rhd A) \wedge (\neg C \rhd \neg A) \wedge (A \rhd B)\rangle$.

The corresponding propositional formula is

$$\begin{array}{ll}(A \vee B) & \wedge(((B \vee C) \vee (A \wedge \neg\neg C)) \leftrightarrow A') \wedge \neg((B \vee C) \wedge \neg C) \\ & \wedge((A \vee (B \wedge \neg\bot)) \leftrightarrow B') \wedge \neg(A \wedge \bot) \\ & \wedge((\bot \vee (C \wedge \neg\bot)) \leftrightarrow C') \wedge \neg(\bot \wedge \bot) \\ \equiv & \\ (A \vee B) & \wedge(((B \vee C) \vee (A \wedge C)) \leftrightarrow A') \wedge \neg((B \vee C) \wedge \neg C) \\ & \wedge((A \vee B) \leftrightarrow B') \\ & \wedge(C \leftrightarrow C')\end{array}$$



## Planning as satisfiability

1. Encode operator sequences of length 0, 1, 2, ... as formulae $\phi_0$, $\phi_1$, $\phi_2$, $\ldots$ (see next slide...)
2. Test satisfiability of $\phi_0$, $\phi_1$, $\phi_2$, $\ldots$
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 $\Rightarrow$ *Bounded Model-Checking*.)



## Planning as satisfiability: encoding 1

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

Let $\mathcal{R}_1(P^0, P^1)$ 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

$$\iota^0 \wedge \mathcal{R}_1(P^0, P^1) \wedge \mathcal{R}_1(P^1, P^2) \wedge \cdots \wedge \mathcal{R}_1(P^{t-1}, P^t) \wedge G^t.$$

Here $\iota^0 = \bigwedge\{p^0 | p \in P, I(p) = 1\} \cup \{\neg p^0 | p \in P, I(p) = 0\}$ and $G^t$ is $G$ with propositions $p$ replaced by $p^t$.

## Planning as satisfiability: encoding 1, example

$I \models A \wedge B$, $\quad G = (A \wedge \neg B) \vee (\neg A \wedge B)$,
$o_1 = \langle \top, (A \rhd \neg A) \wedge (\neg A \rhd A) \rangle$, $o_2 = \langle \top, (B \rhd \neg B) \wedge (\neg B \rhd B) \rangle$,
plan length 3

$$
\begin{array}{l}
(A^0 \wedge B^0) \\
\wedge(((A^0 \leftrightarrow A^1) \wedge (B^0 \leftrightarrow \neg B^1)) \vee ((A^0 \leftrightarrow \neg A^1) \wedge (B^0 \leftrightarrow B^1))) \\
\wedge(((A^1 \leftrightarrow A^2) \wedge (B^1 \leftrightarrow \neg B^2)) \vee ((A^1 \leftrightarrow \neg A^2) \wedge (B^1 \leftrightarrow B^2))) \\
\wedge(((A^2 \leftrightarrow A^3) \wedge (B^2 \leftrightarrow \neg B^3)) \vee ((A^2 \leftrightarrow \neg A^3) \wedge (B^2 \leftrightarrow B^3))) \\
\wedge((A^3 \wedge \neg B^3) \vee (\neg A^3 \wedge B^3))
\end{array}
$$



## Planning as satisfiability: encoding 1, example

One valuation that satisfies the formula:

| | time $i$ | | | |
|---|---|---|---|---|
| | 0 | 1 | 2 | 3 |
| $A^i$ | 1 | 0 | 0 | 0 |
| $B^i$ | 1 | 1 | 0 | 1 |

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.

$$
\begin{array}{r}
(\neg p \wedge p') \rightarrow ((o_1 \wedge \mathit{EPC}_p(e_1)) \vee \cdots \vee (o_n \wedge \mathit{EPC}_p(e_n))) \\
(p \wedge \neg p') \rightarrow ((o_1 \wedge \mathit{EPC}_{\neg p}(e_1)) \vee \cdots \vee (o_n \wedge \mathit{EPC}_{\neg p}(e_n)))
\end{array}
$$

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

$o_i = \langle z, e \rangle$ may affect the state variables as follows.

$$\begin{aligned}
(o_i \wedge \mathit{EPC}_{p_1}(e)) &\rightarrow p_1' \\
(o_i \wedge \mathit{EPC}_{\neg p_1}(e)) &\rightarrow \neg p_1' \\
&\vdots \\
(o_i \wedge \mathit{EPC}_{p_n}(e)) &\rightarrow p_n' \\
(o_i \wedge \mathit{EPC}_{\neg p_n}(e)) &\rightarrow \neg p_n'
\end{aligned}$$

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

$$o_i \rightarrow z$$



## Planning as satisfiability: encoding 2

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

$$\neg(o_i \wedge o_j)$$

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



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

$o_1 = \langle \neg LAMP1, LAMP1 \rangle$, $o_2 = \langle \neg LAMP2, LAMP2 \rangle$

$$\begin{aligned}
&(\neg LAMP1 \wedge LAMP1') \rightarrow o_1 \\
&(LAMP1 \wedge \neg LAMP1') \rightarrow \bot \\
&(\neg LAMP2 \wedge LAMP2') \rightarrow o_2 \\
&(LAMP2 \wedge \neg LAMP2') \rightarrow \bot \\
&o_1 \rightarrow LAMP1' \\
&o_1 \rightarrow \neg LAMP1 \\
&o_2 \rightarrow LAMP2' \\
&o_2 \rightarrow \neg LAMP2
\end{aligned}$$



## Planning as satisfiability: encoding 2

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

$$\iota^0 \wedge \mathcal{R}_2(P^0, P^1) \wedge \mathcal{R}_2(P^1, P^2) \wedge \cdots \wedge \mathcal{R}_2(P^{t-1}, P^t) \wedge G^t$$

Reading the plan from a satisfying assignment $v$:

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