OBDD-based planning algorithms

- Represent adjacency matrices as propositional formulae (usually: binary decision diagrams).
- Compute sets of states reachable by i operators by matrix multiplication on *propositional formulae*.
- In some cases scales up to much bigger matrices than what can be represented as conventional arrays or representations of sparse matrices.

OBDD-based planning algorithms: matrices as formulae/OBDD

matrices	formulas	sets of states
vector $V_{1\times n}$	formula over P	set of states
matrix $M_{n\times n}$	formula over $P \cup P'$	transition relation
$M_{n\times n} \times N_{n\times n}$	$\exists P'.(\phi(P,P') \wedge \psi(P',P''))$	sequential composition
$S_{1\times n} \times M_{n\times n}$	$\exists P.(\phi(P) \wedge \psi(P,P'))$	successor states of S
$M_{n\times n} \times (S_{1\times n})^T$	$\exists P'.(\phi(P) \wedge \psi(P,P'))$	predecessor states of S
$S_{1\times n} + S'_{1\times n}$	$\phi \vee \psi$	set union
	$\phi \wedge \psi$	set intersection

OBDD-based planning algorithms: idea

Same formulae that we used earlier:

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

The conjunction of the formulae $\mathcal{R}_1(P^i,P^{i+1})$ corresponds to reachability by n steps and in terms of matrices the n-fold matric product:

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

Renaming and $\exists$-abstraction

- Renaming substitution: replace propositions $p_1,\ldots,p_n$ in ϕ 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

$$\phi[\top/p] \vee \phi[\bot/p].$$

∃-abstraction, example

$$\exists B.((A \rightarrow B) \wedge (B \rightarrow C))$$
$$= ((A \rightarrow \top) \wedge (\top \rightarrow C)) \vee ((A \rightarrow \bot) \wedge (\bot \rightarrow C))$$
$$\equiv C \vee \neg A \equiv A \rightarrow C$$

$$\exists AB.(A \vee B) = \exists B.(\top \vee B) \vee (\bot \vee B)$$
$$= ((\top \vee \top) \vee (\bot \vee \top)) \vee ((\top \vee \bot) \vee (\bot \vee \bot))$$

∃-abstraction is sometimes called *forgetting*:

$$\exists \text{mon}.\exists \text{tue}.((\text{mon} \vee \text{tue}) \wedge (\text{mon} \rightarrow \text{work}) \wedge (\text{tue} \rightarrow \text{work}))$$
$$\equiv \exists \text{tue}.((\text{work} \wedge (\text{tue} \rightarrow \text{work})) \vee (\text{tue} \wedge (\text{tue} \rightarrow \text{work}))) \equiv \text{work}$$

Matrix multiplication for formulae: definition

Let ϕ be a formula over $P \cup P'$ and ψ be a formula over $P' \cup P''$.

Now product of matrices corresponding to ϕ and ψ' is

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

($\phi \wedge \psi$ alone is the relational product of ϕ and ψ.)

(Q: Is there a valuation of P' "between" valuations of P and P''?)

Matrix multiplication: example 1

$\phi = A \leftrightarrow \neg A'$ (reverse truth-value of A)

$\psi = A' \leftrightarrow A''$ (do nothing)

The sequential composition of these actions is

$$\begin{aligned} \exists A'.\phi \wedge \psi &= ((A \leftrightarrow \neg\top) \wedge (\top \leftrightarrow A'')) \vee ((A \leftrightarrow \neg\bot) \wedge (\bot \leftrightarrow A'')) \\ &\equiv ((A \leftrightarrow \bot) \wedge (\top \leftrightarrow A'')) \vee ((A \leftrightarrow \top) \wedge (\bot \leftrightarrow A'')) \\ &\equiv A \leftrightarrow \neg A'' \end{aligned}$$

Matrix multiplication: example 2

Multiply $(\neg A \leftrightarrow A') \wedge (\neg B \leftrightarrow B')$ and $(A' \leftrightarrow B'') \wedge (B' \leftrightarrow A'')$:

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

This is $\exists A'.\exists B'.(\neg A \leftrightarrow A') \wedge (\neg B \leftrightarrow B') \wedge (A' \leftrightarrow B'') \wedge (B' \leftrightarrow A'')$.

Matrix multiplication: properties

1. Abstracting one variable takes polynomial time on the size of the formula.
2. Abstracting one variable *may double the size* of the formula.

   $\Longrightarrow$ Abstracting n variables may increase the size by factor 2^n.

   $\Longrightarrow$ Not in general feasible if formulae cannot be simplified.