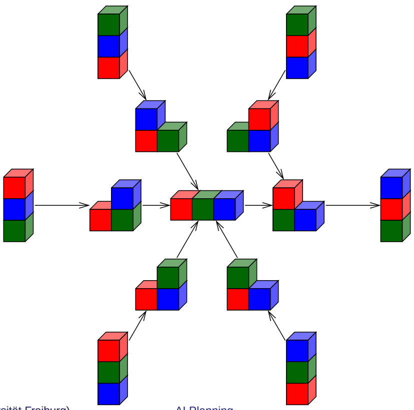
Nondeterministic planning (May 25, 2005)

Plans

1. Memoryless plans map a state/an observation to an operator. We use this definition of plans for fully observable problems only.
2. Conditional plans generalize memoryless plans. They are needed for problems without full observability.
   - The state of the execution of a conditional plan depends on observations on earlier execution steps.
   - The state of the execution = a primitive form of memory.
   - The operator to be executed depends on the state of the execution.

Memoryless plans
Example



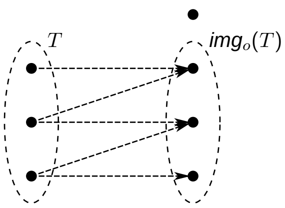
Memoryless plans
Definition

Definition
Let S be the set of all states.
A memoryless plan is a partial function $\pi : S \to O$.

Execution of a memoryless plan
1. Determine the current state s (full observability!!!).
2. If $\pi(s)$ is not defined then terminate execution. If the objective is to reach a goal state, then $\pi(s)$ is not defined if s is a goal state so that the execution terminates.
3. Execute action $\pi(s)$.
4. Goto step 1.

Images

Image
The image of a set T of states with respect to an operator o is the set of those states that can be reached by executing o in a state in T.



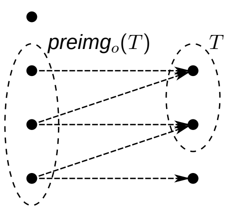
Images
Formal definition

Definition (Image of a state)
$img_o(s) = \{s' \in S | sos'\}$

Definition (Image of a set of states)
$img_o(T) = \bigcup_{s \in T} img_o(s)$

Preimages

Weak preimage
The preimage of a set T of states with respect to an operator o is the set of those states from which a state in T can be reached by executing o.



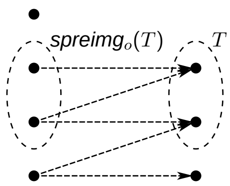
Preimages
Formal definition

Definition (Weak preimage of a state)
$preimg_o(s') = \{s \in S | sos'\}$

Definition (Weak preimage of a set of states)
$preimg_o(T) = \bigcup_{s \in T} preimg_o(s)$.

Strong preimages

Strong preimage
The strong preimage of a set T of states with respect to an operator o is the set of those states from which a state in T is always reached when executing o.



Strong preimages
Formal definition

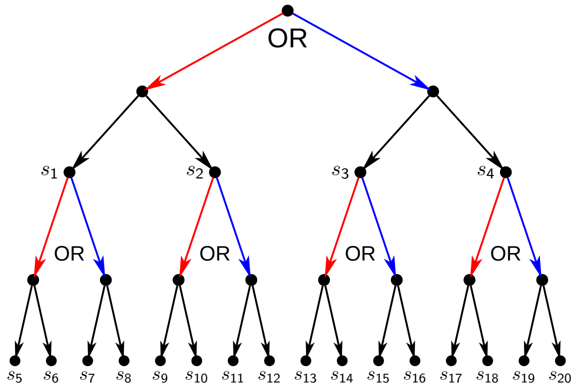
Definition (Strong preimage of a set of states)
$spreimg_o(T) = \{s \in S | s' \in T, sos', img_o(s) \subseteq T\}$

Algorithms for fully observable problems

1. Heuristic search (forward)
   Nondeterministic planning can be viewed as AND-OR search.
   OR nodes: Choice between operators
   AND nodes: Nondeterministically reached state
   Heuristic AND-OR search algorithms: AO*, ...
2. Dynamic programming (backward)
   Idea Compute operator/distance/value for a state based on the operators/distances/values of its all successor states.
   2.1 0 actions needed for goal states.
   2.2 If states with i actions to goals are known, states with $\leq i+1$ actions to goals can be easily identified.

   Automatic reuse of already found plan suffixes.

AND-OR search



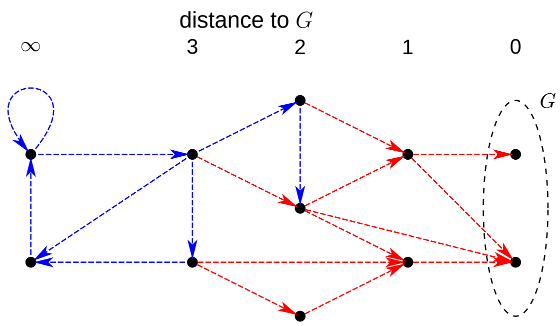
Dynamic programming

Planning by dynamic programming
If for all successors of state s with respect to operator o a plan exists, assign operator o to s.

Base case $i = 0$: In goal states there is nothing to do.
Inductive case $i \geq 1$: If there is $o \in O$ such that for all $s' \in img_o(s)$ s' is a goal state or $\pi(s')$ was assigned on iteration $i-1$, then assign $\pi(s) = o$.

Connection to distances
If s is assigned a value on iteration $i \geq 1$, then the backward distance of s is i.
The dynamic programming algorithm essentially computes the backward distances of states.

Backward distances
Example



Backward distances
Definition of distance sets

Definition
Let G be a set of states and O a set of operators. Define the backward distance sets D_i^{bwd} for G, O that consist of those states for which there is a guarantee of reaching a state in G with at most i operator applications.

$$\begin{aligned} D_0^{bwd} &= G \\ D_i^{bwd} &= D_{i-1}^{bwd} \cup \textstyle\bigcup_{o \in O} spreimg_o(D_{i-1}^{bwd}) \text{ for all } i \geq 1 \end{aligned}$$

Backward distances
Definition

Definition
Let G be as set of states and O a set of operators, and let $D_0^{bwd}, D_1^{bwd}, \ldots$ be the backward distance sets for G and O. Then the backward distance from a state s to G is

$$\delta_G^{bwd}(s) = \begin{cases} 0 \text{ if } s \in G \\ i \text{ if } s \in D_i^{bwd} \setminus D_{i-1}^{bwd} \end{cases}$$

If $s \notin D_i^{bwd}$ for all $i \geq 0$ then $\delta_G^{bwd}(s) = \infty$.

Construction of a plan based on distances

Extraction of a plan from distance sets

1. Let $S' \subseteq S$ be those states having a finite backward distance.
2. Let s be a state with distance $i = \delta_G^{bwd}(s) \geq 1$.
3. Assign to $\pi(s)$ any operator $o \in O$ such that $img_o(s) \subseteq D_{i-1}^{bwd}$.
   Hence o decreases the backward distance by at least one.

The plan π solves the planning problem for $\langle S, I, O, G, P\rangle$ iff $I \subseteq S'$.

Making the algorithm a logic-based algorithm

- An algorithm that represents the states explicitly is feasible for transition systems with at most 10^6 or 10^7 states.
- For planning with bigger transition systems structural properties of the transition system have to be taken advantage of.
- Representing state sets as propositional formulae often allow taking advantage of the structural properties: a formula that represents a set of states or a transition relation that has certain regularities may be very small in comparison to the set or relation.

Making the algorithm a logic-based algorithm

- We use a formula ϕ as a data structure for representing the set $\{s \in S | s \models \phi\}$.
- We show that regression $regr_o^{nd}(\phi)$ for nondeterministic operators is one way of computing strong preimages.
- We present general techniques for computing images, preimages and strong preimages of sets of states represented as formulae.
- Many of the algorithms presented later in the lecture can be lifted to use a logic-based representation, thereby expanding their range of applicability to much bigger transition systems.

Regression for nondeterministic operators
Definition

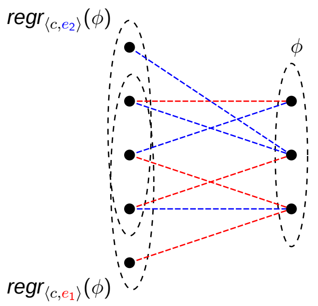
We can easily generalize our regression operation for deterministic operators to regression for nondeterministic operators of a restricted syntactic form.

Definition (Regression for nondeterministic operators)
Let ϕ be a propositional formula and $o = \langle c, e_1 | \cdots | e_n\rangle$ an operator where $e_1, \ldots, e_n$ are deterministic. Define

$$regr_o^{nd}(\phi) = regr_{\langle c,e_1\rangle}(\phi) \wedge \cdots \wedge regr_{\langle c,e_n\rangle}(\phi).$$

Regression for nondeterministic operators
Illustration

$$regr_{\langle c,(e_1|e_2)\rangle}(\phi) = regr_{\langle c,e_1\rangle}(\phi) \wedge regr_{\langle c,e_2\rangle}(\phi)$$



Regression for nondeterministic operators
Correctness

Theorem
Let ϕ be a formula over A, o an operator over A, and S the set of all states over A. Then $\{s \in S | s \models regr_o^{nd}(\phi)\} = spreimg_o(\{s \in S | s \models \phi\})$.

Proof.
Let $o = \langle c, (e_1 | \cdots | e_n)\rangle$.

$$\begin{aligned}
&\{s \in S | s \models regr_o^{nd}(\phi)\} \\
&= \{s \in S | s \models regr_{\langle c,e_1\rangle}(\phi) \wedge \cdots \wedge regr_{\langle c,e_n\rangle}(\phi)\} \\
&= \{s \in S | s \models regr_{\langle c,e_1\rangle}(\phi), \ldots, s \models regr_{\langle c,e_n\rangle}(\phi)\} \\
&= \{s \in S | app_{\langle c,e_1\rangle}(s) \models \phi, \ldots, app_{\langle c,e_n\rangle}(s) \models \phi\} \\
&= \{s \in S | s' \models \phi \text{ for all } s' \in img_o(s), \text{ there is } s' \models \phi \text{ with } sos'\} \\
&= spreimg_o(\{s \in S | s \models \phi\})
\end{aligned}$$

3rd = is by properties of deterministic regression.
4th = is by $img_o(s) = \{app_{\langle c,e_1\rangle}(s), \ldots, app_{\langle c,e_n\rangle}(s)\}$. □

Regression for nondeterministic operators
Example

Example
Let $o = \langle d, (b | \neg c)\rangle$. Then

$$\begin{aligned}
regr_o^{nd}(b \leftrightarrow c) &= regr_{\langle d,b\rangle}(b \leftrightarrow c) \wedge regr_{\langle d,\neg c\rangle}(b \leftrightarrow c) \\
&= (d \wedge (\top \leftrightarrow c)) \wedge (d \wedge (b \leftrightarrow \bot)) \\
&\equiv d \wedge c \wedge \neg b.
\end{aligned}$$

Backward distances with formulas

By using regression we can compute formulas that represent backward distance sets.

Definition
Let G be a formula and O a set of operators. The backward distance sets D_i^{bwd} for G, O are represented by the following formulae.

$$\begin{aligned}
D_0^{bwd} &= G \\
D_i^{bwd} &= D_{i-1}^{bwd} \vee \textstyle\bigvee_{o \in O} regr_o^{nd}(D_{i-1}^{bwd}) \text{ for all } i \geq 1
\end{aligned}$$

Backward distances with formulas

Definition
Let G be a formula and O a set of operators, and let $D_0^{bwd}, D_1^{bwd}, \ldots$ be the formulae representing the backward distance sets for G and O. Then the backward distance from a state s to G is

$$\delta_G^{bwd}(s) = \begin{cases} 0 \text{ if } s \models G \\ i \text{ if } s \models D_i^{bwd} \wedge \neg D_{i-1}^{bwd} \end{cases}$$

If $s \not\models D_i^{bwd}$ for all $i \geq 0$ then $\delta_G^{bwd}(s) = \infty$.

General images and preimages with formulas

- The definition of regression covers only a subclass of nondeterministic operators.
- How to define strong preimages for all operators, and images and preimages?
- Now we apply a general idea:
  1. View operators/actions as binary relations.
  2. Represent these binary relations as formulae.
  3. Define relational operations for relations represented as formulae.

General images and preimages with formulas

Definition
Define the set of state variables possibly changed by e as

$$\begin{aligned} \textit{changes}(a) &= \{a\} \\ \textit{changes}(\neg a) &= \{a\} \\ \textit{changes}(c \rhd e) &= \textit{changes}(e) \\ \textit{changes}(e_1 \wedge \cdots \wedge e_n) &= \textit{changes}(e_1) \cup \cdots \cup \textit{changes}(e_n) \\ \textit{changes}(e_1 | \cdots | e_n) &= \textit{changes}(e_1) \cup \cdots \cup \textit{changes}(e_n) \end{aligned}$$

Assumption
Let $e_1 \wedge \cdots \wedge e_n$ occur in the effect of an operator. If $e_1, \ldots, e_n$ are not all deterministic then a and $\neg a$ may occur as an atomic effect in at most one of $e_1, \ldots, e_n$.

This assumption rules out effects like $(a|b) \wedge (\neg a|c)$ that may make a simultaneously true and false.

General images and preimages with formulas

In nondeterministic choices $e_1 | \cdots | e_n$ the formula for each e_i has to express the changes for exactly the same set B of state variables.

Definition

$$\begin{aligned} \tau_B^{nd}(e) &= \tau_B(e) \text{ when } e \text{ is deterministic} \\ \tau_B^{nd}(e_1 | \cdots | e_n) &= \tau_B^{nd}(e_1) \vee \cdots \vee \tau_B^{nd}(e_n) \\ \tau_B^{nd}(e_1 \wedge \cdots \wedge e_n) &= \tau_{B \setminus (B_2 \cup \cdots \cup B_n)}^{nd}(e_1) \wedge \tau_{B_2}^{nd}(e_2) \wedge \cdots \wedge \tau_{B_n}^{nd}(e_n) \\ & \quad \text{where } B_i = \textit{changes}(e_i) \text{ for } i \in \{2, \ldots, n\} \end{aligned}$$

General images and preimages with formulas

Example
We translate the effect

$$e = (a|(d \rhd a)) \wedge (c|d)$$

into a propositional formula. The set of state variables is $A = \{a, b, c, d\}$.

$$\begin{aligned} \tau_{\{a,b,c,d\}}^{nd}(e) &= \tau_{\{a,b\}}^{nd}(a|(d \rhd a)) \wedge \tau_{\{c,d\}}^{nd}(c|d) \\ &= (\tau_{\{a,b\}}^{nd}(a) \vee \tau_{\{a,b\}}^{nd}(d \rhd a)) \wedge (\tau_{\{c,d\}}^{nd}(c) \vee \tau_{\{c,d\}}^{nd}(d)) \\ &= ((a' \wedge (b \leftrightarrow b')) \vee (((a \vee d) \leftrightarrow a') \wedge (b \leftrightarrow b'))) \wedge \\ & \quad ((c' \wedge (d \leftrightarrow d')) \vee ((c \leftrightarrow c') \wedge d')) \end{aligned}$$

General images and preimages with formulas

Definition
Let A be a set of state variables. Let $o = \langle c, e \rangle$ be an operator over A in normal form. Define $\tau_A^{nd}(o) = c \wedge \tau_A^{nd}(e)$.

Lemma
Let o be an operator. Then

$$\begin{aligned} & \{v | v \textit{ is a valuation of } A \cup A', v \models \tau_A^{nd}(o)\} \\ & = \{s \cup s'[A'/A] | s, s' \in S, s' \in \textit{img}_o(s)\}. \end{aligned}$$

Existential and universal abstraction

The most important operations performed on transition relations represented as propositional formulae are based on existential abstraction and universal abstraction.

Definition
Existential abstraction of a formula ϕ with respect to $a \in A$:

$$\exists a.\phi = \phi[\top/a] \vee \phi[\bot/a].$$

Universal abstraction is defined analogously by using conjunction instead of disjunction.

Definition
Universal abstraction of a formula ϕ with respect to $a \in A$:

$$\forall a.\phi = \phi[\top/a] \wedge \phi[\bot/a].$$

∃-abstraction
Examples

Example

$$\begin{aligned} & \exists b.((a \rightarrow b) \wedge (b \rightarrow c)) \\ & = ((a \rightarrow \top) \wedge (\top \rightarrow c)) \vee ((a \rightarrow \bot) \wedge (\bot \rightarrow c)) \\ & \equiv c \vee \neg a \\ & \equiv a \rightarrow c \end{aligned}$$

$$\begin{aligned} & \exists ab.(a \vee b) = \exists b.(\top \vee b) \vee (\bot \vee b) \\ & = ((\top \vee \top) \vee (\bot \vee \top)) \vee ((\top \vee \bot) \vee (\bot \vee \bot)) \\ & = (\top \vee \top) \vee (\top \vee \bot) = \top \end{aligned}$$

Example
∃-abstraction is also known as forgetting:

$$\begin{aligned} & \exists \text{mon} \exists \text{tue}((\text{mon} \vee \text{tue}) \wedge (\text{mon} \rightarrow \text{work}) \wedge (\text{tue} \rightarrow \text{work})) \\ & \equiv \exists \text{tue}((\text{work} \wedge (\text{tue} \rightarrow \text{work})) \vee (\text{tue} \wedge (\text{tue} \rightarrow \text{work}))) \equiv \text{work} \end{aligned}$$

$\forall$ and $\exists$-abstraction in terms of truth-tables

Example

$\forall a$ and $\exists a$ correspond to combining pairs of lines with the same valuation for variables other than a.

Example

$$\exists c.(a \vee (b \wedge c)) \equiv a \vee b \quad \forall c.(a \vee (b \wedge c)) \equiv a$$

a	b	c	$a \vee (b \wedge c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

a	b	$\exists c.(a \vee (b \wedge c))$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	$\forall c.(a \vee (b \wedge c))$
0	0	0
0	1	0
1	0	1
1	1	1

Properties of abstraction operations

Definition

Existential and universal abstraction of ϕ with respect to a set of atomic propositions $B = \{b_1, \ldots, b_n\}$ are

$$\begin{aligned} \exists B.\phi &= \exists b_1.(\exists b_2.(\ldots \exists b_n.\phi \ldots)) \\ \forall B.\phi &= \forall b_1.(\forall b_2.(\ldots \forall b_n.\phi \ldots)). \end{aligned}$$

Properties of abstracted formulas

1. Let ϕ be a formula over A. Then $\exists A.\phi$ and $\forall A.\phi$ are formulae that consist of the constants $\top$ and $\bot$ and the logical connectives only.
2. The truth-values of these formulae are independent of the valuation of A, that is, their values are the same for all valuations.
3. $\exists A.\phi \equiv \top$ if and only if ϕ is satisfiable.
4. $\forall A.\phi \equiv \top$ if and only if ϕ is valid.

Properties of $\forall$ and $\exists$ abstraction

Lemma

If ϕ is a formula over $A \cup A'$ and v a valuation of A then

1. $v \models \exists A'.\phi$ *iff* $v \cup v' \models \phi$ *for some valuation* v' *of* A'.
2. $v \models \forall A'.\phi$ *iff* $v \cup v' \models \phi$ *for all valuations* v' *of* A'.

Size of abstracted formulae

- Abstracting one variable takes polynomial time in the size of the formula.
- Abstracting one variable may double the formula size.
- Abstracting n variables may increase size by factor 2^n.
- For making abstraction practical the formulae must be simplified, for example with equivalences like $\top \wedge \phi \equiv \phi$, $\bot \wedge \phi \equiv \bot$, $\top \vee \phi \equiv \top$, $\bot \vee \phi \equiv \phi$, $\neg\bot \equiv \top$, and $\neg\top \equiv \bot$.

Images by $\exists$-abstraction

Let

- $A = \{a_1, \ldots, a_n\}$,
- $A' = \{a'_1, \ldots, a'_n\}$,
- ϕ_1 be a formula on A representing a row vector $V_{1 \times 2^n}$ (equivalently, a set of valuations of A), and
- ϕ_2 a formula on $A \cup A'$ representing a matrix $M_{2^n \times 2^n}$ (equivalently, a binary relation on valuations of A).

The product matrix VM of size 1×2^n is represented by

$$\exists A.(\phi_1 \wedge \phi_2)$$

which is a formula on A'.
To obtain a formula over A we have to rename the variables.

$$(\exists A.(\phi_1 \wedge \phi_2))[A/A']$$

Images by $\exists$-abstraction

Example

Let $A = \{a, b\}$ be the state variables.

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

represents the image of $\{00, 10\}$ with respect to a relation.

$$\begin{aligned} &\exists a.\exists b.(\neg b \wedge (b \leftrightarrow \neg b')) \\ &\equiv \exists b.(\neg b \wedge (b \leftrightarrow \neg b')) \\ &\equiv (\neg\top \wedge (\top \leftrightarrow \neg b')) \vee (\neg\bot \wedge (\bot \leftrightarrow \neg b')) \\ &\equiv b' \end{aligned}$$

The formula b represents $\{01, 11\}$.

Matrix multiplication by $\exists$-abstraction

Let

- $A = \{a_1, \ldots, a_n\}$,
- $A' = \{a'_1, \ldots, a'_n\}$,
- $A'' = \{a''_1, \ldots, a''_n\}$,
- ϕ_1 be a formula on $A \cup A'$ representing matrix M_1 and
- ϕ_2 a formula on $A' \cup A''$ representing matrix M_2.

The matrices M_1 and M_2 have size $2^n \times 2^n$.
The product matrix $M_1 M_2$ is represented by

$$\exists A'.(\phi_1 \wedge \phi_2)$$

which is a formula on $A \cup A''$.

Matrix multiplication by ∃-abstraction

Example

Example

Let $\phi_1 = a \leftrightarrow \neg a'$ and $\phi_2 = a' \leftrightarrow a''$ represent two actions, reversing the truth-value of a and doing nothing. The sequential composition of these actions is

$$\begin{aligned} \exists a'.\phi_1 \wedge \phi_2 &= ((a \leftrightarrow \neg\top) \wedge (\top \leftrightarrow a'')) \vee ((a \leftrightarrow \neg\bot) \wedge (\bot \leftrightarrow a'')) \\ &\equiv ((a \leftrightarrow \bot) \wedge (\top \leftrightarrow a'')) \vee ((a \leftrightarrow \top) \wedge (\bot \leftrightarrow a'')) \\ &\equiv (\neg a \wedge a'') \vee (a \wedge \neg a'') \\ &\equiv a \leftrightarrow \neg a''. \end{aligned}$$

Matrix multiplication

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

$$\begin{aligned} &\exists a'.\exists b'.(\neg a \leftrightarrow a') \wedge (\neg b \leftrightarrow b') \wedge (a' \leftrightarrow b'') \wedge (b' \leftrightarrow a'') \\ &\equiv (\neg a \leftrightarrow b'') \wedge (\neg b \leftrightarrow a''). \end{aligned}$$

Images and preimages by formula manipulation

Define $s[A'/A] = \{\langle a', s(a)\rangle | a \in A\}$.

Lemma

Let ϕ be a formula on A and v a valuation of A. Then $v \models \phi$ iff $v[A'/A] \models \phi[A'/A]$.

Definition

Let o be an operator and ϕ a formula. Define

$$\begin{aligned} \mathit{img}_o(\phi) &= (\exists A.(\phi \wedge \tau_A^{nd}(o)))[A/A'] \\ \mathit{preimg}_o(\phi) &= \exists A'.(\tau_A^{nd}(o) \wedge \phi[A'/A]) \\ \mathit{spreimg}_o(\phi) &= \forall A'.(\tau_A^{nd}(o) \rightarrow \phi[A'/A]) \wedge \exists A'.\tau_A^{nd}(o). \end{aligned}$$

Images by formula manipulation

Theorem

Let $T = \{s \in S | s \models \phi\}$. Then $\{s \in S | s \models \mathit{img}_o(\phi)\} = \{s \in S | s \models (\exists A.(\phi \wedge \tau_A^{nd}(o)))[A/A']\} = \mathit{img}_o(T)$.

Proof.

$s' \models (\exists A.(\phi \wedge \tau_A^{nd}(o)))[A/A']$ □
iff $s'[A'/A] \models \exists A.(\phi \wedge \tau_A^{nd}(o))$
iff there is valuation s of A s.t. $(s \cup s'[A'/A]) \models \phi \wedge \tau_A^{nd}(o)$
iff there is valuation s of A s.t. $s \models \phi$ and $(s \cup s'[A'/A]) \models \tau_A^{nd}(o)$
iff there is $s \in T$ s.t. $(s \cup s'[A'/A]) \models \tau_A^{nd}(o)$
iff there is $s \in T$ s.t. $s' \in \mathit{img}_o(s)$
iff $s' \in \mathit{img}_o(T)$.

Preimages by formula manipulation

Theorem

Let $T = \{s \in S | s \models \phi\}$. Then $\{s \in S | s \models \mathit{preimg}_o(\phi)\} = \{s \in S | s \models \exists A'.(\tau_A^{nd}(o) \wedge \phi[A'/A])\} = \mathit{preimg}_o(T)$.

Proof.

$s \models \exists A'.(\tau_A^{nd}(o) \wedge \phi[A'/A])$
iff there is $s'_0 : A' \rightarrow \{0,1\}$ s.t. $(s \cup s'_0) \models \tau_A^{nd}(o) \wedge \phi[A'/A]$
iff there is $s'_0 : A' \rightarrow \{0,1\}$ s.t. $s'_0 \models \phi[A'/A]$ and $(s \cup s'_0) \models \tau_A^{nd}(o)$
iff there is $s' : A \rightarrow \{0,1\}$ s.t. $s' \models \phi$ and $(s \cup s'_0) \models \tau_A^{nd}(o)$
iff there is $s' \in T$ s.t. $(s \cup s'[A'/A]) \models \tau_A^{nd}(o)$
iff there is $s' \in T$ s.t. $s' \in \mathit{img}_o(s)$
iff there is $s' \in T$ s.t. $s \in \mathit{preimg}_o(s')$
iff $s \in \mathit{preimg}_o(T)$.
Above we define $s' = s'_0[A/A']$ (and hence $s'_0 = s'[A'/A]$.) □

Strong preimages by formula manipulation

Theorem

Let $T = \{s \in S | s \models \phi\}$. Then $\{s \in S | s \models \mathit{spreimg}_o(\phi)\} = \{s \in S | s \models \forall A'.(\tau_A^{nd}(o) \rightarrow \phi[A'/A]) \wedge \exists A'.\tau_A^{nd}(o)\} = \mathit{spreimg}_o(T)$.

Proof.

See the lecture notes. □

Strong preimages vs. regression

Corollary

Let $o = \langle c, (e_1 | \cdots | e_n)\rangle$ be an operator such that all e_i are deterministic. The formula $\mathit{spreimg}_o(\phi)$ is logically equivalent to $\mathit{regr}_o^{nd}(\phi)$.

Proof.

$\{s \in S | s \models \mathit{regr}_o(\phi)\} = \mathit{spreimg}_o(\{s \in S | s \models \phi\}) = \{s \in S | s \models \mathit{spreimg}_o(\phi)\}$. □

Summary of matrix/logic/relational operations

matrices	formulas	state sets
vector $V_{1\times n}$	formula on A	set
matrix $M_{n\times n}$	formula on $A \cup A'$	relation
$V_{1\times n} + V'_{1\times n}$	$\phi_1 \vee \phi_2$	union
	$\phi_1 \wedge \phi_2$	intersection
$V_{1\times n} \times M_{n\times n}$	$(\exists A.(\phi \wedge \tau_A^{nd}(o)))[A/A']$	$\mathit{img}_o(T)$
$M_{n\times n} \times V_{n\times 1}$	$\exists A'.(\tau_A^{nd}(o) \wedge \phi[A'/A])$	$\mathit{preimg}_o(T)$
	$\forall A'.(\tau_A^{nd}(o) \rightarrow \phi[A'/A]) \wedge \exists A'.\tau_A^{nd}(o)$	$\mathit{spreimg}_o(T)$

Images and preimages of sets of operators

The union of images of ϕ with respect to all operators $o \in O$ is

$$\bigvee_{o \in O} img_o(\phi).$$

This can be computed more directly by using the disjunction $\bigvee_{o \in O} \tau_A(o)$ of the transition formulae:

$$\exists A.(\phi \wedge (\bigvee_{o \in O} \tau_A(o)))[A/A'].$$

Same works for preimages.

Image computation vs. planning by satisfiability

- We tested plan existence by testing satisfiability of

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

  where $\mathcal{R}_1(A, A') = \bigvee_{o \in O} \tau_A(o)$.
- $\exists$-abstracting $A^0 \cup \cdots \cup A^{t-1}$ yields

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

- This is equivalent to conjoining the t-fold image of ι

$$\bigvee_{o \in O} img_o(\cdots \bigvee_{o \in O} img_o(\iota) \cdots)$$

  with G to test goal reachability in t steps.
  We can do the same with preimages starting from G.

Shannon expansion

Definition
3-place connective if-then-else is defined by

$$\text{ite}(a, \phi_1, \phi_2) = (a \wedge \phi_1) \vee (\neg a \wedge \phi_2)$$

where a is a proposition.

Definition
Shannon expansion of a formula ϕ with respect to $a \in A$ is

$$\phi \equiv (a \wedge \phi[\top/a]) \vee (\neg a \wedge \phi[\bot/a]) = \text{ite}(a, \phi[\top/a], \phi[\bot/a])$$

Binary decision diagrams
Example

By repeated application of Shannon expansion any propositional formula can be transformed to an equivalent formula containing no other connectives than *ite* and propositional variables only in the first position of *ite*.

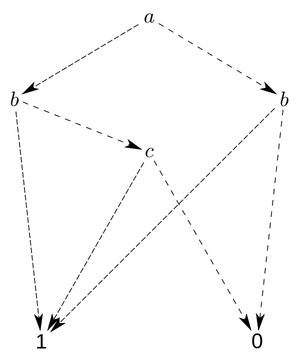
Example
$$\begin{aligned}
& (a \vee b) \wedge (b \vee c) \\
\equiv\ & ite(a, (\top \vee b) \wedge (b \vee c), (\bot \vee b) \wedge (b \vee c)) \\
\equiv\ & ite(a, b \vee c, b) \\
\equiv\ & ite(a, ite(b, \top \vee c, \bot \vee c), ite(b, \top, \bot)) \\
\equiv\ & ite(a, ite(b, \top, c), ite(b, \top, \bot)) \\
\equiv\ & ite(a, ite(b, \top, ite(c, \top, \bot)), ite(b, \top, \bot))
\end{aligned}$$

Binary decision diagrams
Canonicity

Transformation to ordered BDDs
1. Fix an ordering $a_1, \ldots, a_n$ on all propositional variables.
2. Apply Shannon expansion to all variables in this order.
3. Represent the resulting formulae as directed acyclic graphs (DAG) so that shared subformulae occur only once.

Theorem
Let ϕ_1 and ϕ_2 be two ordered BDDs obtained by using the same variable ordering. Then $\phi_1 \equiv \phi_2$ if and only if ϕ_1 and ϕ_2 are isomorphic (the same DAG.)

Binary decision diagrams: example



Satisfiability algorithms vs. BDDs

Comparison: formula size, runtime

technique	size of $\mathcal{R}_1(P, P')$	runtime for plan length n
satisfiability	not a problem	exponential in n
BDDs	major problem	less dependent on n

Comparison: resource consumption

technique	critical resource
satisfiability	runtime
BDDs	memory

Comparison: application domain

technique	types of problems
satisfiability	lots of state variables, short plans
BDDs	few state variables, long plans

Properties of CPC normal forms

Trade-offs between different CPC normal forms
Normal forms that allow faster reasoning are more expensive to construct from an arbitrary propositional formula and may be much bigger.

Properties of different normal forms

	$\vee$	$\wedge$	$\neg$	$\phi \in$TAUT?	$\phi \in$SAT?	$\phi \equiv \phi'$?
circuits	poly	poly	poly	co-NP-hard	NP-hard	co-NP-hard
formulae	poly	poly	poly	co-NP-hard	NP-hard	co-NP-hard
DNF	poly	exp	exp	co-NP-hard	in P	co-NP-hard
CNF	exp	poly	exp	in P	NP-hard	co-NP-hard
BDD	exp	exp	poly	in P	in P	in P

For BDDs one $\vee/\wedge$ is polynomial time/size (size is doubled) but repeated $\vee/\wedge$ lead to exponential size.