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 is a primitive form of memory.
   - The operator to be executed depends on the state of the execution.

Images

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

\[
\text{img}_\alpha(T) = \{ s' \in S | s' = s\alpha \}
\]

Preimages

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

\[
\text{preimg}_\alpha(T) = \{ s \in S | \exists s' \in T: s' = s\alpha \}
\]
**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 $o$.

- $\text{spreimg}_o(T) = \{ s \mid \exists i \geq 0 \, s \in \text{spreimg}_o(D_i) \}$

**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

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** If 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.

**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 $s' \in \text{img}_o(s)$, $s'$ 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**

**Definition**

Let $G$ be a set of states and $O$ a set of operators. Define the backward distance sets $D_i^{\text{bad}}$ 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.

- $D_0^{\text{bad}} = G$
- $D_i^{\text{bad}} = D_{i-1}^{\text{bad}} \cup \text{spreimg}_o(D_{i-1}^{\text{bad}})$ for all $i \geq 1$

**Definition (Strong preimage of a set of states)**

$\text{spreimg}_o(T) = \{ s \mid \exists i \geq 0 \, s \in D_i^{\text{bad}}, s \in \text{img}_o(s) \in T \}$

**Backward distances**

**Example**

- Distance to $G$
  - $\infty$
  - 3
  - 2
  - 1
  - 0

**Backward distances**

**Definition**

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

- $\delta_i^G(s) = 0$ if $s \in G$
- $\delta_i^G(s) = D_i^{\text{bad}}(s)$ if $s \in D_i^{\text{bad}} \setminus D_{i-1}^{\text{bad}}$

If $s \notin D_i^{\text{bad}}$ for all $i \geq 0$ then $\delta_i^G(s) = \infty$. 

---

(Albert-Ludwigs-Universität Freiburg) AI Planning  May 30, 2005  9 / 56

(Albert-Ludwigs-Universität Freiburg) AI Planning  May 30, 2005  10 / 56

(Albert-Ludwigs-Universität Freiburg) AI Planning  May 30, 2005  11 / 56

(Albert-Ludwigs-Universität Freiburg) AI Planning  May 30, 2005  12 / 56

(Albert-Ludwigs-Universität Freiburg) AI Planning  May 30, 2005  13 / 56

(Albert-Ludwigs-Universität Freiburg) AI Planning  May 30, 2005  14 / 56

(Albert-Ludwigs-Universität Freiburg) AI Planning  May 30, 2005  15 / 56

(Albert-Ludwigs-Universität Freiburg) AI Planning  May 30, 2005  16 / 56
Construction of a plan based on distances

Extraction of a plan from distance sets
1. Let \( S' \subset S \) be those states having a finite backward distance.
2. Let \( s \) be a state with distance \( i = d_{\text{Bwd}}(s) \geq 1 \).
3. Assign to \( \pi(s) \) any operator \( o \in O \) such that \( \text{img}_o(s) \subset D_{\text{Bwd}}^i \).

Then \( \pi \) solves the planning problem for \( (S, I, O, G, P) \) if \( I \subseteq S' \).

Making the algorithm a logic-based algorithm

- An algorithm that represents the states explicitly is feasible for transition systems with most \( 10^8 \) to \( 10^9 \) states.
- For planning with bigger transition systems structural properties of the transition system have to be taken advantage of.
- Representing state sets as propositional formulas 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.

Regression

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 \( \phi \) be a propositional formula and \( o = (c, e_1, \ldots, e_n) \) an operator where \( e_1, \ldots, e_n \) are deterministic. Define

\[
\text{regr}_o^d(\phi) = \text{regr}_{(c,e_1)}(\phi) \land \cdots \land \text{regr}_{(c,e_n)}(\phi).
\]

Regression for nondeterministic operators

Example

Let \( o = (d, (b \neg c)) \). Then

\[
\text{regr}_o^d(b \rightarrow c) = \text{regr}_{d,b}(b \rightarrow c) \land \text{regr}_{d,b \rightarrow c}(b \rightarrow c) = (d \land (b \rightarrow (\neg c) \land (d \land (b \rightarrow \bot)))
\]

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 \( D_{\text{Bwd}}^i \) for \( G, O \) are represented by the following formulae.

\[
D_{\text{Bwd}}^0 = G
\]

\[
D_{\text{Bwd}}^i = D_{\text{Bwd}}^{i-1} \lor \bigvee_{o \in O} \text{regr}_o^d(D_{\text{Bwd}}^{i-1}) \text{ for } i \geq 1.
\]
General images and preimages with formulas

**Definition**
Let \( \Delta \) be a set of state variables, and let \( D_{\Delta}^b \) be the backward distance set for \( \Delta \). Then the backward distance from state \( s \) to \( G \) is
\[
\delta_{D_{\Delta}^b}^b(s) = \begin{cases} 
0 & \text{if } s \models G \\
\infty & \text{if } s \models \neg D_{\Delta}^b \land \neg D_{\Delta}^{b-1} 
\end{cases}
\]
If \( s \models D_{\Delta}^b \) for all \( i \geq 0 \) then \( \delta_{D_{\Delta}^b}^b(s) = \infty \).

Existential and universal abstraction

**Definition**
Existential abstraction of a formula \( \phi \) with respect to \( a \in A' \):
\[
\exists a, \phi = \phi[T/a] \lor \phi[\bot/a].
\]

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

**Example**
We translate the effect
\[
e = (a(d \Rightarrow a)) \land (c(d))
\]
into a propositional formula. The set of state variables is
\[
A = \{a, b, c, d\}.
\]

- 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 formulas.
  3. Define relational operations for relations represented as formulas.

Example
We translate the effect
\[
e = (a(d \Rightarrow a)) \land (c(d))
\]
into a propositional formula. The set of state variables is
\[
A = \{a, b, c, d\}.
\]

**Definition**
Let \( \Delta \) be a set of state variables. Let \( o = (c, e) \) be an operator over \( \Delta \) in normal form. Define \( \tau_{\Delta}^o_{\Delta}(o) = c \land \tau_{\Delta}^o_{\Delta}(e) \).

**Lemma**
Let \( o \) be an operator. Then
\[
\{ e \mid e \text{ is a valuation of } A \cup A', e \vdash \tau_{\Delta}^o_{\Delta}(o) \} = \{ s \cup s'[A/\Delta], s, s' \in \text{img}(o) \}.
\]

**Examples**

- **\( \exists \)-abstraction**
  - **Example**
    \[
    \exists b, c = (a \land b) \land (a \land c) = (a \land b) \land (a \land c).
    \]
    - \( a \lor \neg a \Rightarrow a \Rightarrow d \lor e \Rightarrow e \Rightarrow c \lor c \Rightarrow (a \lor b) \lor (a \lor b) \Rightarrow (a \lor b) \lor (a \lor b) \Rightarrow (a \lor b) \lor (a \lor b) \Rightarrow T
    \]
    - \( \exists b, (a \lor b) = \exists b, (a \lor b) \lor (a \lor b) \Rightarrow (a \lor b) \lor (a \lor b) \Rightarrow (a \lor b) \lor (a \lor b) \Rightarrow (a \lor b) \lor (a \lor b) \Rightarrow T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
  - **Example**
    \[
    \exists a \lor b = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = \exists a \lor b \lor (a \lor b) = T
    \]
∀ and ∃-abstraction in terms of truth-tables

Example

∀φ and ∃φ correspond to combining pairs of lines with the same valuation for variables other than φ.

Example

∀c : (a ∨ (b ∧ c)) ≡ a ∨ b, ∀c : (a ∨ (b ∧ c)) ≡ a

Properties of abstraction operations

Definition

Existential and universal abstraction of φ with respect to a set of atomic propositions B = {b₁, ..., bₙ} are

∃B φ = ∃b₁.(∃b₂.(...∃bₙ φ...)), ∀B φ = ∀b₁.(∀b₂.(...∀bₙ φ...)).

Properties of ∀ and ∃ abstraction

1. Let φ be a formula over A. Then ∃A φ and ∀A φ are formulae that consist of the constants T and ⊥ 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. ∃A φ ≡ T if and only if φ is satisfiable.
4. ∀A φ ≡ T if and only if φ is valid.

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 T ∧ φ ≡ φ, ⊥ ∨ φ ≡ ⊥, T ∨ φ ≡ T, ⊥ ∨ φ ≡ φ, ¬⊥ ≡ T, and ¬T ≡ ⊥.

Examples by ∃-abstraction

Let
- A = {a₁, ..., aₙ},
- A' = {a₁', ..., aₙ'},
- φ be a formula on A representing a row vector V₁×2ⁿ (equivalently, a set of valuations of A), and
- φ' a formula on A ∪ A' representing a matrix M₂ⁿ×2ⁿ (equivalently, a binary relation on valuations of A).

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

∃A φ₁ ∧ φ₂

which is a formula on A'.

To obtain a formula over A we have to rename the variables.

∃A (φ₁ ∧ φ₂)[A/A']

Images by ∃-abstraction

Example

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

\[
\begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1
\end{pmatrix}
\times
\begin{pmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1
\end{pmatrix}
= \begin{pmatrix}
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{pmatrix}
\]

represents the image of (00)₁₀ with respect to a relation.

∃ₐ₀∃ₐ₁(¬b ∧ (b ↔ ¬b'))

≡ ∃ₐ₀(¬b ∧ (b ↔ ¬b'))

≡ (¬¬b ∧ (T ↔ ¬b')) ∨ (¬⊥ ∧ (⊥ ↔ ¬b'))

≡ b'

The formula b represents {1, 0, 11}.
Matrix multiplication by $\exists$-abstraction

Example

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

$3\alpha'. \phi_1 \land \phi_2$

$= ((\alpha \rightarrow T) \land (T \rightarrow \alpha')) \lor ((\alpha \rightarrow \bot) \land (\bot \rightarrow \alpha'))$

$= \neg (\alpha \land \alpha') \lor (\alpha \land \neg \alpha')$

$= \alpha \rightarrow \neg \alpha''.$

Images and preimages by formula manipulation

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

Lemma

Let $\phi$ 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 $\alpha$ be an operator and $\phi$ a formula. Define

$\text{img}(\phi) = (\exists A. (\phi \land r_{\alpha}^0)(A/A'))$

$\text{preimg}(\phi) = \exists A'. (r_{\alpha}^0(A') \land o[A'/A])$

$\text{spreimg}(\phi) = \forall A'. (r_{\alpha}^0(A') \rightarrow o[A'/A]) \land \exists A'. r_{\alpha}^0(A')$.

Images by formula manipulation

Theorem

Let $T = \{ s \in S | s \models \phi \}$. Then $\{ s \in S | s \models \text{img}(\phi) \} = \{ s \in S | s \models (\exists A. (\phi \land r_{\alpha}^0)(A/A')) \} = \text{preimg}(T)$.

Proof.

$s \models \exists A'. (r_{\alpha}^0(A') \land o[A'/A])$

iff there is $s' : A' \rightarrow \{ 0, 1 \}$ s.t. $s \cup s' \models r_{\alpha}^0(A') \land o[A'/A]$

iff there is $s' : A' \rightarrow \{ 0, 1 \}$ s.t. $s' \models r_{\alpha}^0(A')$ and $(s \cup s') \models r_{\alpha}^0(A')$

iff there is $s' : A' \rightarrow \{ 0, 1 \}$ s.t. $s' \models r_{\alpha}^0(A')$ and $(s \cup s') \models r_{\alpha}^0(A')$

iff there is $s' : T$ s.t. $(s \cup s')(A' \rightarrow A) \models r_{\alpha}^0(A')$

iff there is $s' \models \text{preimg}(s)$

iff there is $s' \models \text{preimg}(T)$.

Above we define $s' = s'(A/A')$ (and hence $s' = s'(A'/A)$).

Preimages by formula manipulation

Theorem

Let $T = \{ s \in S | s \models \phi \}$. Then $\{ s \in S | s \models \text{spreimg}(\phi) \} = \{ s \in S | s \models \forall A'. (r_{\alpha}^0(A') \rightarrow o[A'/A]) \land \exists A'. r_{\alpha}^0(A') \} = \text{spreimg}(T)$.

Proof.

See the lecture notes.

Strong preimages vs. regression

Corollary

Let $\circ = (\circ_1 \circ_2 \cdots \circ_n)$ be an operator such that all $\circ_i$ are deterministic. The formula $\text{spreimg}(\circ)$ is logically equivalent to $\text{regre}_{\circ}$.

Proof.

$\{ s \in S | s \models \text{regre}(\circ) \} = \text{spreimg}(\{ s \in S | s \models \phi \}) = \{ s \in S | s \models \text{spreimg}(\phi) \}.$

Summary of matrix/logic/relation operations

<table>
<thead>
<tr>
<th>matrices</th>
<th>formulas</th>
<th>state sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>vector $V_{1 \times n}$</td>
<td>formula on $A$</td>
<td>union $\text{img}(T)$</td>
</tr>
<tr>
<td>matrix $M_{n \times m}$</td>
<td>formula on $A \cup A'$</td>
<td>intersection $\text{preimg}(T)$</td>
</tr>
<tr>
<td>$V_{1 \times n} \cup V_{1 \times m}$</td>
<td>$A \cup A'$</td>
<td>$\text{preimg}(T)$</td>
</tr>
<tr>
<td>$V_{1 \times n} \lor V_{1 \times m}$</td>
<td>$A \lor A'$</td>
<td>$\text{preimg}(T)$</td>
</tr>
<tr>
<td>$V_{1 \times n} \times M_{n \times m}$</td>
<td>$A \land A'$</td>
<td>$\text{preimg}(T)$</td>
</tr>
<tr>
<td>$V_{1 \times n} \land M_{n \times m}$</td>
<td>$A \land A'$</td>
<td>$\text{preimg}(T)$</td>
</tr>
</tbody>
</table>
Images and preimages of sets of operators

The union of images of \( \phi \) with respect to all operators \( \sigma \in \mathcal{O} \) is

\[
\bigcup_{\sigma \in \mathcal{O}} \text{img}_\sigma(\phi).
\]

This can be computed more directly by using the disjunction \( \bigvee_{\sigma \in \mathcal{O}} \tau_\sigma(\phi) \) of the transition formulae:

\[
\exists A \left( \phi \wedge \left( \bigvee_{\sigma \in \mathcal{O}} \tau_\sigma(\phi) \right) \right)[A/\mathcal{A}].
\]

Same works for preimages.

Binary decision diagrams BDDs

Shannon expansion

Definition

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

\[
\text{ite}(a, \phi_1, \phi_2) = (a \wedge \phi_1) \lor \neg(a \wedge \phi_2)
\]

where \( a \) is a proposition.

Definition

Shannon expansion of a formula \( \phi \) with respect to \( a \in A \) is

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

Binary decision diagrams BDDs

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

Satisfiability algorithms vs. BDDs

Comparison: formula size, runtime

<table>
<thead>
<tr>
<th>Technique</th>
<th>Comparison for different size of ( R\left( A^{0 \ldots n} \right) )</th>
<th>Runtime for plan length ( n ) for ( \mathcal{O} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfiability</td>
<td>not a problem</td>
<td>exponential in ( n )</td>
</tr>
<tr>
<td>BDDs</td>
<td>major problem</td>
<td>less dependent on ( n )</td>
</tr>
</tbody>
</table>

Comparison: resource consumption

<table>
<thead>
<tr>
<th>Technique</th>
<th>Comparison of resources for different types of problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfiability</td>
<td>runtime</td>
</tr>
<tr>
<td>BDDs</td>
<td>memory</td>
</tr>
</tbody>
</table>

Comparison: application domain

<table>
<thead>
<tr>
<th>Technique</th>
<th>Comparison of applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfiability</td>
<td>lots of state variables, short plans</td>
</tr>
<tr>
<td>BDDs</td>
<td>few state variables, long plans</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Normal Form</th>
<th>( \vee \wedge \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>poly</td>
<td>poly</td>
<td>poly</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>Formulae</td>
<td>poly</td>
<td>poly</td>
<td>poly</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>DNF</td>
<td>poly</td>
<td>exp</td>
<td>poly</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>CNF</td>
<td>exp</td>
<td>poly</td>
<td>exp</td>
<td>in P</td>
</tr>
<tr>
<td>BDD</td>
<td>exp</td>
<td>poly</td>
<td>exp</td>
<td>in P</td>
</tr>
</tbody>
</table>

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