COL:750/7250

Foundations of Automatic Verification

Instructor: Priyanka Golia

Course Webpage

Bounded vs Unbounded Model Checking

Bounded:

We unroll the transition system up to a fixed depth k.

Is there a counterexample of length ≤ k?

If no counterexample is found, we increase k and repeat.

It only checks for violations up to length k

Output— counterexample

Tools: CBMC, NuSMV

Unbounded:

We still unroll transitions, but with a different purpose: to construct an inductive invariant (proof that holds for all k).

We generalize beyond specific length and reason about all reachable states.

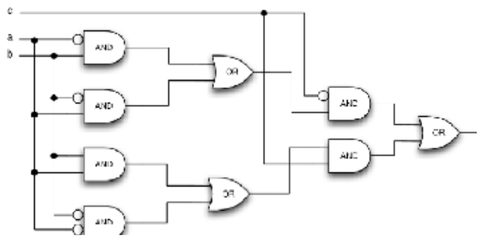
The property is proven inductively.

Output— proof/ counterexample

Tools: NuSMV, IC3, PDR

Formal Verification

```
PC1 (char[] SP, char[] UI) {
        for (int i=0; i<UI.length(); i++) {
                if(SP[i] != UI[i]) return No;
        }
        return Yes;
}
```



System

Satisfies

Properties

$S(I,O) \models P(I,O)$

Is the always the case that S satisfies Property P?

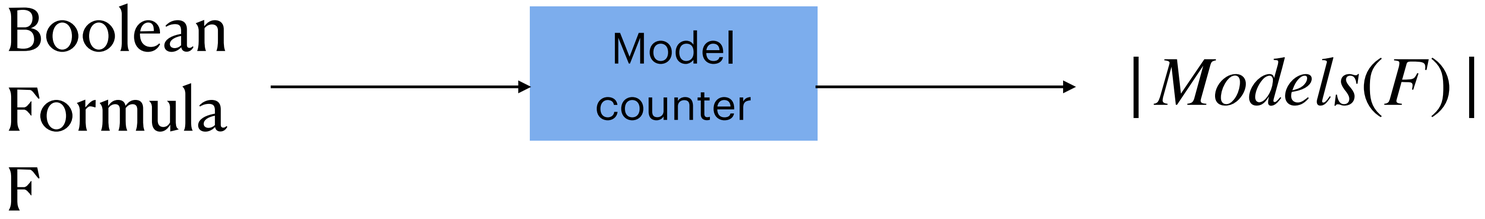
How often S satisfies P?

Why S doesn't satisfy P?

How often System satisfies Property?

Model Counting!

Finding out how many solutions are there for a given set of constraints.



How often System satisfies Property?

Model Counting!

Finding out how many solutions are there for a given set of constraints.

$$
\begin{aligned}
&ModelCounter(F, count)\{\\
&\quad Result, \sigma = CheckSAT(F)\\
&\quad if\ (Result == SAT)\{\\
&\qquad count++\ \}\\
&\quad else\ \ Return\ count\\
&\quad ModelCounter(F \wedge \neg\sigma, count)\}
\end{aligned}
$$

Assuming access to a NP oracle !

How often System satisfies Property?

Model Counting!

Finding out how many solutions are there for a given set of constraints.

$ModelCounter(F)\{$

$If\ F\ is\ 0\ then\ Return\ 0$

$If\ F\ is\ 1\ then\ Return\ 1$

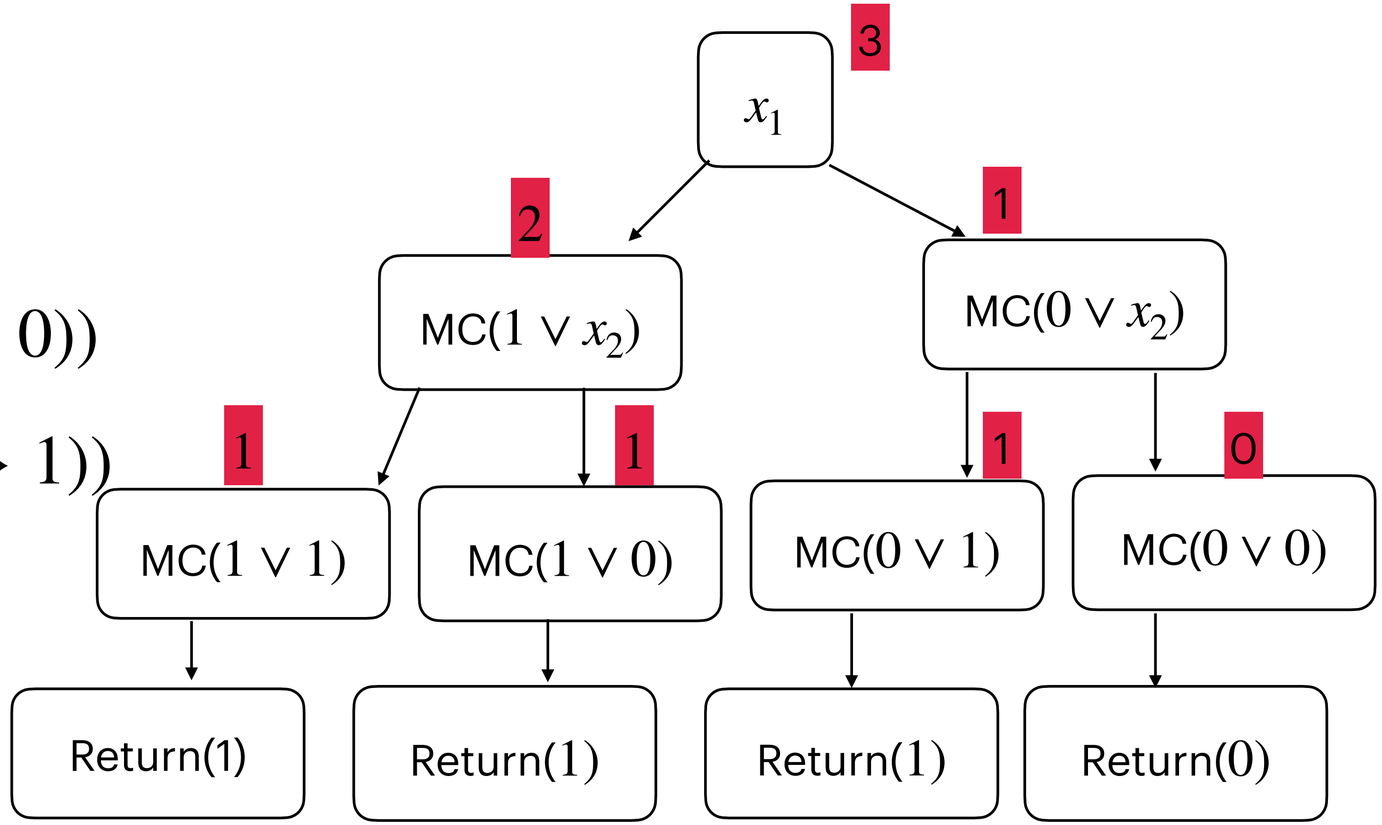
$pick\ \ x \leftarrow VARs(F)$

$C_o = ModelCounter(F(x \mapsto 0))$

$C_1 = ModelCounter(F(x \mapsto 1))$

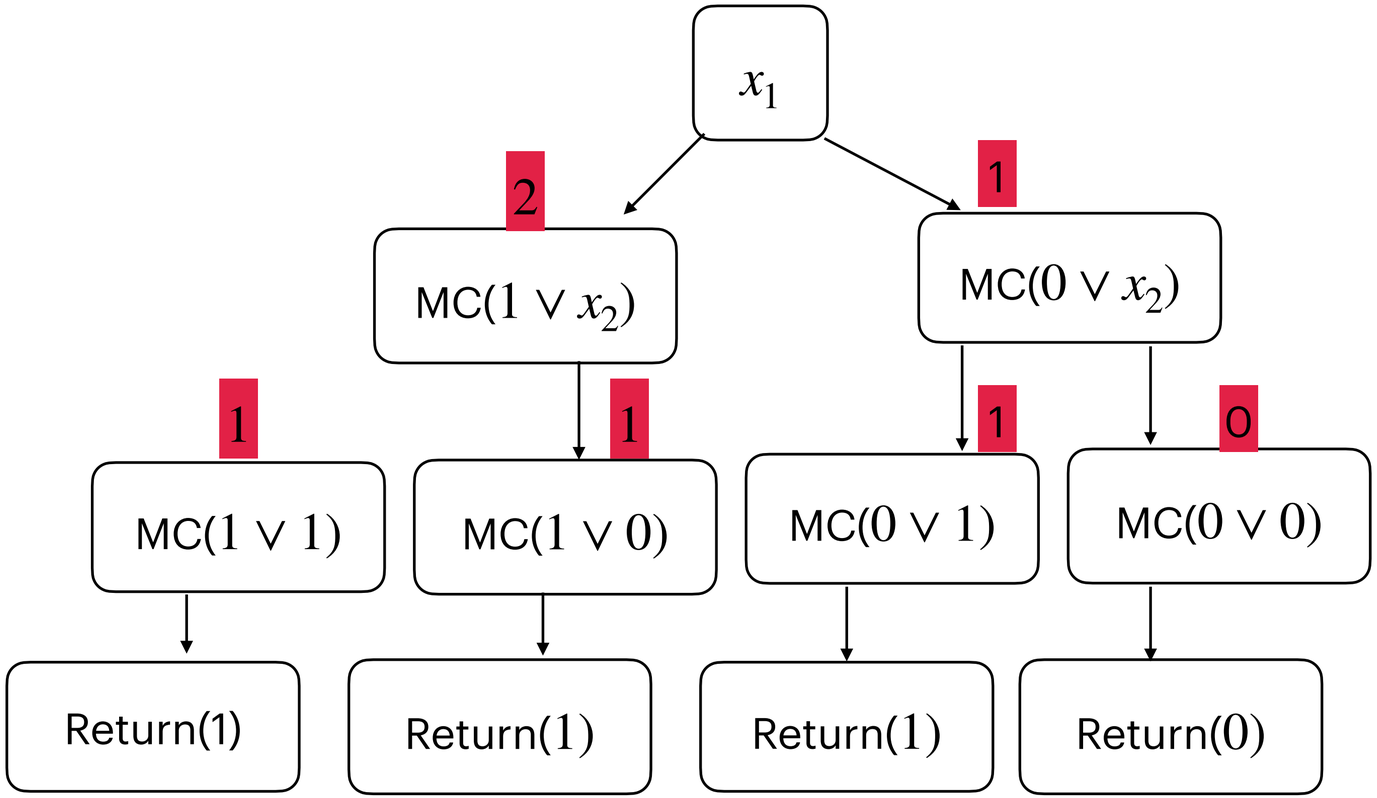
$return\ \ C_o + C_1\ \}$

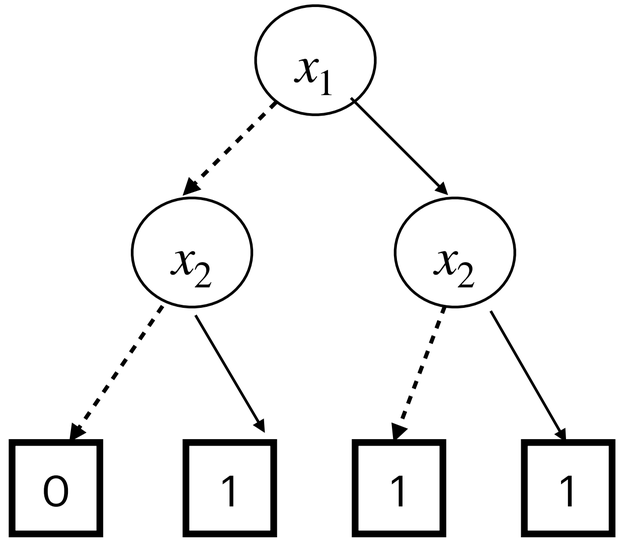
$F = x_1 \vee x_2$



How often System satisfies Property?

$$F = x_1 \vee x_2$$



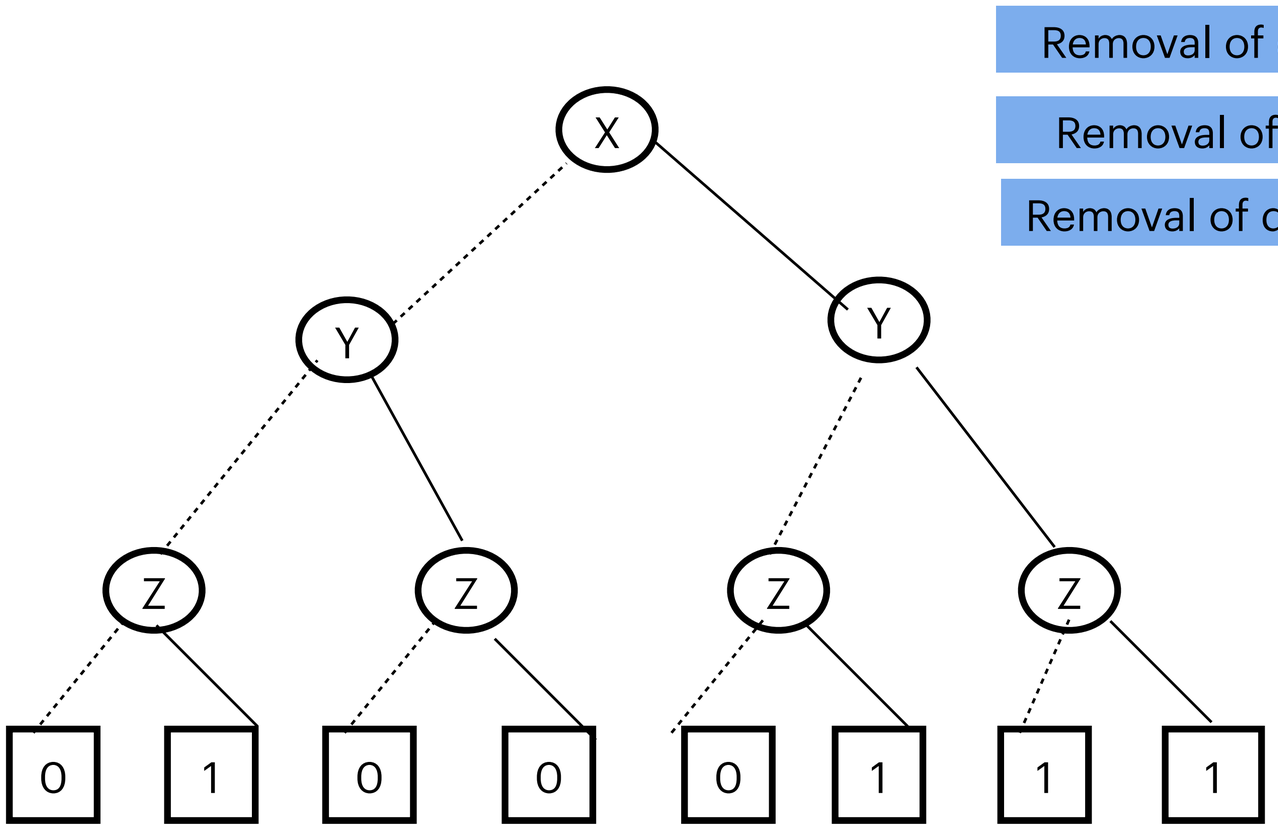


In OBDD, Model count is Sum of leaf nodes.

Model Counting

ROBDD — Reduced Ordered Binary Decision Diagrams

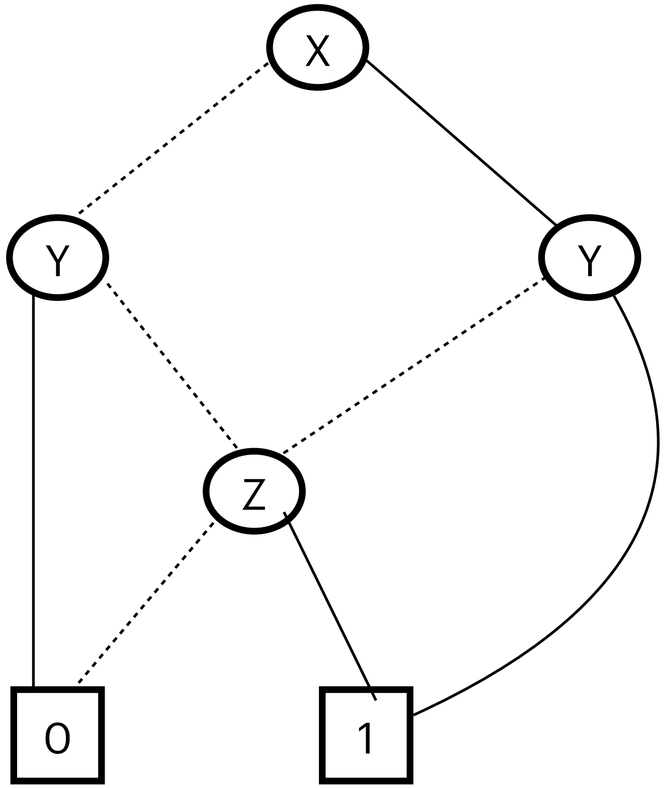
$F = (x \wedge y) \vee (\neg y \wedge z)$



Removal of duplicate leaves

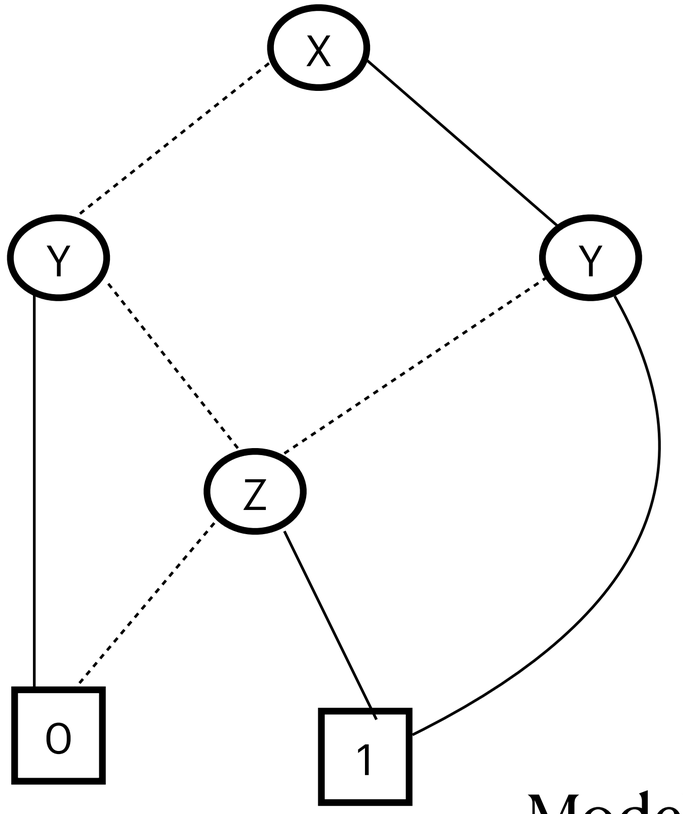
Removal of duplicate tests

Removal of duplicate sub-tree

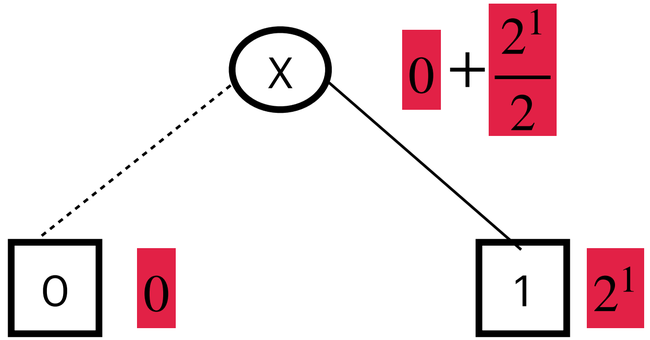


Model Counting

$F = (x \wedge y) \vee (\neg y \wedge z)$



Key Observation: We are fixing a variable as we move from the child to the parent node.

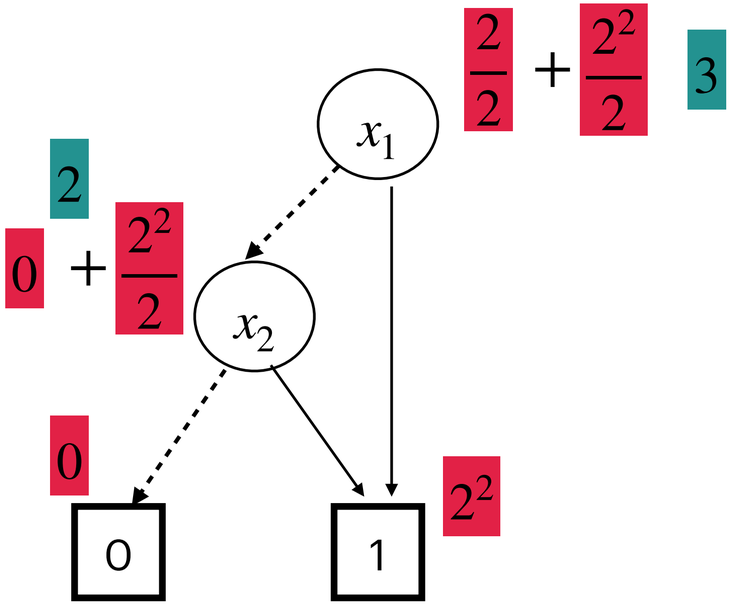


Bottom-up approach.

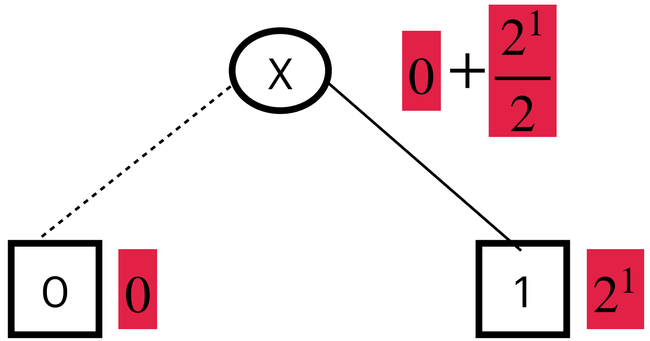
Model Counting in ROBDD?

Model Counting

$F = x_1 \vee x_2$



Key Observation: We are fixing a variable as we move from the child to the parent node.

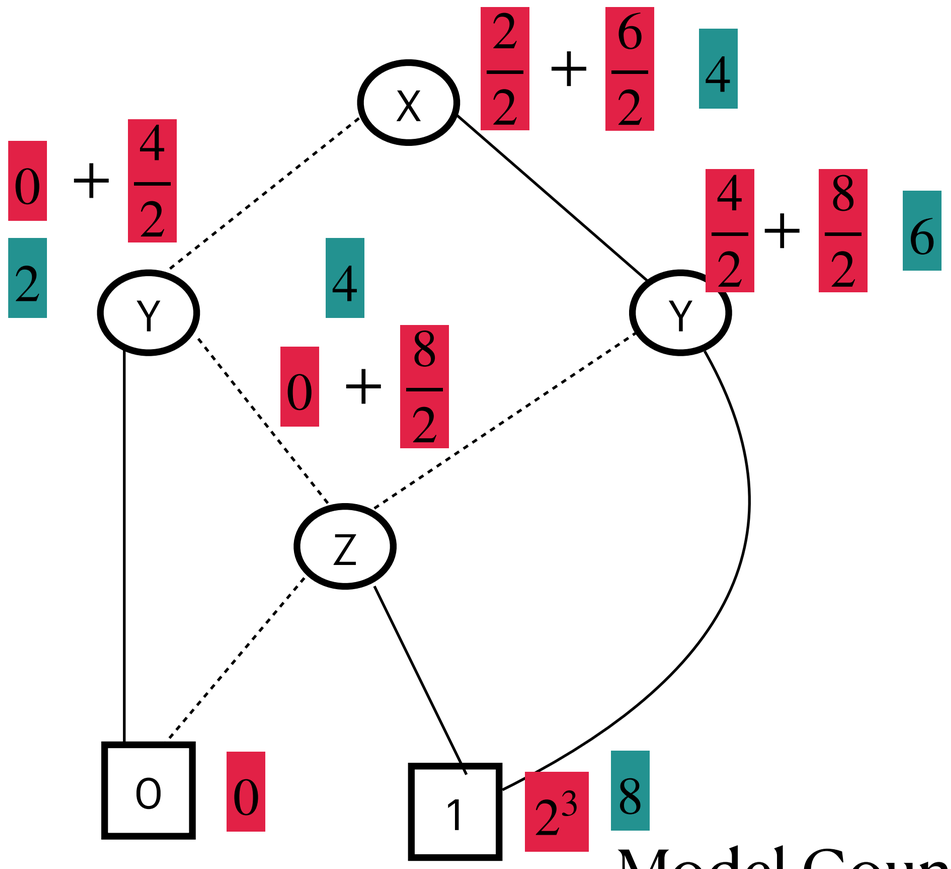


Bottom-up approach.

Model Counting in ROBDD?

Model Counting

$$F = (x \wedge y) \vee (\neg y \wedge z)$$



$$|Models(F)| = 4$$

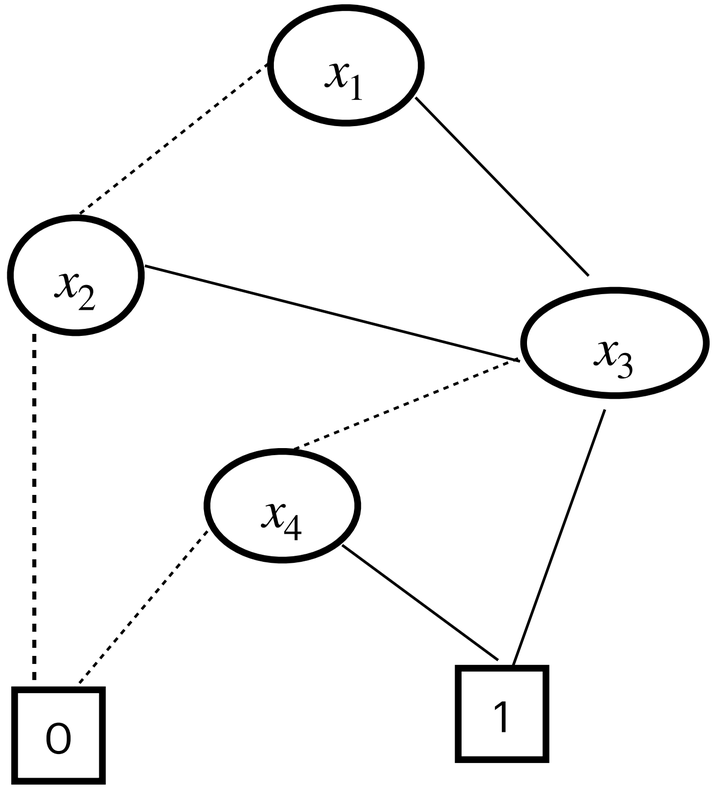
Model Counting in ROBDD?

Model Counting

$F = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$ $\quad$ $x_1 > x_2 > x_3 > x_4$

Model Counting

$F = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$ $\quad\quad$ $x_1 > x_2 > x_3 > x_4$



ROBDD vs CNF

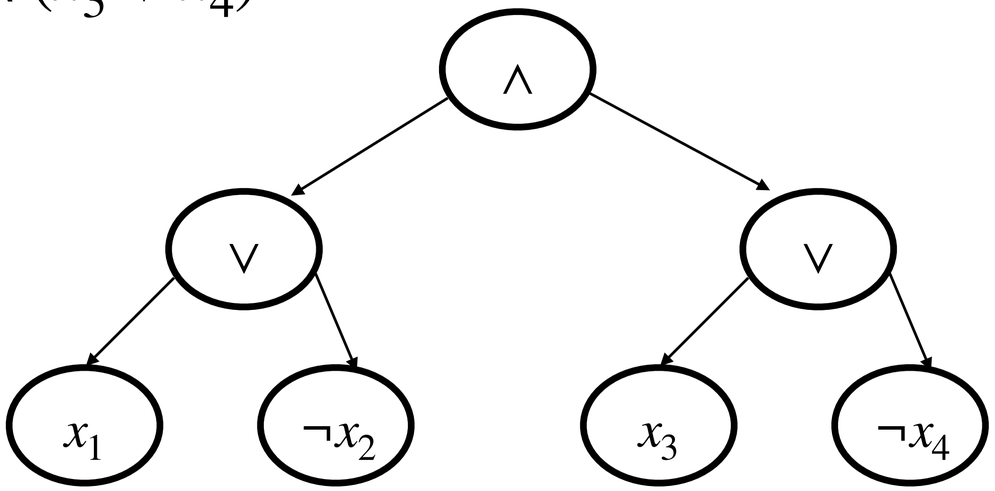
	CNF	ROBDD
SAT	NP-Hard	$O(\lvert F_{ROBDD} \rvert)$
Model Count	#P	$O(\lvert F_{ROBDD} \rvert)$
UNSAT	Co-NP	$O(1)$

Different Compilation Forms

NNF: Normal Negation Form

1. Each non-terminal node is either $\wedge$ or $\vee$
2. Each terminal node is either a literal or 0 or 1

$F = (x_1 \vee \neg x_2) \wedge (x_3 \vee x_4)$



Different Compilation Forms

d-NNF: Deterministic Normal Negation Form (*Darwiche* 1998)

A NNF is deterministic if for every $\vee$ (OR) node with children $\{c_1, c_2, \ldots, c_k\}$ following holds:

$$\forall i \neq j \; Models(c_i) \cap Models(c_j) = \emptyset$$

Any two children of $\vee$ (OR) node don't share models

Different Compilation Forms

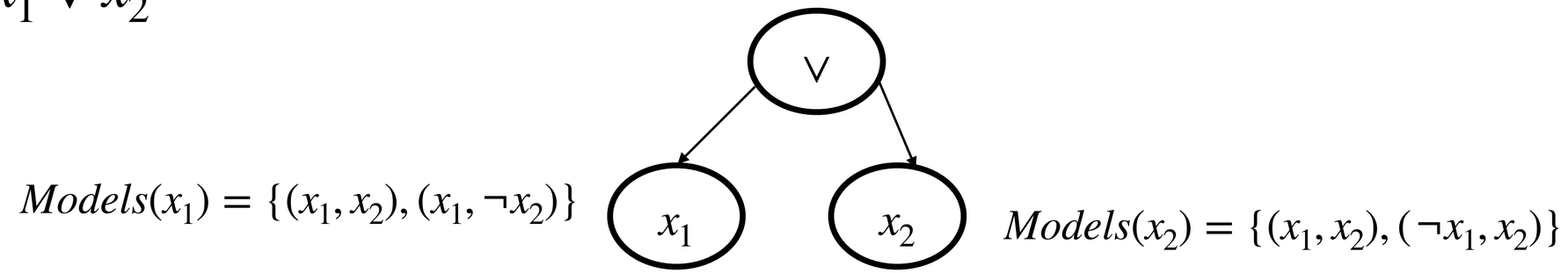
d-NNF: Deterministic Normal Negation Form (*Darwiche* 1998)

A NNF is deterministic if for every ∨ (OR) node with children $\{c_1, c_2, \ldots, c_k\}$ following holds:

$$\forall i \neq j \; Models(c_i) \cap Models(c_j) = \varnothing$$

Any two children of ∨ (OR) node don't share models

$F = x_1 \vee x_2$



Different Compilation Forms

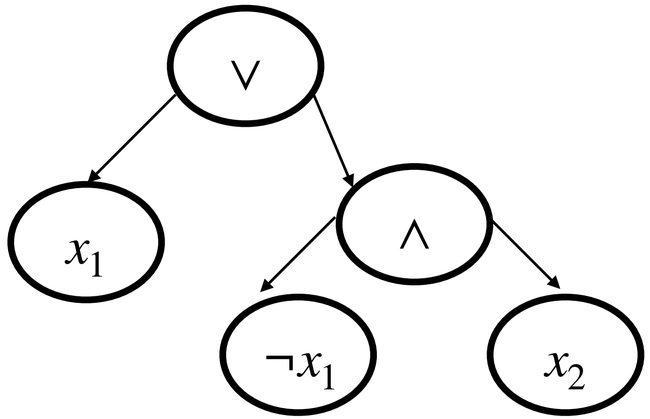
d-NNF: Deterministic Normal Negation Form (*Darwiche* 1998)

A NNF is deterministic if for every $\vee$ (OR) node with children $\{c_1, c_2, \ldots, c_k\}$ following holds:

$$\forall i \neq j \; Models(c_i) \cap Models(c_j) = \emptyset$$

Any two children of $\vee$ (OR) node don't share models

$F = x_1 \vee x_2$



$F \; in \; d - NNF$

Different Compilation Forms

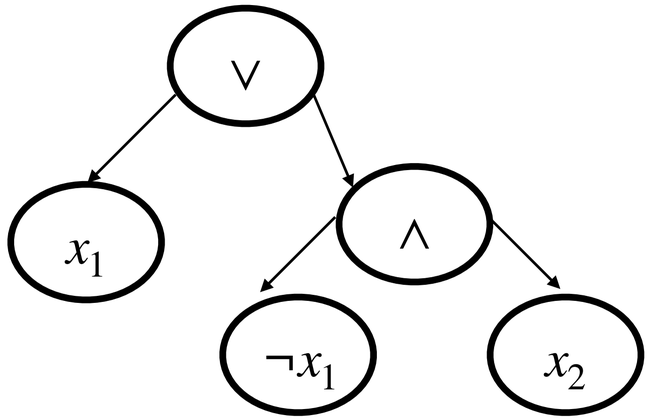
DNNF: Decomposable Normal Negation Form (*Darwiche* 2011)

A NNF is decomposable if for every $\wedge$ (AND) node with children $\{c_1, c_2, \ldots, c_k\}$ following holds:

$$\forall i \neq j \; Vars(c_i) \cap Vars(c_j) = \emptyset$$

Any two children of $\wedge$ (AND) node don't share variables/literals

$F = x_1 \vee x_2$

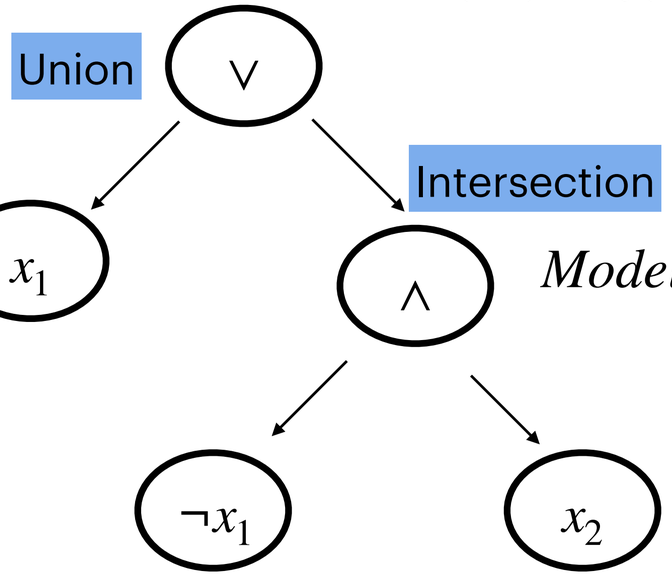


F in DNNF

Model Counting in d-DNNF

$F = x_1 \vee x_2$

$Models(\vee) = \{(x_1, x_2), (x_1, \neg x_2), (\neg x_1, x_2)\}$



$Models(x_1) = \{(x_1, x_2), (x_1, \neg x_2)\}$

$Models(\wedge) = \{(\neg x_1, x_2)\}$

$Models(\neg x_1) = \{(\neg x_1, x_2), (\neg x_1, \neg x_2)\}$

$Models(x_2) = \{(\neg x_1, x_2), (\neg x_1, x_2)\}$

$F \text{ in } d-DNNF$

We can’t store models at every node! We need to store count!

Model Counting in d-DNNF

(Suggestion from the class) Please check if is it correct!!!!

Model count of a terminal node:

1. If node is 0, then Model count is 0
2. If node is 1, then Model count is $2^{|Vars(F)|}$
3. If node is a literal, Model count is $2^{|Vars(F)-1|}$

Model count of a AND node with children $\{c_1, c_2, \ldots, c_k\}$

$$\frac{\prod_{i \in [1,k]} ModelCount(c_i)}{2^{|Vars(F)| \times (k-1)}}$$

Model count of a OR node with children $\{c_1, c_2, \ldots, c_k\}$

$$\sum_{i \in [1,k]} ModelCount(c_i)$$

Model Counting in d-DNNF

(Suggestion from the class) Please check if is it correct!!!!

Model count of a terminal node:

1. If node is 0, then Model count is 0
2. If node is 1, then Model count is 1
3. If node is a literal, Model count is 1

Model count of a AND node with children $\{c_1, c_2, \ldots, c_k\}$

$$\prod_{i \in [1,k]} ModelCount(c_i)$$

Model count of a OR node with children $\{c_1, c_2, \ldots, c_k\}$

$$\sum_{i \in [1,k]} (ModelCount(c_i) \times 2^{\left| \bigcup_{i \in [1,k]} Vars(c_i) - Vars(c_i) \right|})$$