COL:750

Foundations of Automatic Verification

Instructor: Priyanka Golia

Course Webpage 

https://priyanka-golia.github.io/teaching/COL-750/index.html

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 $\{c_1, c_2, \ldots, c_k\}$

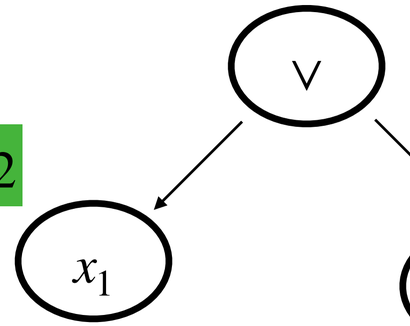
$$\sum_{i \in [1,k]} (ModelCount(c_i) \times 2^{\left| \bigcup_{i \in [1,k]} Vars(c_i) - Vars(c_i) \right|})$$

Model Counting in d-DNNF

Model count of a terminal node:

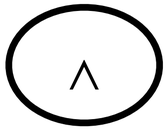
1. If node is 0, then Model count is 0
2. If node is 1, then Model count is $2^{|Vars(F)|}$
3. If node is a literal, Model count is $2^{|Vars(F)-1|}$

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



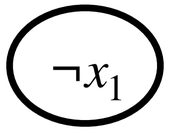
ModelCount(x_1) = 2

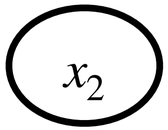
$Models(x_1) = \{(x_1, x_2), (x_1, \neg x_2)\}$



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

ModelCount($\neg x_1$) = 2





ModelCount(x_2) = 2

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

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

Model Counting in d-DNNF

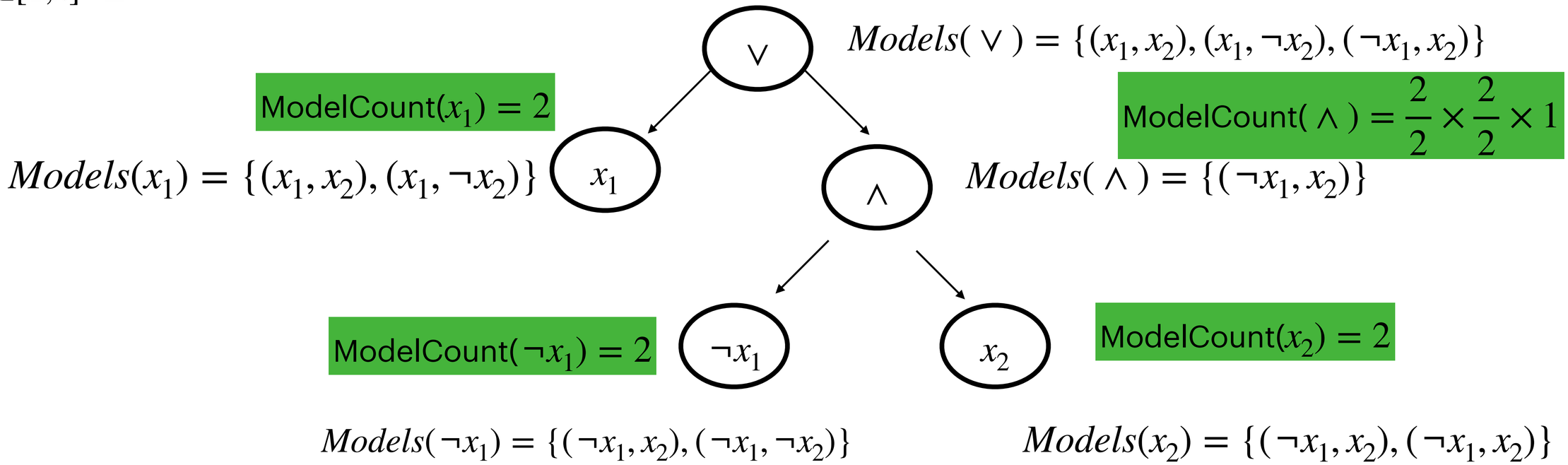
Model count of a AND node with children $\{c_1, c_2, \ldots, c_k\}$

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

Children don't share a literal, each child may wrongly assign these missing literals and thus overcount.

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

we need to account for the fact that the variables that are not assigned in $\wedge$

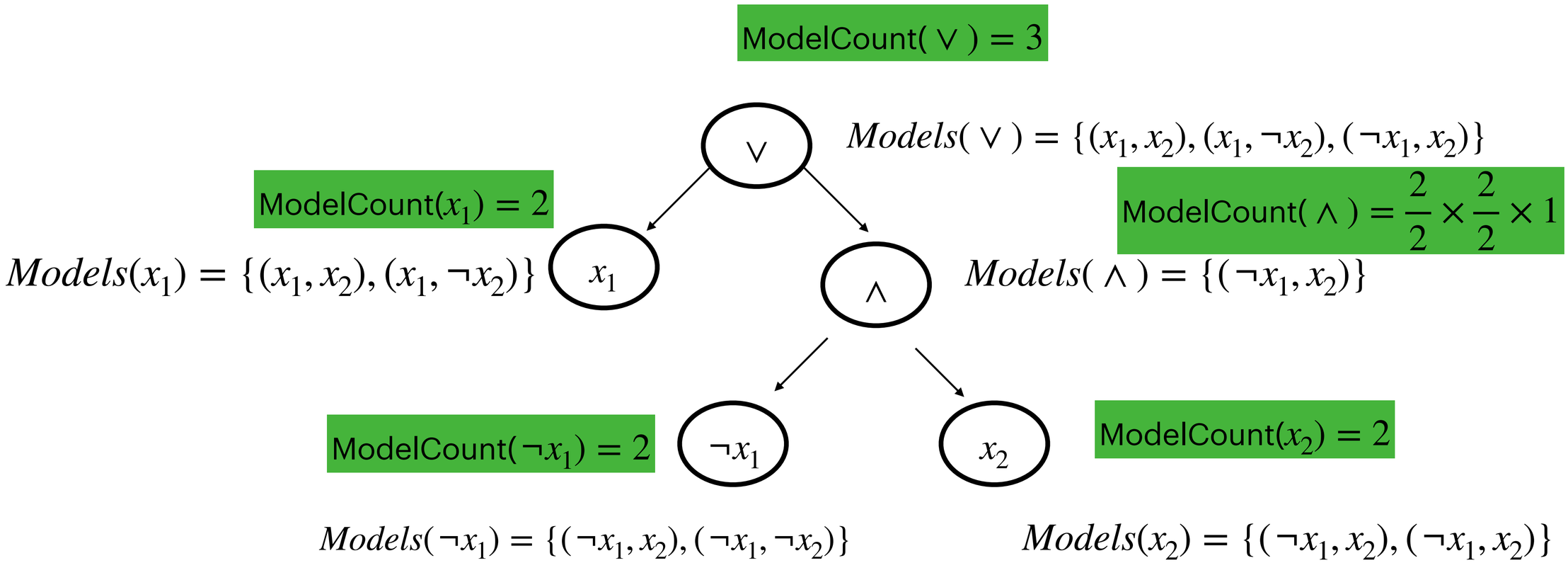


Model Counting in d-DNNF

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

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

Children don't share models



Model Counting in d-DNNF

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 $\{c_1, c_2, \ldots, c_k\}$

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

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

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

Model Counting in d-DNNF

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

$$F = (x_1 \vee x_2 \vee x_3)$$

In order to convert this to d-NNF,
Shannan Expansion:

$$F(x_1, x_2) = F(1,x_2) \vee F(0,x_2)$$

$$(x_1 \wedge (1 \vee x_2 \vee x_3)) \vee (\neg x_1 \wedge (0 \vee x_2 \vee x_3))$$

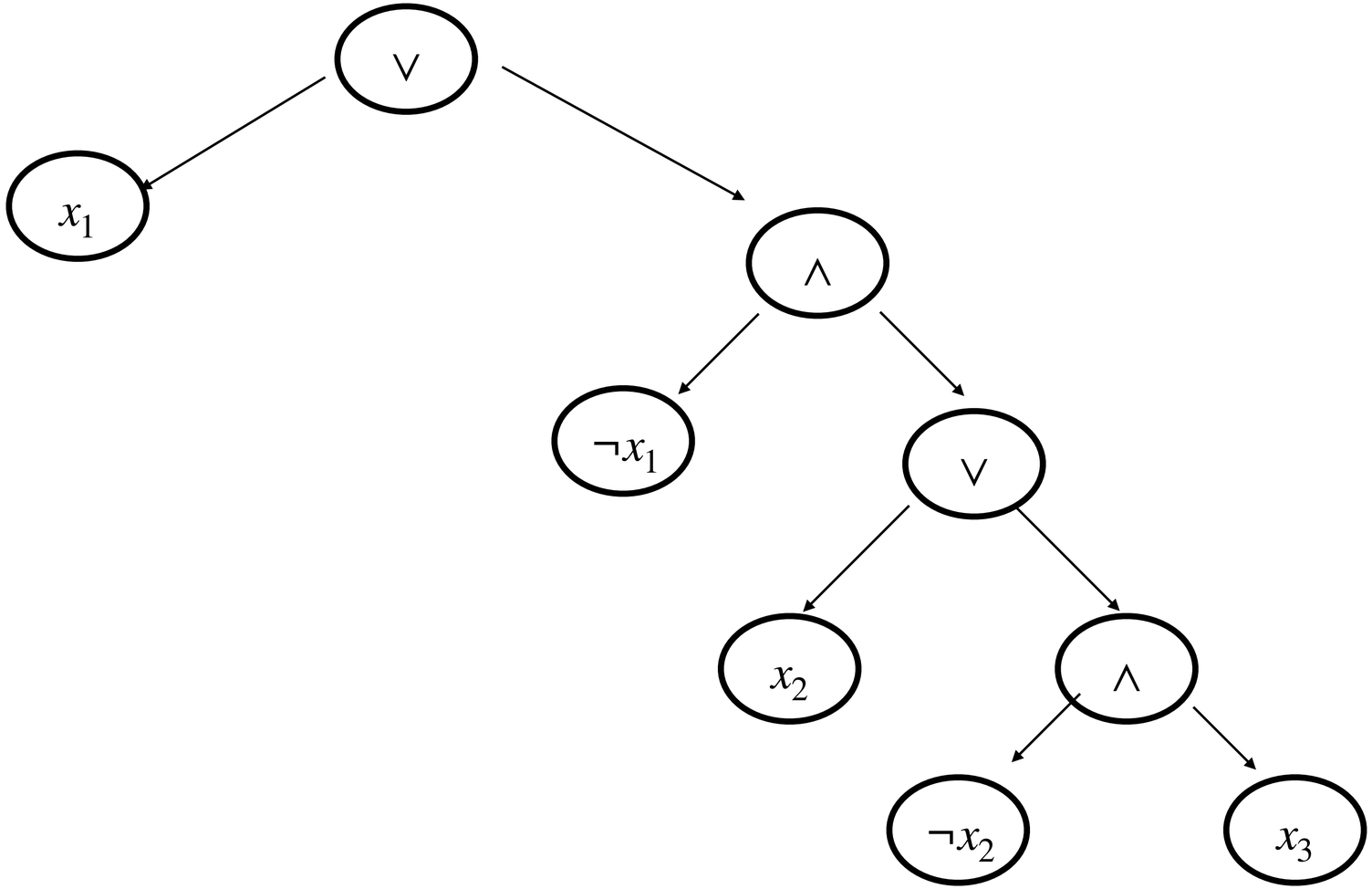
$$(x_1) \vee (\neg x_1 \wedge (x_2 \vee x_3))$$

$$(x_1) \vee (\neg x_1 \wedge ((x_2 \wedge (1 \vee x_3)) \vee (\neg x_2 \wedge (0 \vee x_3))))$$

$$(x_1) \vee (\neg x_1 \wedge (x_2 \vee (\neg x_2 \wedge x_3)))$$

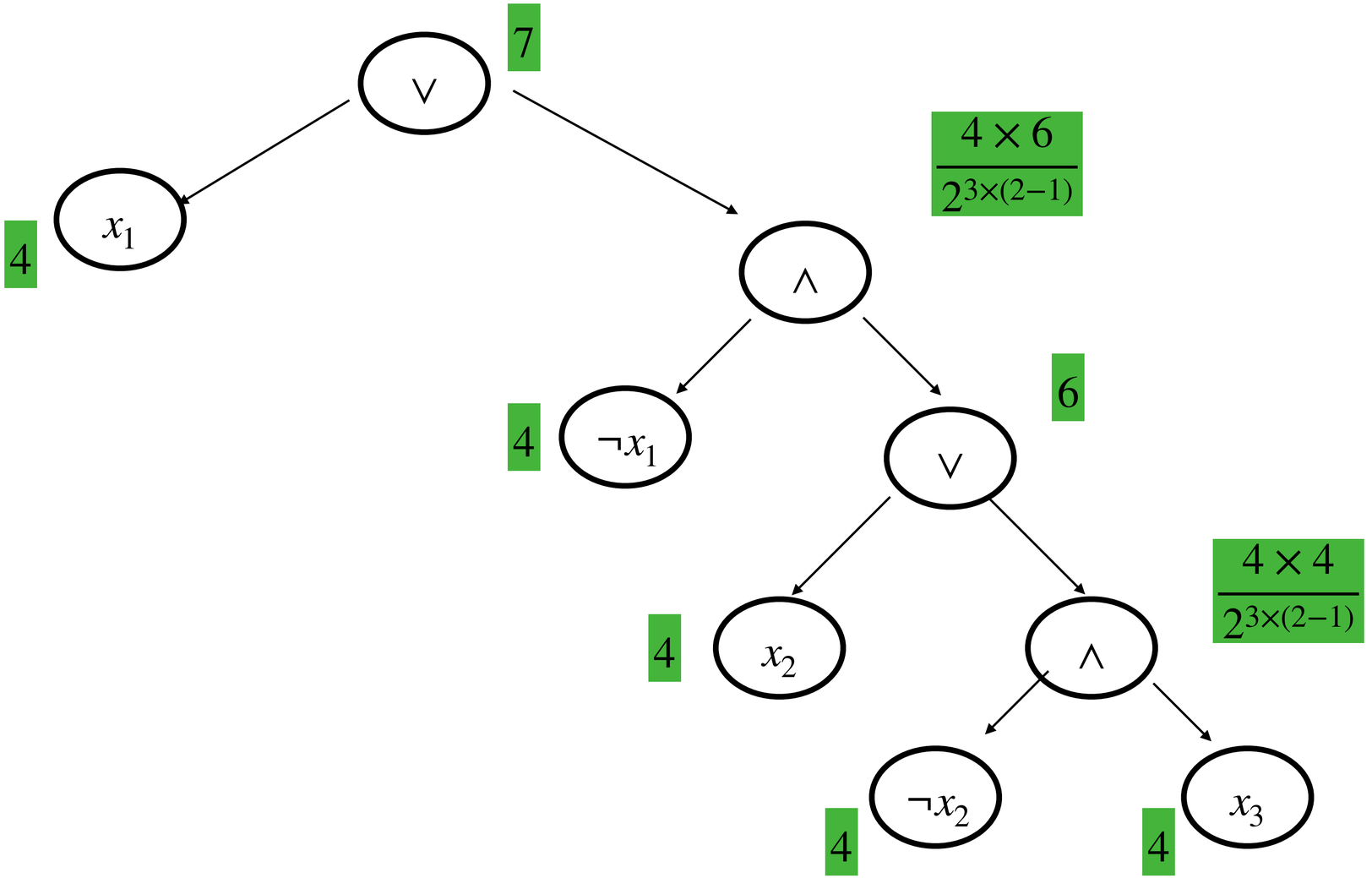
Model Counting in d-DNNF

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



Model Counting in d-DNNF

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

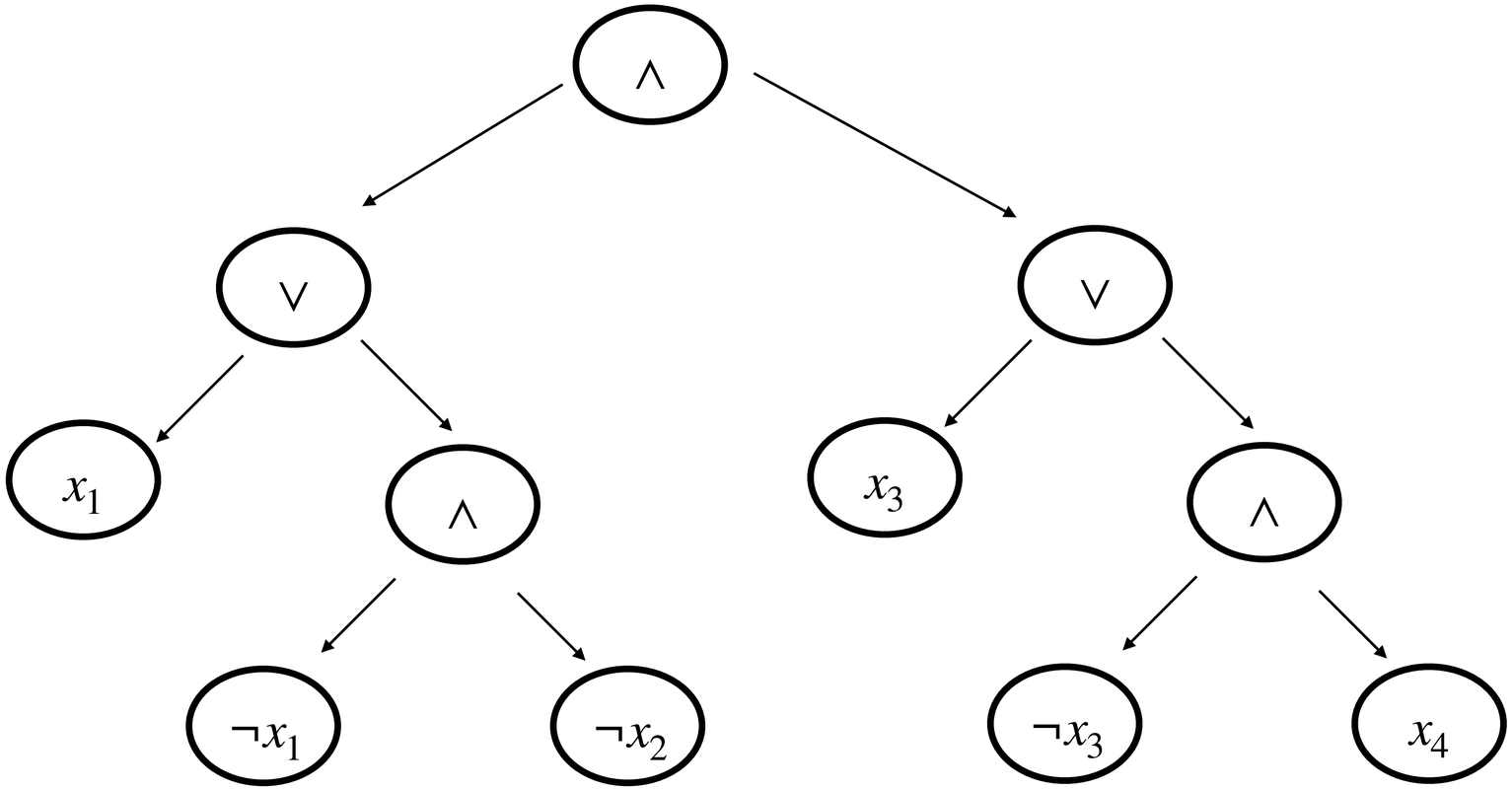


Model Counting in d-DNNF

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

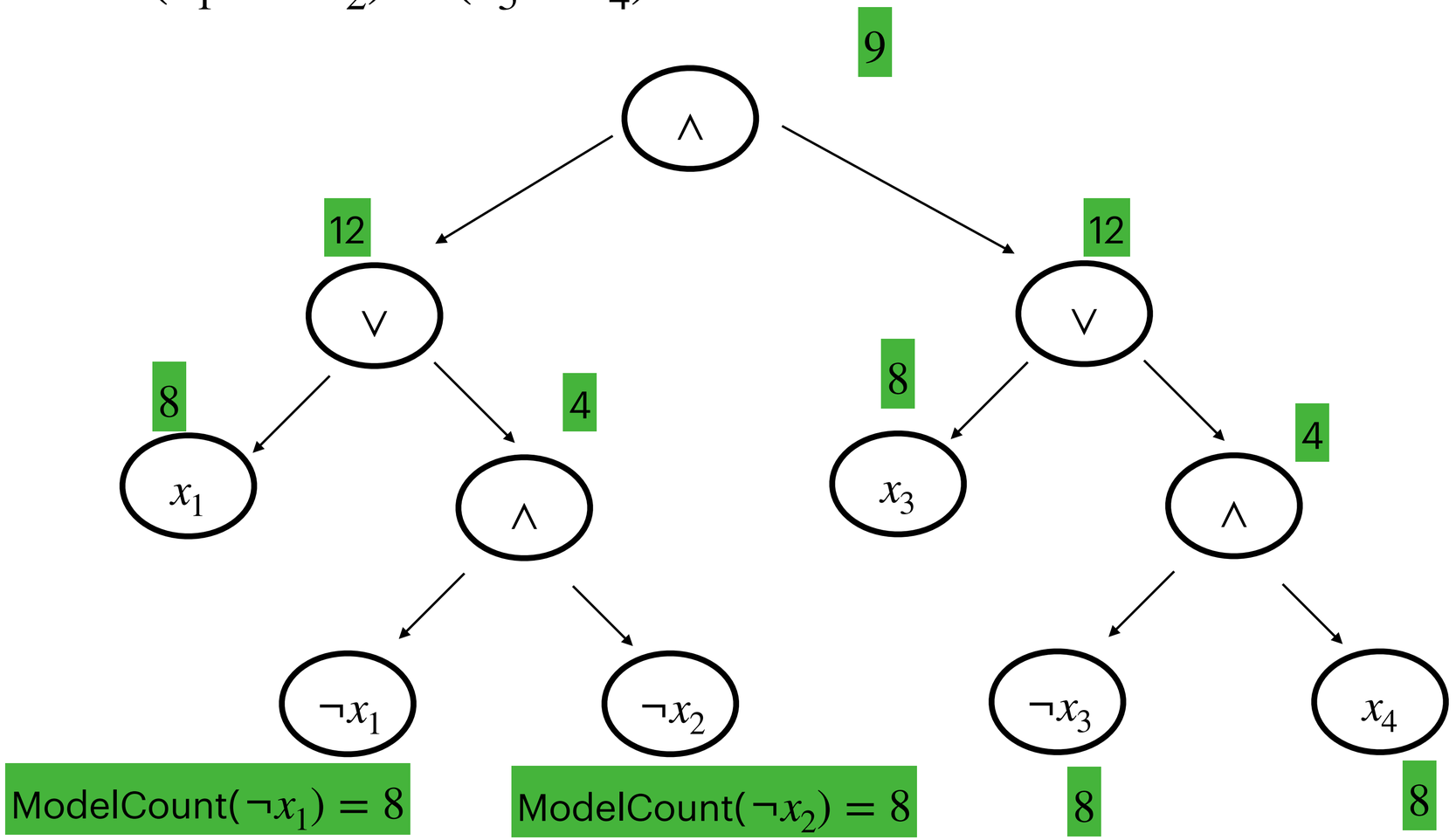
Model Counting in d-DNNF

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



Model Counting in d-DNNF

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



Model Counting in d-DNNF

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

Shannon Expansion on common variables.

$$F = x_1 \wedge ((1 \vee x_2) \wedge (\neg 1 \vee x_3)) \quad \vee (\neg x_1 \wedge ((0 \vee x_2) \wedge (\neg 0 \vee x_3)))$$

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

Model Counting in d-DNNF

Tools like d4, c2d, Dsharp for conversion

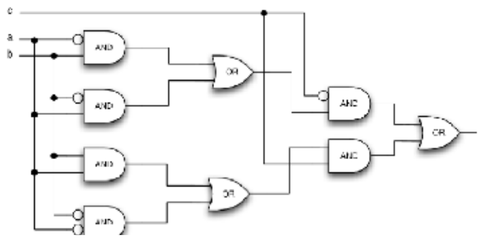
Just like ROBDD, may result in exponential size formula, but model counting is linear in the size of the formula

Efficient model counter, GANAK

By Shubham Sharma, a dual-degree student from IITK as his MTP project

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?

Why S doesn’t satisfy P?

Computing UNSAT core of a formula

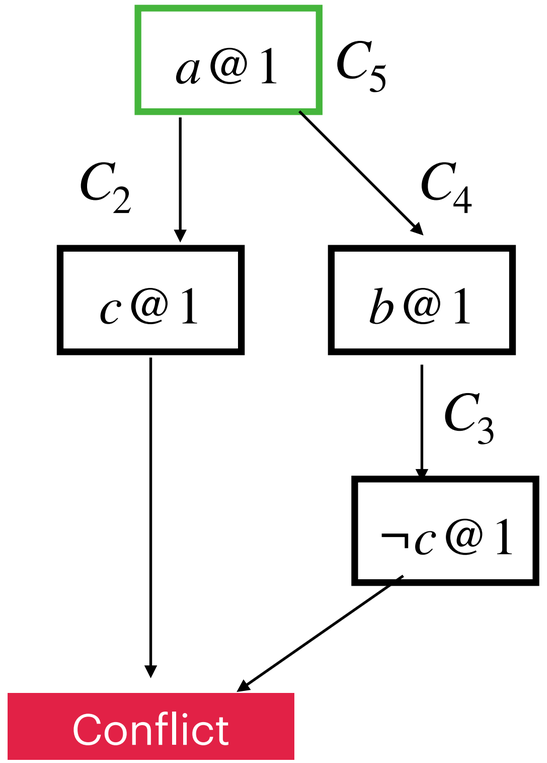
UNSAT Core: Given an unsatisfiable Boolean formula F in CNF, a subset of its clauses whose conjunction is also unsatisfiable is called an UNSAT core of F.

Computing UNSAT core of a formula

UNSAT Core: Given an unsatisfiable Boolean formula F in CNF, a subset of its clauses whose conjunction is also unsatisfiable is called an UNSAT core of F.

$$F = (a \vee b) \wedge (\neg a \vee c) \wedge (\neg b \vee \neg c) \wedge (\neg a \vee b) \wedge (a)$$

$$\text{UNSAT Core} = \{C_2, C_3, C_4, C_5\}$$



Computing UNSAT core of a formula

UNSAT Core: Given an unsatisfiable Boolean formula F in CNF, a subset of its clauses whose conjunction is also unsatisfiable is called an UNSAT core of F.

$c_1 = a \vee \neg c$ $\quad$ $c_3 = \neg b \vee c$ $\quad$ $c_5 = b \vee c$

$c_2 = b$ $\quad$ $c_4 = \neg b \vee \neg c$ $\quad$ $c_6 = \neg a \vee b \vee \neg c$

$F = c_1 \wedge c_2 \wedge c_3 \wedge c_4 \wedge c_5 \wedge c_6$ $\quad$ How many different unsat cores for F?

$UC_1 = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ $\quad$ $UC_4 = \{c_1, c_2, c_3, c_4, c_6\}$ $\quad$ $UC_7 = \{c_1, c_2, c_3, c_4\}$

$UC_2 = \{c_2, c_3, c_4, c_5, c_6\}$ $\quad$ $UC_5 = \{c_2, c_3, c_4, c_5\}$ $\quad$ $UC_8 = \{c_2, c_3, c_4\}$

$UC_3 = \{c_1, c_2, c_3, c_4, c_5\}$ $\quad$ $UC_6 = \{c_2, c_3, c_4, c_6\}$ $\quad$ $UC_9 = \{c_1, c_3, c_4, c_5, c_6\}$

UNSAT Core Minimal Unsatisfiable Set.

Consider a subset $M \subseteq C$, where C is a set of all clauses of Formula F

Minimal Unsatisfiable Set (MUS): M is a MUS of F if and only if M is unsatisfiable, **and** all proper subsets of M are satisfiable.

$$F = a \wedge \neg a \wedge b \wedge (\neg a \vee \neg b) \qquad M_1 = \{a, \neg a\} \qquad M_2 = \{a, b, (\neg a \vee \neg b)\}$$

A MUS is an unsatisfiable set that can't be reduced without causing it to become satisfiable.

UNSAT Core Minimal Unsatisfiable Subset.

$c_1 = a \vee \neg c$ $c_3 = \neg b \vee c$ $c_5 = b \vee c$

$c_2 = b$ $c_4 = \neg b \vee \neg c$ $c_6 = \neg a \vee b \vee \neg c$

$F = c_1 \wedge c_2 \wedge c_3 \wedge c_4 \wedge c_5 \wedge c_6$ Minimal unsat cores for F?

$UC_1 = \{c_1, c_2, c_3, c_4, c_5, c_6\}$

$UC_2 = \{c_2, c_3, c_4, c_5, c_6\}$

$UC_3 = \{c_1, c_2, c_3, c_4, c_5\}$

$UC_4 = \{c_1, c_2, c_3, c_4, c_6\}$

$UC_5 = \{c_2, c_3, c_4, c_5\}$

$UC_6 = \{c_2, c_3, c_4, c_6\}$

$UC_7 = \{c_1, c_2, c_3, c_4\}$

$UC_8 = \{c_2, c_3, c_4\}$

$UC_9 = \{c_1, c_3, c_4, c_5, c_6\}$

UNSAT Core Minimal Correction Set.

Consider a subset $M' \subseteq C$, where C is a set of all clauses of Formula F

Minimal Correction Set (MCS): M' is a MCS of F if and only if $C \backslash M'$ is satisfiable, **and** $\forall m \in M', C \backslash \{M' \backslash m\}$ is unsatisfiable.

$$F = a \wedge \neg a \wedge b \wedge (\neg a \vee \neg b) \qquad M'_1 = \{a\} \quad M'_2 = \{\neg a, b\} \quad M'_3 = \{\neg a, \neg a \vee \neg b\}$$

An MCS is a minimal set of clauses whose removal from a formula F makes F satisfiable.

UNSAT Core Minimal Correction Set.

$c_1 = a \vee \neg c$ $c_3 = \neg b \vee c$ $c_5 = b \vee c$

$c_2 = b$ $c_4 = \neg b \vee \neg c$ $c_6 = \neg a \vee b \vee \neg c$

$F = c_1 \wedge c_2 \wedge c_3 \wedge c_4 \wedge c_5 \wedge c_6$ Minimal correction cores for F?

MCS_1 $\{c_3\}$

MCS_2 $\{c_4\}$

How are MUSes and MCSes related?

$$F = a \wedge \neg a \wedge b \wedge (\neg a \vee \neg b)$$

MUSes $M_1 = \{a, \neg a\}$ $M_2 = \{a, b, (\neg a \vee \neg b)\}$

MCSes $M'_1 = \{a\}$ $M'_2 = \{\neg a, b\}$ $M'_3 = \{\neg a, \neg a \vee \neg b\}$

Hitting Set

A hitting set H of a collection of sets S is a set that "hits" every set in S, that is,
$\forall s \in S, H \cap s \neq \emptyset$

H has non empty intersection with every set s of S.

$$S = \{\{a, b\}, \{a, c\}, \{c, d\}\}$$

$$H_1 = \{a, c\} \qquad H_2 = \{a, b, c\} \qquad H_3 = \{a, c, d\} \qquad H_4 = \{a, d\}$$

A minimal hitting set is a hitting set such that no strict subset of it is also a hitting set.

$$H_1 = \{a, c\} \qquad H_4 = \{a, d\}$$

MUSes and MCSes

Every MCS is a minimal hitting set of the set of MUSes

Every MUS is a minimal hitting set of the set of MCSes.

$$F = a \wedge \neg a \wedge b \wedge (\neg a \vee \neg b)$$

MUSes $M_1 = \{a, \neg a\}$ $M_2 = \{a, b, (\neg a \vee \neg b)\}$

MCSes $M'_1 = \{a\}$ $M'_2 = \{\neg a, b\}$ $M'_3 = \{\neg a, \neg a \vee \neg b\}$

Can we come up with an algorithm to find MUS?

Computing MUS

Key Observation: each clause is a critical clause in MUS

Consider a subset $C' \subseteq C$, where C is a set of all clauses.
C' is a said to be Critical for formula F if removal of C' from F, causes F to become satisfiable

Find an UNSAT Core UC.

For each clauses $c \in UC$

If c is NOT a critical clause in UC

$$UC \leftarrow UC \backslash \{c\}$$

How do we check if clause is a critical clause or not ?

Return UC

Computing MUS

Key Observation: each clause is a critical clause in MUS

Find an UNSAT Core UC.

For each clauses $c \in UC$

If NOT CheckSAT($F \backslash \{c\}$)

$UC \leftarrow UC \backslash \{c\}$

Return UC

$F = a \wedge \neg a \wedge b \wedge (\neg a \vee \neg b)$

Computing MUS

Key Observation: each clause is a critical clause in MUS

Find an UNSAT Core UC.

UnknownClauses $\leftarrow Clauses(UC)$

CriticalClauses $\leftarrow \emptyset$

While ($UnknownClauses \neq \emptyset$) {

Choose a clauses c in UnknownClauses

UnknownClauses $\leftarrow$ UnknownClauses $\setminus\ c$

$(SAT?, \sigma, UC') \leftarrow$ CheckSAT($UnknownClauses \cup CriticalClauses$)

If SAT:

CriticalClauses $\leftarrow$ CriticalClauses $\cup\ c$

Else:

UnknownClauses $\leftarrow$ UnknownClauses $\cap$ Clauses(UC')

}

Return CriticalClauses

Adding a single clause at a time to Critical Clauses!! Can we do better?

Critical Clauses

$(SAT?, \sigma, UC') \leftarrow$ CheckSAT($UnknownClauses \cup CriticalClauses \cup \neg c$)

$\sigma \not\models c$

Let there be a another satisfying assignment σ' such that $\sigma' = \sigma_{\downarrow(\neg v)}$ where $v \in Vars(c)$

$\sigma' \models c \quad \sigma' \not\models UC$ *UC is UNSAT*

$\exists C' \in UC \backslash c, s\,.\,t\,., \sigma' \not\models C'$ There has to be at least a clause c' in $UC \backslash c$, such that, $\sigma' \not\models c'$

Clauses in C' are also critical clauses.

Critical Clauses

Recursive Model Rotation (RMR)

$$UC = (\neg a \vee \neg b) \wedge (b \vee \neg c) \wedge (a \vee b) \wedge (a \vee \neg b) \wedge (b \vee c)$$

Check if $(\neg a \vee \neg b)$ is a critical clause or not?

$$\text{CheckSAT}(UC \backslash (\neg a \vee \neg b) \wedge (a \wedge b))$$

CheckSAT($UnknownClauses \cup CriticalClauses \cup \neg c$)

UnknownClauses = $UC \backslash c$, Critical Clauses = $\varnothing$

$$\sigma = \langle a = 1, b = 1, c = 1 \rangle$$

$$\sigma' = \sigma_{\downarrow \neg v}, v \in Vars(c) \; is \; \langle a = 0, b = 0 \rangle$$

$\sigma' \models c$ $\sigma' \not\models UC$

$\sigma' \not\models (a \vee b)$ This is also a critical clause.

Computing MUS

Key Observation: each clause is a critical clause in MUS

Find an UNSAT Core UC.

UnknownClauses, CriticalClauses $\leftarrow$ $Clauses(UC), \emptyset$

While ($UnknownClauses \neq \emptyset$) {

Choose a clauses c in UnknownClauses

UnknownClauses $\leftarrow$ UnknownClauses $\setminus$ c

$(SAT?, \sigma, UC') \leftarrow$ CheckSAT($UnknownClasues \cup CriticalClauses \cup \neg c$)

If SAT:

CriticalClauses $\leftarrow$ CriticalClauses $\cup$ c

MoreCriticalClauses $\leftarrow$ RMR(σ, c, $Unknowclauses$, $CriticalClasues$)

CriticalClauses $\leftarrow$ CriticalClauses $\cup$ MoreCriticalClauses

UnknownClauses $\leftarrow$ UnknownClauses $\setminus$ MoreCriticalClauses

Else:

UnknownClauses $\leftarrow$ UnknownClauses $\cap$ Clauses(UC')

}

Return CriticalClauses

Computing MUS

$$F = a \wedge \neg a \wedge b \wedge (\neg a \vee \neg b)$$

$\text{UnknownClauses} = \{a, \neg a, b, \neg a \vee \neg b\}$ Critical Clauses $\{\varnothing\}$

Computing MUS

$$F = a \wedge \neg a \wedge b \wedge (\neg a \vee \neg b)$$

UnknownClauses = $\{a, \neg a, b, \neg a \vee \neg b\}$ Critical Clauses $\{\varnothing\}$

1. Choose $\{a\}$, check if it is critical

$$\text{CheckSAT } ((\neg a \wedge b \wedge (\neg a \vee \neg b)) \wedge \neg a)$$

$$\text{It is SAT. } \sigma = \langle a = 0, b = 1 \rangle$$

2. Look for other critical clauses.

$$\sigma' = \sigma_{\downarrow(\neg(a=0)} = \langle a = 1 \rangle$$

$\sigma' \not\models \neg a$ This will also be added to critical clauses.

UnknownClauses = $\{b, \neg a \vee \neg b\}$ Critical Clauses $\{a, \neg a\}$

Computing MUS

$$F = a \wedge \neg a \wedge b \wedge (\neg a \vee \neg b)$$

UnknownClauses = $\{b, \neg a \vee \neg b\}$ Critical Clauses $\{a, \neg a\}$

1. Choose $\{b\}$, check if it is critical

CheckSAT $(((a \wedge \neg a \wedge (\neg a \vee \neg b)) \wedge \neg b))$ It is UNSAT.

UnknownClauses = $\{\neg a \vee \neg b\}$ Critical Clauses $\{a, \neg a\}$

1. Choose $\{\neg a \vee \neg b\}$, check if it is critical

CheckSAT $((a \wedge \neg a) \wedge \neg(\neg a \vee \neg b))$ It is UNSAT.

UnknownClauses = $\varnothing$ Critical Clauses $\{a, \neg a\}$ MUS: $\{a, \neg a\}$