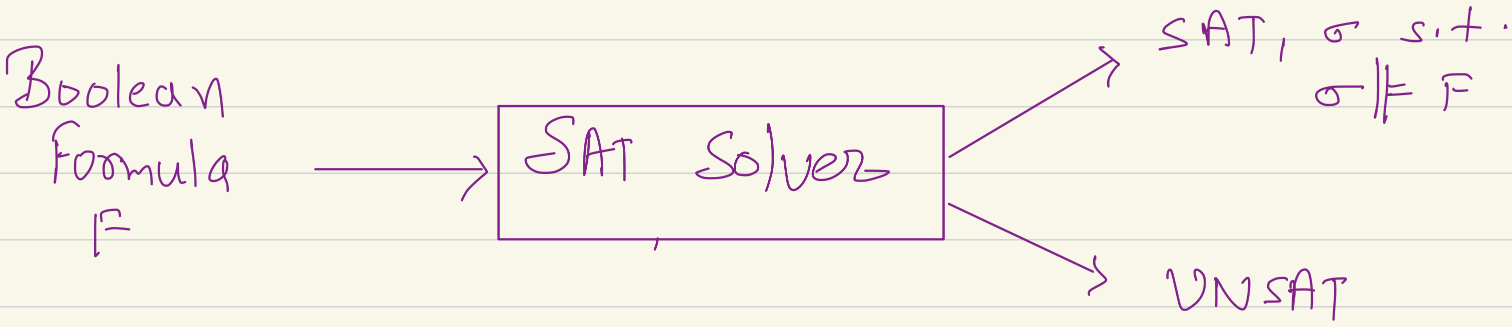


How do we trust the implementation of SAT solver?

If solver says F is SAT & we have σ, then we can verify it

But what if solver says F is UNSAT?

More SAT solver - kissat

- cryptominisAT

→ ⋮

> 20k lines of code

there has been cases of <u>SAME</u> bug in more than one solver.

[taken from GANAK (model counter repo)] https://github.com/meelgroup/ganak/issues/8

↳ A instance with 87 variables & 25 clauses

GANAK, sharpSAT	MiniC2D CAchet
967445862998626248587584	9674458629986261412085760

GANAK

↳ proposed & developed by dual degree CS IIT Kanpur student, 2020 batch

We ask SAT Solver to produce a proof for the UNSAT results

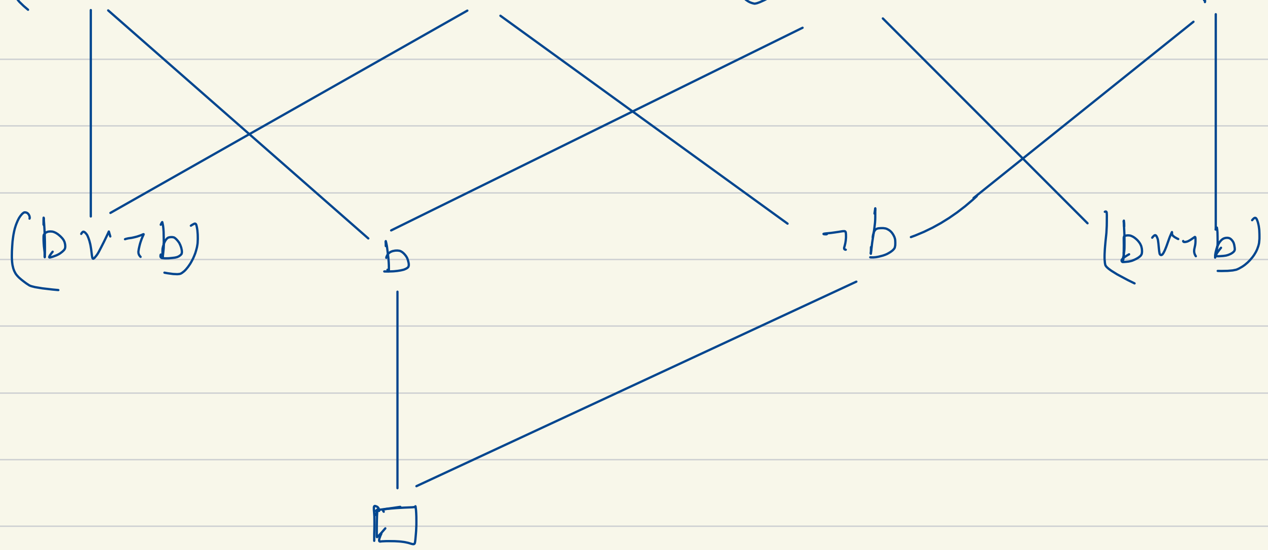
Say:

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

F is UNSAT, why?

How do we prove that there is no solution?

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



<u>Using Resolution</u>

RUP (Reverse Unit Propagation)

↳ write each <u>derived</u> clauses one after the other.

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

RUP Proof :

$(b) \wedge$ → derived from $(a \vee b) \wedge (\neg a \vee b)$

$(\neg b)$ → derived from $(a \vee \neg b) \wedge (\neg a \vee \neg b)$

The idea is that the clauses in RUP proof must hold & they together leads to a conflict.

Boolean Formula F $\longrightarrow$ $\boxed{\text{SAT Solver}}$

$\longrightarrow$ SAT, σ s.t. $\sigma \models F$

$\longrightarrow$ UNSAT, proof
RUP, DRUP, DRAT, RAT, ...

Boolen Formula F + proof (RUP) $\longrightarrow$ $\boxed{\text{Proof checker}}$ $\longrightarrow$ validates the proof.

each clauses of proof holds true.

* To validate given RUP proof, you need to prove that i^{th} clauses in RUP holds true under all clauses of formula & upto $(i-1)^{th}$ clauses of RUP.

how to prof that C must holds true:

if $F \wedge \neg C \rightarrow$ UNSAT

then C must hold true.

* RUP :- each line of the Proof must be checkable by simple propagation.

UNSAT CORE

Given an unsatisfiable Boolean formula in CNF from, a subset of clauses of F whose conjunction is still unsatisfiable is called UNSAT CORE of the formula F.

Minimal Unsatisfiable Subset.

MUS : Consider $M \subseteq C$, where C is the set of all clauses of Formula F. M is a MUS of F if and only if M is unsatisfiable & all proper subsets of M is satisfiable.

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

Compute MUS

$\{(a), (\neg a)\}$

$\{(a), (b), (\neg a \vee \neg b)\}$

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

Minimal Correction Set

MCS : Consider $M' \subseteq C$, where C is the set of all clauses of Formula F. M' is called MCS if and only if C / M' is satisfiable & $\forall m \in M'$, $C \setminus \{M' \setminus m\}$ is unsatisfiable.

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

Compute MCS

MCS is a minimal set of removals from F that causes F to become satisfiable.

What is the relation b/w MCS & MVS ?

Hitting Set

A hitting set H of a collection of sets C is a set that "hits" every set in C, in the sense that it has non empty intersection with each such set: $\forall c \in \underbrace{C}_{\text{collection of sets}}, \underbrace{H}_{\text{hitting set}} \cap c \neq \phi$.

Knowing this, Now do you see the relation b/w MCS & MUS?

AllMUSes : Set containing all MUSes of formula F

AllMCSes : Set containing all MCSes of formula F.

→ $M \in$ AllMUSes, if and only if M is a minimal hitting set of AllMCSes.

→ Dually, $C \in$ AllMCSes if and only if C is a minimal hitting set of AllMUSes.

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

All MUSes : $\{ \{ (a), (\neg a) \}, \{ (a) \wedge (b) \wedge (\neg a \vee \neg b) \} \}$

All MCSes : $\{ \{a\}, \{ (\neg a), (b) \}, \{ (\neg a), (\neg a \vee \neg b) \} \}$

Minimal hitting set of All MUSes :

- $\{a\}$
- $\{\neg a, b\}$
- $\{\neg a, (\neg a \vee \neg b)\}$

Minimal hitting set of All MCSes :

- $\{ (a), (\neg a) \}$
- $\{ a, b, \neg a \vee \neg b \}$

Let C be the collection of clauses, such that $C \subseteq F$.

C is said to be critical for formula F, if:

- → C must be contained in every MUS of F.
- → C is an MCS of F.
  - ↳ Removal of C from F, causes F to become satisfiable.

Note: every clause in an MUS is critical for it.

Can we come up with algorithm to find MUS ?!

To compute MUS

Input : Unsatisfiable formula F., as set of clauses.

critical_clauses $\leftarrow \phi$

unknown_status $\leftarrow$ F.

choose one clauses from unknown_status,

? check if that is a "critical" clause or not?

if yes, add to critical_clauses

if not, ignore?

what else
can be done?

Q: How do we check if a clause "c" is "critical"?

Let $F' = \{F \setminus c\} \wedge (\neg c)$

if F' is SAT

↳

if F' is UNSAT

↳

Q: How do we check if a clause 'c' is "critical"?

Let $F' = \{F \setminus c\} \wedge (\neg c)$

if F' is SAT

$\hookrightarrow$ c is critical

if F' is UNSAT

$\hookrightarrow$ c is not critical; but we can work with UNSAT CORE of F'.

To compute MUS

Input: Unsatisfiable formula F., as set of clauses.

critical_clauses $\leftarrow \phi$

unknown_status $\leftarrow$ F.

while(unknown_status $\neq \phi$) do

{ c $\leftarrow$ choose c $\in$ unknown_status.

unknown_status $\leftarrow$ unknown_status \ c

(sat?, σ, UC) $\leftarrow$ SATSolver(critical_clauses $\cup$ unknown_status $\cup$ $\{\neg c\}$)

if sat then

critical_clauses $\leftarrow$ critical_clauses $\cup$ $\{c\}$

else

if UC $\subseteq$ critical_clauses $\cup$ unknown_clauses.

then unknown_clauses $\leftarrow$ unknown_clauses $\cap$ UC.

}

To compute MUS

Input: Unsatisfiable formula F, as set of clauses.

critical_clauses $\leftarrow \phi$

unknown_status $\leftarrow F$.

while(unknown_status $\neq \phi$) do

{ $c \leftarrow$ choose $c \in$ unknown_status.

unknown_status $\leftarrow$ unknown_status $\setminus c$

(sat?, σ, UC) $\leftarrow$ SATSolver(critical_clauses $\cup$ unknown_status $\cup \{\neg c\}$)

if sat then

critical_clauses $\leftarrow$ critical_clauses $\cup \{c\}$] can we do better?

else

if UC $\subseteq$ critical_clauses $\cup$ unknown_clauses.

then unknown_clauses $\leftarrow$ unknown_clauses $\cap$ UC.

}

Recursive Model Rotation (RMR)

Let us try to understand this via an example

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

let c be $(\neg x_1 \vee \neg x_2)$

To check if c is critical :-

$F' = (F \setminus \{\neg x_1 \vee \neg x_2\}) \wedge x_1 \wedge x_2$

$\sigma = \langle x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1 \rangle, \; \sigma \models F'$

so c is critical clause.

Recursive Model Rotation (RMR)

Let us try to understand this via an example

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

Let c be $(\neg x_1 \vee \neg x_2)$

To check if c is critical :-

$$F' = (F \setminus \{\neg x_1 \vee \neg x_2\}) \wedge x_1 \wedge x_2$$

$\sigma = \langle x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1 \rangle$, $\sigma \Vdash F'$

So c is critical clause.

Question is given c & σ, can we find other critical clauses?

Note that $\sigma \nVdash C$.

Let us consider another assignment σ', such that

$$\sigma' = \langle x_1 \mapsto 0, x_2 \mapsto 1, x_3 \mapsto 1 \rangle$$

$\sigma' = \sigma|_{\{\neg x\}}$

where $x \in C$.

Now notice that : $\sigma' \Vdash C$, but ofcourse $\sigma' \nVdash F$

there has to be at least one $c' \in F \setminus C$ such that $\sigma' \nVdash c$.

In this example : $(x_1 \vee \neg x_2)$

→ this a critical clause too.

To compute MUS (input unsatisfiable formula F)

critical_clauses $\leftarrow \phi$

unknown_status $\leftarrow$ F.

while (unknown_status $\neq \phi$) do

{ c $\leftarrow$ choose c $\in$ unknown_status.

unknown_status $\leftarrow$ unknown_status \ c

(sat?, σ, UC) $\leftarrow$ SATSolver(critical clauses $\cup$ unknown_status $\cup$ $\{\neg c\}$)

if sat then

critical_clauses $\leftarrow$ critical_clauses $\cup$ {c}

More_critical_cls $\leftarrow$ RMR(σ, c, critical_clauses, unknown_clauses)

critical_clauses $\leftarrow$ critical_clauses $\cup$ more_critical_cls.

unknown_clauses $\leftarrow$ unknown_clauses \ more_critical_cls

else if UC $\subseteq$ critical_clauses $\cup$ unknown_clauses.

then unknown_clauses $\leftarrow$ unknown_clauses $\cap$ UC.

}