# SAT and SMT solvers in practice

(Inspired by Elizabeth Polgreen)

Joachim Tilsted Kristensen

University of Oslo, November 16, 2022

# Agenda

- What is the satisfiability problem.
- Why is SAT interesting?
- What is an SMT solver.
- How to deploy a SAT/SMT solver.

# Repetition SAT

## Formulas

- Syntax : $\phi \ := \ p \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \Rightarrow \psi$

# Repetition SAT

## Formulas

- Syntax : $\phi \;:=\; p \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \Rightarrow \psi$

## Terminology

- Satisfiable : There exists a solution.
- Unsatisfiable : There are no solutions.
- Invalid : There exists an assignment which is not a solution.
- Valid : All assignments are solutions.

# Repetition SAT

## Formulas

- Syntax : $\phi ::= p \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \Rightarrow \psi$

## Terminology

- Satisfiable : There exists a solution.
- Unsatisfiable : There are no solutions.
- Invalid : There exists an assignment which is not a solution.
- Valid : All assignments are solutions.

## NP-Completeness

- Hardness : We can verify a solution in polynomial time.
- Completeness : We can reduce the halting problem to SAT.
- Also : We can reduce SAT to 3SAT.

# Difference between SAT and SMT

## SAT Instance

$$\exists p, q \in \texttt{Bool}.(p \wedge \neg q)?$$

# Difference between SAT and SMT

## SAT Instance

$$\exists p, q \in \texttt{Bool}.(p \wedge \neg q)?$$

## SAT Solution

$$[p \mapsto \top, q \mapsto \bot]$$

# Difference between SAT and SMT

## SAT Instance

$$\exists p, q \in \texttt{Bool}.(p \land \neg q)?$$

## SAT Solution

$$[p \mapsto \top, q \mapsto \bot]$$

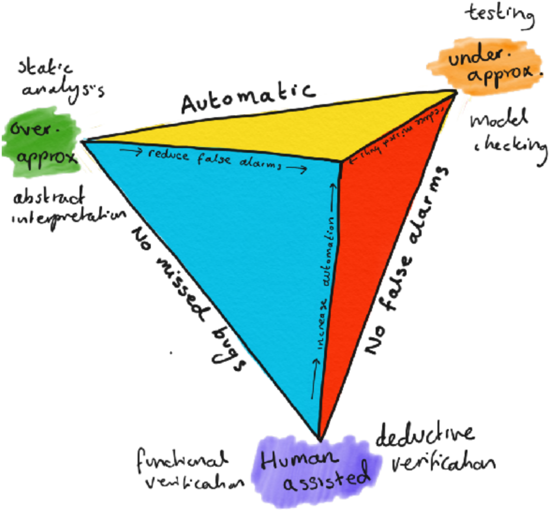
## SMT Instance

$$\exists m, n \in \mathbb{N}.(n > 0 \land m < 0)?$$

## SMT Solution

$$[m \mapsto -2, n \mapsto 3]$$

# Motivation (Verification Tools)




Triangle of Verification, Martin Brain

# Motivation (Example)

Are these programs equivalent?

```
if(!a && !b) h();
else
    if(!a) g();
    else f();
```

```
if(a) f();
else
    if(b) g();
    else h();
```

# Motivation (Example)

Are these programs equivalent?

```
if(!a && !b) h();
else
    if(!a) g();
    else f();
```

```
if(a) f();
else
    if(b) g();
    else h();
```

How about these programs?

if $\neg a \wedge \neg b$ then $h$
else
  if $\neg a$ then $g$
  else $f$

if $a$ then $f$
else
  if $b$ then $g$
  else $h$

# Motivation (Example)

Are these programs equivalent?

```
if(!a && !b) h();
else
    if(!a) g();
    else f();
```

```
if(a) f();
else
    if(b) g();
    else h();
```

How about these programs?

if $\neg a \wedge \neg b$ then $h$
else
  if $\neg a$ then $g$
  else $f$

if $a$ then $f$
else
  if $b$ then $g$
  else $h$

Is this a valid formula?

$$
\begin{array}{ll}
 & (\neg a \wedge \neg b) \wedge h \;\vee\; \neg(\neg a \wedge \neg b) \wedge (\neg a \wedge g \;\vee\; a \wedge f) \\
\iff & a \wedge f \;\vee\; \neg a \wedge (b \wedge g \;\vee\; \neg b \wedge h)\,.
\end{array}
$$

# Complexity of 3SAT

## Naive algorithm

- $F(x_1, \ldots, x_n)$ is a formula.
- $\alpha$ is an assignment.
- we check $F(\alpha')$ for all $\alpha'$.
- Worst case $O(|F| \cdot 2^n)$

# Example Heuristic - Divide an Conquer

## Algorithm (outline)

- pick a variable $x_i$ from $F$ (a formula in CNF).

# Example Heuristic - Divide an Conquer

## Algorithm (outline)

- pick a variable $x_i$ from $F$ (a formula in CNF).
- define $F(x_i = \top)$ as follows:

# Example Heuristic - Divide an Conquer

## Algorithm (outline)

- pick a variable $x_i$ from $F$ (a formula in CNF).
- define $F(x_i = \top)$ as follows:
- remove the clauses from $F$ that contain $x_i$

# Example Heuristic - Divide an Conquer

## Algorithm (outline)

- pick a variable $x_i$ from $F$ (a formula in CNF).
- define $F(x_i = \top)$ as follows:
- remove the clauses from $F$ that contain $x_i$
- if a clause contains only $\neg x_i$ then $F(x_i = \top)$ is unsatisfiable

# Example Heuristic - Divide an Conquer

## Algorithm (outline)

- pick a variable $x_i$ from $F$ (a formula in CNF).
- define $F(x_i = \top)$ as follows:
- remove the clauses from $F$ that contain $x_i$
- if a clause contains only $\neg x_i$ then $F(x_i = \top)$ is unsatisfiable
- remove other clauses containing $\neg x_i$

# Example Heuristic - Divide an Conquer

## Algorithm (outline)

- pick a variable $x_i$ from $F$ (a formula in CNF).
- define $F(x_i = \top)$ as follows:
- remove the clauses from $F$ that contain $x_i$
- if a clause contains only $\neg x_i$ then $F(x_i = \top)$ is unsatisfiable
- remove other clauses containing $\neg x_i$
- repeat until $F$ does not contain any variables

# Example Heuristic - Divide an Conquer

## Algorithm (outline)

- pick a variable $x_i$ from $F$ (a formula in CNF).
- define $F(x_i = \top)$ as follows:
- remove the clauses from $F$ that contain $x_i$
- if a clause contains only $\neg x_i$ then $F(x_i = \top)$ is unsatisfiable
- remove other clauses containing $\neg x_i$
- repeat until $F$ does not contain any variables
- analogously for $F(x_i = \bot)$

# Example Heuristic - Divide an Conquer

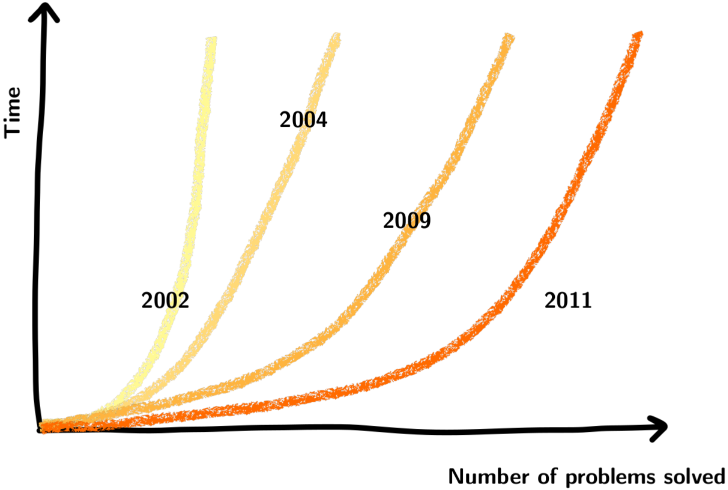
## Algorithm (outline)

- pick a variable $x_i$ from $F$ (a formula in CNF).
- define $F(x_i = \top)$ as follows:
- remove the clauses from $F$ that contain $x_i$
- if a clause contains only $\neg x_i$ then $F(x_i = \top)$ is unsatisfiable
- remove other clauses containing $\neg x_i$
- repeat until $F$ does not contain any variables
- analogously for $F(x_i = \bot)$

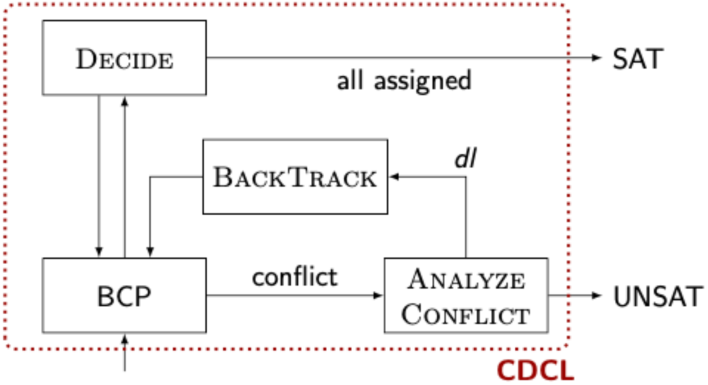
## Analysis (Outline)

- If F contains $(x_i \vee x_j)$ then we need to check $F(x_i = \top)$ and $F(x_i = \bot)(x_j = \top)$.
- Worst case $O(|F| \cdot\ 1.84^n)$. (Hromkovic - 2002).

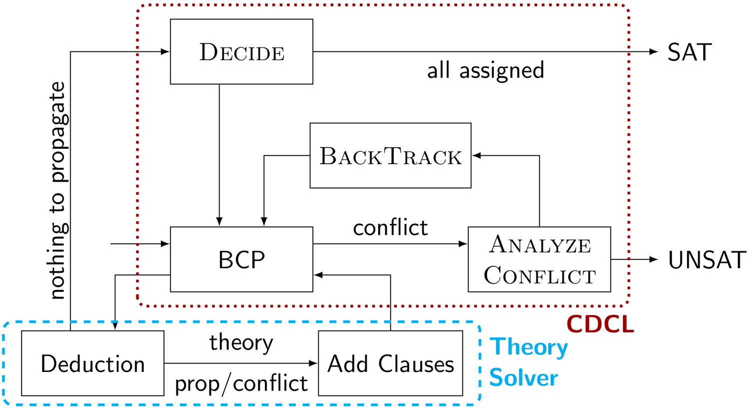
# Progress in Heuristics



# Conflict Driven Clause Learning (overview)



# Conflict Driven Clause Learning (with theories)



# Central Question. How do I use it?

## Old school

- Reduce your program to a formula.
- Write a Dimacs file.
- Run solver on said file.

## Construct instance via FFI

- Don't write Dimacs by hand.
- Import a library.

# Example (Sudoku)

## Strategy

- Sudoku is NP Complete.
- So, we reduce it to SMT and call Z3.
- We can call Z3 from Python3.

# Links from the lecure.

https://github.com/Z3Prover/z3
https://jix.github.io/varisat/manual/0.2.0/formats/dimacs.html
https://jcrouser.github.io/CSC250/projects/sudoku.html