Understanding, Exploiting and Extending SAT solvers

Laurent Simon  
Labri, Bordeaux, France

With the help of
George Katsirelos  
INRA, Toulouse, France

for the practical Session
Today’s Itinerary

Introduction

Propositional Logic: simple and complex

DP & DPLL

Applications

CDCL

Understanding CDCL as complex systems

Conclusion
Today’s Itinerary

Introduction

Propositional Logic: simple and complex

DP & DPLL

Applications

CDCL

Understanding CDCL as complex systems

Conclusion
Today’s Roadmap: learns & practice

Learn the algorithms behind modern SAT solvers

Practice with the 100% pure python SAT Solver pysat bitbucket

Play with it in python (your favorite editor) or with the Jupyter Notebooks

You have a few minutes to check your Python3 / Jupyter installation

Go to
http://www.labri.fr/perso/lsimon/toulouse2018

to get ready
Why studying SAT?

SAT, the canonical NP-Complete problem.

Related to a one-million dollar question (is NP=P?)

- The **main open problem** of Theoretical Computer Science
- **Must be faced** in most real-world problems
- But, the *easiest of the hard* problems

\[ \Leftarrow \text{Hope here!} \]

*S. Aaronson, MIT*: « *If P = NP*, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in ‘creative leaps,’ no fundamental gap between solving a problem and recognizing the solution once it’s found. **Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss. »*[^1]

[^1]: citation taken from [Vardi, 2015]
Why studying SAT? – 2

Reduction to SAT: prove the hardness of new problems
- many reductions exist from many problems to SAT
- SAT "captures" the difficulty of many other problems

SAT Solvers are (incredibly) efficient and "user-friendly"
- Can be used as a black box
- Can be used as an open box (special heuristics, ...)
- Can be extended in many ways
  - SAT solvers as Oracles on many close formulas
  - SAT solvers working at an abstract level
  - SAT solvers working on "on the fly" generated constraints
  - ...
More and more efficient SAT solving
More and more efficient SAT solving
More and more efficient SAT solving

2005
More and more efficient SAT solving
More and more efficient SAT solving

![Graph showing the number of solved problems over time for 2009.](image)
More and more efficient SAT solving

2011
More and more efficient SAT solving

![Graph showing the increase in efficiency of SAT solving over time, with 2014 highlighted.]
More and more efficient SAT solving
More and more efficient SAT solving
Today’s Itinerary

Introduction

Propositional Logic: simple and complex

DP & DPLL

Applications

CDCL

Understanding CDCL as complex systems

Conclusion
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The facts: are true, or false.

The variables: are $\top$, or $\bot$.

Reasoning by calculus

If we know that:

- $A$ implies $B$
- $B$ implies $C$

Then, we can deduce:

- $A$ implies $C$

If we have:

- $\neg A \lor B$
- $\neg B \lor C$

Then, by resolution:

- $\neg A \lor C$

Logic defined more than 2300 years ago!
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The **facts**:
- are true, or false.

The **variables**:
- are $\top$, or $\bot$.

Reasoning by calculus

If we know that:
- A implies B
- B implies C

Then, we can **deduce**:
- A implies C

If we have:
- $\neg A \lor B$
- $\neg B \lor C$

Then, by **resolution**:
- $\neg A \lor C$

Logic defined more than 2300 years ago!
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The **facts**: are true, or false.

The **variables**: are $\top$, or $\bot$.

Reasoning by calculus

If we know that:

- A implies B
- B implies C

Then, we can **deduce**:

- A implies C

If we have:

- $\neg A \lor B$
- $\neg B \lor C$

Then, by **resolution**:

- $\neg A \lor C$

Logic defined more than 2300 years ago!
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The facts: are true, or false.

The variables: are $\top$, or $\bot$.

Reasoning by calculus

If we know that:
- A implies B
- B implies C
Then, we can deduce:
- A implies C

If we have:
- $\neg A \lor B$
- $\neg B \lor C$
Then, by resolution:
- $\neg A \lor C$

Logic defined more than 2300 years ago!
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The facts: are true, or false.

The variables: are T, or ⊥.

Reasoning by calculus

If we know that:
- A implies B
- B implies C
Then, we can deduce:
- A implies C

If we have:
- ¬A ∨ B
- ¬B ∨ C
Then, by resolution:
- ¬A ∨ C

Logic defined more than 2300 years ago!
Simple is beautiful... But speed is essential

Is it really interesting to study how to implement a CDCL?

The paradigm shift is essentially due to

- **Tight data structures**
- **Algorithms built upon this data structure**

The way "Modern" SAT solvers are solving problems has nothing common with a human strategy

Interesting problems are not toy problems

Being fast is the way computers are not so dumb
What can be done with a so simple logic?

The facts are propositional variables
The knowledge is a propositional formula

\[ \neg x_1 \lor \neg x_2 \lor x_3 \]
\[ \land \neg x_3 \]
\[ \land x_1 \lor x_2 \]
\[ \land x_2 \lor x_3 \]

- Variables: \( x_1 \ldots x_3 \);
- Literals: \( x_1, \neg x_1 \);
- Clauses: \( \neg x_1 \lor \neg x_2 \lor x_3 \);
- Formula \( \Sigma \) written in CNF (conjonction of clauses);

Big questions:
- **SAT**: is there an assignment of variables making the formula true?
- **UNSAT**: is the theory contradictory?
- **PI**: deduce all you can from \( \Sigma \)
What can be done with a so simple logic?

The facts are propositional variables
The knowledge is a propositional formula

\[ \neg x_1 \lor \neg x_2 \lor x_3 \]
\[ \land \quad \neg x_3 \]
\[ \land \quad x_1 \lor x_2 \]
\[ \land \quad x_2 \lor x_3 \]

- Variables: \( x_1 \ldots x_3 \);
- Literals: \( x_1, \neg x_1 \);
- Clauses: \( \neg x_1 \lor \neg x_2 \lor x_3 \);
- Formula \( \Sigma \) written in CNF (conjunction of clauses);

Big questions

- **SAT**: is there an assignment of variables making the formula true?
- **UNSAT**: is the theory contradictory?
- **PI**: deduce all you can from \( \Sigma \)
What can be done with a so simple logic?

The facts are propositional variables
The knowledge is a propositional formula

$\neg x_1 \lor \neg x_2 \lor x_3$
$\land$
$\neg x_3$
$\land$
$x_1 \lor x_2$
$\land$
$x_2 \lor x_3$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>

Big questions

- **SAT**: is there an assignment of variables making the formula true?
- **UNSAT**: is the theory contradictory?
- **PI**: deduce all you can from $\Sigma$

Variables: $x_1 \ldots x_3$;
Literals: $x_1$, $\neg x_1$;
Clauses: $\neg x_1 \lor \neg x_2 \lor x_3$;
Formula $\Sigma$ written in CNF (conjunction of clauses);
What can be done with a so simple logic?

The facts are propositional variables
The knowledge is a propositional formula

\( \neg x_1 \lor \neg x_2 \lor x_3 \land \neg x_3 \land x_1 \lor x_2 \land x_2 \lor x_3 \)

- Variables: \( x_1 \ldots x_3 \);
- Literals: \( x_1, \neg x_1 \);
- Clauses: \( \neg x_1 \lor \neg x_2 \lor x_3 \);
- Formula \( \Sigma \) written in CNF (conjunction of clauses);

Big questions
- SAT: is there an assignment of variables making the formula true?
- UNSAT: is the theory contradictory?
- PI: deduce all you can from \( \Sigma \)
What can be done with a so simple logic?

The facts are propositional variables
The knowledge is a propositional formula

\[
\neg x_1 \lor \neg x_2 \lor x_3 \\
\land x_1 \lor \neg x_3 \\
\land x_1 \lor x_2 \\
\land x_2 \lor x_3
\]

- Variables: \(x_1 \ldots x_3\);
- Literals: \(x_1, \neg x_1\);
- Clauses: \(\neg x_1 \lor \neg x_2 \lor x_3\);
- Formula \(\Sigma\) written in CNF (conjunction of clauses);

Big questions
- **SAT**: is there an assignment of variables making the formula true?
- **UNSAT**: is the theory contradictory?
- **PI**: deduce all you can from \(\Sigma\)
The SAT problem

**Definition**

**Input:** A set of clauses built from a propositional language with $n$ variables.

**Output:** Is there an assignment of the $n$ variables that satisfies all those clauses?

**Example** (with different notations)

\[ \Sigma_1 = (\neg a \lor b) \land (\neg b \lor c) = (a' + b).(b' + c) = \{ \neg a \lor b, \neg b \lor c \} \]

\[ \Sigma_2 = \Sigma_1 \land a \land \neg c = \Sigma_1 \cup \{ a, \neg c \} \]

For $\Sigma_1$, the answer is yes, for $\Sigma_2$ the answer is no

\[ \Sigma_1 \models \neg a \lor c = \neg(a \land \neg c) \]
Why working on CNF? Bec(l)ause!

Why considering SAT only CNF only?

\[(SAT \text{ is trivial on DNF!})\]

– Human-designed systems are conjunctions of properties (in general)
– We want to do symbolic reasoning (not trying all possible solutions)
– There are (too) many ways of applying rules at each step
– We need to restrict the possibilities at each step

But, what about re-writing any formula into CNF?

• Very hard in general
• But, very easy \emph{if we just want to check SAT}
Working on CNF [Tseitin 1968]

Idea: introduce new variables encoding the satifiability of subformulas

Let’s say we need to check \( SAT(f) \) with

\[
f \equiv g \lor h
\]

We introduce \( x_f, x_g \) and \( x_h \) representing the satifiability of \( f, g \) and \( h \), respectively

\[
(\neg x_f \lor x_g \lor x_h) \land (x_f \lor \neg x_g) \land (x_f \lor \neg x_h)
\]

\( x_f \) encodes the satisfiability of \( f \). Easy! (linear, no blow-up!)

(Introducing new variables is so powerful, isn’t it?)
It is like naming all the wires in a circuit

\[ f = (x_1 \land x_2) \lor ((x_3 \land x_4) \oplus (x_5 \land x_6)) \]

Adds \( y_1, y_2, y_3, y_4, y_f \) and:

\[
\Sigma_f \equiv \left( \begin{array}{l}
(y_f \leftrightarrow y_3 \land y_4) \\
\land (y_4 \leftrightarrow y_1 \oplus y_2) \\
\land (y_1 \leftrightarrow x_3 \land x_4) \\
\land (y_2 \leftrightarrow x_5 \land x_6) \\
\land (y_f)
\end{array} \right)
\]

\( f \) is satisfiable iff \( \Sigma_f \) is
How can we solve it?

- Try all possible solutions?
- Try to guess a solution?
- Any other idea?

In this Master Class we will not consider local search (we can prove UNSAT!)
At the heart of most procedures: resolution

**The Resolution Rule (Cut)** [Gentzen 1934, Robinson 1965]

Let $c_1 = (x \lor a_1 \lor \ldots a_n)$ and $c_2 = (\neg x \lor b_1 \lor \ldots b_m)$

\[
c = (a_1 \lor \ldots a_n \lor b_1 \lor \ldots b_m)
\]

is obtained by resolution on $x$ between $c_1$ and $c_2$

It is a particular case of the following **deduction rule**:

if $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$

In general, SAT solvers are only using this rule
(but many, many times per second)

Finding which ones to trigger is the secret of efficient SAT solvers
Today’s Itinerary

Introduction

Propositional Logic: simple and complex

DP & DPLL

Applications

CDCL

Understanding CDCL as complex systems

Conclusion
Davis & Putnam: the firsts SAT steps

1958: Hilary Putnam and Martin Davis look for funding their research around propositional logic

« What we’re interested in is good algorithms for propositional calculus » (NSA)

Before that, only inefficient methods (truth tables, ...)

First papers

- *Computational Methods in The Propositional calculus* [Davis Putnam 1958]
- *A Computing Procedure for Quantification Theory* [Davis Putnam 1960]

2Rapport interne NSA
1960, already a first (kind of) competition!

« The superiority of the present procedure (i.e. DP) over those previously available is indicated in part by the fact that a formula on which Gilmore’s routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes »

[Davis et Putnam 1960], page 202.
Principles of DP-60

DP-60: forgets variables one after the other
Example: forgets $x_1$.

\[ x_1 \lor x_4 \]
\[ \overline{x_1} \lor x_4 \lor x_{14} \]
\[ \overline{x_1} \lor x_3 \lor x_8 \]
\[ x_1 \lor x_8 \lor x_{12} \]
\[ x_1 \lor x_5 \lor \overline{x_9} \]
\[ x_2 \lor x_{11} \]
\[ \overline{x_3} \lor x_7 \lor x_{13} \]
\[ \overline{x_3} \lor \overline{x_7} \lor x_{13} \lor x_9 \]
\[ x_8 \lor \overline{x_7} \lor \overline{x_9} \]
Principles of DP-60

DP-60: forgets variables one after the other
Example: forgets \( x_1 \).

\[
\begin{align*}
  x_1 & \lor x_4 \\
  x_1 & \lor x_8 \lor x_{12} \\
  x_1 & \lor x_5 \lor \overline{x_9} \\
  \overline{x_1} & \lor x_4 \lor x_{14} \\
  \overline{x_1} & \lor \overline{x_3} \lor \overline{x_8} \\
  x_2 & \lor x_{11} \\
  \overline{x_3} & \lor \overline{x_7} \lor x_{13} \\
  \overline{x_3} & \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \\
  x_8 & \lor \overline{x_7} \lor \overline{x_9}
\end{align*}
\]
Principles of DP-60

DP-60: forgets variables one after the other
Example: forgets $x_1$.

$$x_1 \lor \left( \begin{array}{c}
  x_4 \\
  x_8 \lor x_{12} \\
  x_5 \lor \overline{x}_9
\end{array} \right)$$

$$\overline{x}_1 \lor \left( \begin{array}{c}
  x_4 \lor \overline{x}_{14} \\
  \overline{x}_3 \lor \overline{x}_8
\end{array} \right)$$

$$x_2 \lor x_{11}$$
$$\overline{x}_3 \lor \overline{x}_7 \lor x_{13}$$
$$\overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9$$
$$x_8 \lor \overline{x}_7 \lor \overline{x}_9$$
Principles of DP-60

DP-60: forgets variables one after the other
Example: forgets $x_1$.

\[
\left( \begin{array}{c}
  x_4 \\
  x_8 \lor x_{12} \\
  x_5 \lor \overline{x_9}
\end{array} \right) \lor \left( \begin{array}{c}
  x_4 \lor x_{14} \\
  \overline{x_3} \lor \overline{x_8}
\end{array} \right)
\]

$x_2 \lor x_{11}$
$\overline{x_3} \lor \overline{x_7} \lor x_{13}$
$\overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9$
$x_8 \lor \overline{x_7} \lor \overline{x_9}$
Principles of DP-60

DP-60: forgets variables one after the other
Example: forgets $x_1$.

\[
\begin{align*}
x_4 & \lor x_{14} \\
x_4 & \lor \overline{x_3} \lor \overline{x_8} \\
x_8 & \lor \overline{x_{12}} \lor x_4 \lor x_{14} \\
x_5 & \lor \overline{x_9} \lor x_4 \lor x_{14} \\
x_5 & \lor \overline{x_9} \lor x_3 \lor \overline{x_8} \\
x_2 & \lor x_{11} \\
\overline{x_3} & \lor \overline{x_7} \lor x_{13} \\
\overline{x_3} & \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \\
x_8 & \lor \overline{x_7} \lor \overline{x_9}
\end{align*}
\]
Principles of DP-60

DP-60: forgets variables one after the other
Example: forgets $x_1$.

\begin{align*}
x_4 & \lor x_{14} \\
x_4 & \lor \overline{x_3} \lor \overline{x_8} \\
x_8 & \lor x_{12} \lor x_4 \lor x_{14} \\
x_5 & \lor \overline{x_9} \lor x_4 \lor x_{14} \\
x_5 & \lor \overline{x_9} \lor \overline{x_3} \lor \overline{x_8} \\
x_2 & \lor x_{11} \\
\overline{x_3} & \lor \overline{x_7} \lor x_{13} \\
\overline{x_3} & \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \\
x_8 & \lor \overline{x_7} \lor \overline{x_9}
\end{align*}
Variable elimination – on a trivial example

\[ (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \]

\[ (\neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \]

\[ (x_3 \lor \neg x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \]

\[ (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \]

\[ x_3 \models \]

- Formula is SAT (and \( x_3 \) is True in all models)
Variable elimination – on a trivial example

\((x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models\)

\((\neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models\)

\((x_3 \lor \neg x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models\)

\((x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models\)

\(x_3 \models\)

Formula is SAT (and \(x_3\) is True in all models)
Variable elimination – on a trivial example

\[ (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \]

\[ (\neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \]

\[ (x_3 \lor \neg x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \]

\[ (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \]

\[ x_3 \models \]

Formula is SAT (and \( x_3 \) is True in all models)
Variable elimination – on a trivial example

\[(x_1 \lor \lnot x_2 \lor \lnot x_3) \land (\lnot x_1 \lor \lnot x_2 \lor \lnot x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \lnot x_4) \models\]

\[(\lnot x_2 \lor \lnot x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \lnot x_4) \models\]

\[(x_3 \lor \lnot x_3) \land (x_3 \lor x_4) \land (x_3 \lor \lnot x_4) \models\]

\[(x_3 \lor x_4) \land (x_3 \lor \lnot x_4) \models\]

\[x_3 \models\]

\[\top\]

- Formula is SAT (and \(x_3\) is True in all models)
Variable elimination – on a trivial example

\[(x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \equiv \]

\[(\neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \equiv \]

\[(x_3 \lor \neg x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \equiv \]

\[(x_3 \lor x_4) \land (x_3 \lor \neg x_4) \equiv \]

\[x_3 \equiv \]

\[\top \]

- Formula is SAT (and \(x_3\) is True in all models)
Variable elimination – on a trivial example

\[
(\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \\
(\neg x_2 \lor \neg x_3) \land (x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \\
(x_3 \lor \neg x_3) \land (x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \\
(x_3 \lor x_4) \land (x_3 \lor \neg x_4) \models \\
\neg x_3 \models \\
\top
\]

- Formula is SAT (and \( x_3 \) is True in all models)
Untractable Space Problems

Combinatorial explosion, even on very small problems!
But possible on some very special cases (SAT pre-processing)
Untractable Space Problems

Combinatorial explosion, even on very small problems!

But possible on some very special cases (SAT pre-processing)
1962-2001 : DPLL rules the world

Systematically explore the space of partial models (backtrack)

- Choose a literal
- Try to find a solution with this literal set to True
- If it is not possible:
  Finds a solution with this literal set to False

Backtrack search on partial models
Systematic (ordered) exploration ensures completeness
1962-2001: DPLL rules the world

Systematically explore the space of partial models (backtrack)

- Choose a literal
- Try to find a solution with this literal set to True
- If it is not possible:
  Finds a solution with this literal set to False

Backtrack search on partial models
Systematic (ordered) exploration ensures completeness
Backtrack search

- How to choose the right literal to branch on?
- First search for a model or a contradiction?
Backtrack search

- How to choose the right literal to branch on?
- First search for a model or a contradiction?
An example of DPLL

<table>
<thead>
<tr>
<th>Formula</th>
<th>Simplified Formula</th>
<th>Partial Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 \lor x_4$</td>
<td>$x_1 \lor x_4$</td>
<td></td>
</tr>
<tr>
<td>$\overline{x_1} \lor x_4 \lor x_{14}$</td>
<td>$\overline{x_1} \lor x_4 \lor x_{14}$</td>
<td></td>
</tr>
<tr>
<td>$x_1 \lor x_3 \lor \overline{x_8}$</td>
<td>$x_1 \lor x_3 \lor \overline{x_8}$</td>
<td></td>
</tr>
<tr>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td></td>
</tr>
<tr>
<td>$x_2 \lor x_{12}$</td>
<td>$x_2 \lor x_{12}$</td>
<td></td>
</tr>
<tr>
<td>$\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$</td>
<td>$\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$</td>
<td></td>
</tr>
<tr>
<td>$\overline{x_3} \lor x_7 \lor \overline{x_{13}}$</td>
<td>$\overline{x_3} \lor x_7 \lor \overline{x_{13}}$</td>
<td></td>
</tr>
<tr>
<td>$x_8 \lor \overline{x_7} \lor \overline{x_{12}}$</td>
<td>$x_8 \lor \overline{x_7} \lor \overline{x_{12}}$</td>
<td></td>
</tr>
</tbody>
</table>

$x_1$ appears in 4 clauses and 1 binary clause
An example of DPLL

**Formula**

- $x_1 \lor x_4$
- $\overline{x_1} \lor x_4 \lor x_{14}$
- $x_1 \lor \overline{x_3} \lor \overline{x_8}$
- $x_1 \lor x_8 \lor x_{12}$
- $x_2 \lor x_{12}$
- $\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$
- $\overline{x_3} \lor x_7 \lor \overline{x_{13}}$
- $x_8 \lor \overline{x_7} \lor \overline{x_{12}}$

**Simplified Formula**

- $x_1 \lor x_4$
- $\overline{x_1} \lor x_4 \lor x_{14}$
- $x_1 \lor \overline{x_3} \lor \overline{x_8}$
- $x_1 \lor x_8 \lor x_{12}$
- $x_2 \lor x_{12}$
- $\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$
- $\overline{x_3} \lor x_7 \lor \overline{x_{13}}$
- $x_8 \lor \overline{x_7} \lor \overline{x_{12}}$

**Partial Model**

| Lev. Lit. Back? | 1 | $\overline{x_1}$ (d) |

$x_4$ appears in 1 unary clause
An example of DPLL

Formula

\[
\begin{align*}
  x_1 & \lor x_4 \\
  \overline{x_1} & \lor x_4 \lor x_{14} \\
  x_1 & \lor \overline{x_3} \lor \overline{x_8} \\
  x_1 & \lor x_8 \lor x_{12} \\
  x_2 & \lor x_{12} \\
  \overline{x_3} & \lor \overline{x_{12}} \lor x_{13} \\
  \overline{x_3} & \lor x_7 \lor \overline{x_{13}} \\
  x_8 & \lor \overline{x_7} \lor \overline{x_{12}}
\end{align*}
\]

Simplified Formula

\[
\begin{align*}
  x_1 & \lor x_4 \\
  \overline{x_1} & \lor x_4 \lor x_{14} \\
  x_1 & \lor \overline{x_3} \lor \overline{x_8} \\
  x_1 & \lor x_8 \lor x_{12} \\
  x_2 & \lor x_{12} \\
  \overline{x_3} & \lor \overline{x_{12}} \lor x_{13} \\
  \overline{x_3} & \lor x_7 \lor \overline{x_{13}} \\
  x_8 & \lor \overline{x_7} \lor \overline{x_{12}}
\end{align*}
\]

Partial Model

\[
\begin{align*}
  \text{Lev. Lit. Back?} \\
  1 & \overline{x_1} \quad (d) \\
  + & \quad x_4
\end{align*}
\]

\(x_3\) appears in 3 clauses incl. 1 (new) binary clause
An example of DPLL

**Formula**

- $x_1 \lor x_4$
- $\overline{x}_1 \lor x_4 \lor x_{14}$
- $x_1 \lor \overline{x}_3 \lor \overline{x}_8$
- $x_1 \lor x_8 \lor x_{12}$
- $x_2 \lor x_{12}$
- $\overline{x}_3 \lor \overline{x}_{12} \lor x_{13}$
- $\overline{x}_3 \lor x_7 \lor \overline{x}_{13}$
- $x_8 \lor \overline{x}_7 \lor \overline{x}_{12}$

**Simplified Formula**

- $x_1 \lor x_4$
- $\overline{x}_1 \lor x_4 \lor x_{14}$
- $x_1 \lor \overline{x}_3 \lor \overline{x}_8$
- $x_1 \lor x_8 \lor x_{12}$
- $x_2 \lor x_{12}$
- $\overline{x}_3 \lor \overline{x}_{12} \lor x_{13}$
- $\overline{x}_3 \lor x_7 \lor \overline{x}_{13}$
- $x_8 \lor \overline{x}_7 \lor \overline{x}_{12}$

**Partial Model**

<table>
<thead>
<tr>
<th>Lev. Lit. Back?</th>
<th>1</th>
<th>$\overline{x}_1$ (d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>2</td>
<td>$x_4$</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$x_3$ (d)</td>
</tr>
</tbody>
</table>

$\overline{x}_8$ appears in one unary clause
An example of DPLL

<table>
<thead>
<tr>
<th>Formula</th>
<th>Simplified Formula</th>
<th>Partial Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 \lor x_4$</td>
<td>$x_1 \lor x_4$</td>
<td>(1) $\overline{x_1}$ (d)</td>
</tr>
<tr>
<td>$\overline{x_1} \lor x_4 \lor x_{14}$</td>
<td>$\overline{x_1} \lor x_4 \lor x_{14}$</td>
<td>+ $x_4$</td>
</tr>
<tr>
<td>$x_1 \lor \overline{x_3} \lor \overline{x_8}$</td>
<td>$x_1 \lor \overline{x_3} \lor \overline{x_8}$</td>
<td>2 $x_3$ (d)</td>
</tr>
<tr>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>+ $\overline{x_8}$</td>
</tr>
<tr>
<td>$x_2 \lor x_{12}$</td>
<td>$x_2 \lor x_{12}$</td>
<td>(1) $\overline{x_1}$ (d)</td>
</tr>
<tr>
<td>$\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$</td>
<td>$\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$</td>
<td>+ $x_4$</td>
</tr>
<tr>
<td>$\overline{x_3} \lor x_7 \lor \overline{x_{13}}$</td>
<td>$\overline{x_3} \lor x_7 \lor \overline{x_{13}}$</td>
<td>2 $x_3$ (d)</td>
</tr>
<tr>
<td>$x_8 \lor \overline{x_7} \lor \overline{x_{12}}$</td>
<td>$x_8 \lor \overline{x_7} \lor \overline{x_{12}}$</td>
<td>+ $\overline{x_8}$</td>
</tr>
</tbody>
</table>

$x_{12}$ appears in 1 unary clause
An example of DPLL

Formula

\[ x_1 \lor x_4 \]
\[ \overline{x_1} \lor x_4 \lor x_{14} \]
\[ x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ x_1 \lor x_8 \lor x_{12} \]
\[ x_2 \lor x_{12} \]
\[ \overline{x_3} \lor \overline{x_{12}} \lor x_{13} \]
\[ \overline{x_3} \lor x_7 \lor \overline{x_{13}} \]
\[ x_8 \lor \overline{x_7} \lor \overline{x_{12}} \]

Simplified Formula

\[ x_1 \lor x_4 \]
\[ \overline{x_1} \lor x_4 \lor x_{14} \]
\[ x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ x_1 \lor x_8 \lor x_{12} \]
\[ x_2 \lor x_{12} \]
\[ \overline{x_3} \lor \overline{x_{12}} \lor x_{13} \]
\[ \overline{x_3} \lor x_7 \lor \overline{x_{13}} \]
\[ x_8 \lor \overline{x_7} \lor \overline{x_{12}} \]

Partial Model

<table>
<thead>
<tr>
<th>Lev.</th>
<th>Lit.</th>
<th>Back?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \overline{x}_1 )</td>
<td>(d)</td>
</tr>
<tr>
<td>+</td>
<td>( x_4 )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( x_3 )</td>
<td>(d)</td>
</tr>
<tr>
<td>+</td>
<td>( \overline{x}_8 )</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>( x_{12} )</td>
<td></td>
</tr>
</tbody>
</table>

\( x_{13}, \overline{x}_7 \) appear in unary clauses
An example of DPLL

<table>
<thead>
<tr>
<th>Formula</th>
<th>Simplified Formula</th>
<th>Partial Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 \lor x_4$</td>
<td>$x_1 \lor x_4$</td>
<td>Lev. Lit. Back?</td>
</tr>
<tr>
<td>$\overline{x}<em>1 \lor x_4 \lor x</em>{14}$</td>
<td>$\overline{x}<em>1 \lor x_4 \lor x</em>{14}$</td>
<td>1 \ $\overline{x}_1$ (d)</td>
</tr>
<tr>
<td>$x_1 \lor \overline{x}_3 \lor \overline{x}_8$</td>
<td>$x_1 \lor \overline{x}_3 \lor \overline{x}_8$</td>
<td>+ \ $x_4$</td>
</tr>
<tr>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>2 \ $x_3$ (d)</td>
</tr>
<tr>
<td>$x_2 \lor x_{12}$</td>
<td>$x_2 \lor x_{12}$</td>
<td>+ \ $\overline{x}_8$</td>
</tr>
<tr>
<td>$\overline{x}<em>3 \lor \overline{x}</em>{12} \lor x_{13}$</td>
<td>$\overline{x}<em>3 \lor \overline{x}</em>{12} \lor x_{13}$</td>
<td>+ \ $x_{12}$</td>
</tr>
<tr>
<td>$\overline{x}<em>3 \lor x_7 \lor \overline{x}</em>{13}$</td>
<td>$\overline{x}<em>3 \lor x_7 \lor \overline{x}</em>{13}$</td>
<td>+ \ $x_{13}$</td>
</tr>
<tr>
<td>$x_8 \lor \overline{x}<em>7 \lor \overline{x}</em>{12}$</td>
<td>$x_8 \lor \overline{x}<em>7 \lor \overline{x}</em>{12}$</td>
<td></td>
</tr>
</tbody>
</table>

$x_7$, $\overline{x}_7$ appear in unary clauses
An example of DPLL

**Formula**

\[
\begin{align*}
& x_1 \lor x_4 \\
& \overline{x}_1 \lor x_4 \lor x_{14} \\
& x_1 \lor \overline{x}_3 \lor \overline{x}_8 \\
& x_1 \lor x_8 \lor x_{12} \\
& x_2 \lor x_{12} \\
& \overline{x}_3 \lor \overline{x}_{12} \lor x_{13} \\
& \overline{x}_3 \lor x_7 \lor \overline{x}_{13} \\
& x_8 \lor \overline{x}_7 \lor \overline{x}_{12}
\end{align*}
\]

**Simplified Formula**

\[
\begin{align*}
& x_1 \lor x_4 \\
& \overline{x}_1 \lor x_4 \lor x_{14} \\
& x_1 \lor \overline{x}_3 \lor \overline{x}_8 \\
& x_1 \lor x_8 \lor x_{12} \\
& x_2 \lor x_{12} \\
& \overline{x}_3 \lor \overline{x}_{12} \lor x_{13} \\
& \overline{x}_3 \lor x_7 \lor \overline{x}_{13} \\
& x_8 \lor \overline{x}_7 \lor \overline{x}_{12}
\end{align*}
\]

**Partial Model**

<table>
<thead>
<tr>
<th>Lev.</th>
<th>Lit.</th>
<th>Back?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\overline{x}_1$</td>
<td>(d)</td>
</tr>
<tr>
<td>2</td>
<td>$x_3$</td>
<td>(d)</td>
</tr>
</tbody>
</table>

Conflicting! Undo everything until last decision.
An example of DPLL

**Formula**

\[ \neg x_1 \lor x_4 \]

\[ x_1 \lor x_4 \lor x_{14} \]

\[ x_1 \lor \neg x_3 \lor \neg x_8 \]

\[ x_1 \lor x_8 \lor x_{12} \]

\[ x_2 \lor x_{12} \]

\[ \neg x_3 \lor \neg x_{12} \lor x_{13} \]

\[ \neg x_3 \lor x_7 \lor x_{13} \]

\[ x_8 \lor \neg x_7 \lor \neg x_{12} \]

**Simplified Formula**

\[ x_1 \lor x_4 \]

\[ x_1 \lor \neg x_4 \lor x_{14} \]

\[ x_1 \lor x_3 \lor \neg x_8 \]

\[ x_1 \lor x_8 \lor x_{12} \]

\[ x_2 \lor x_{12} \]

\[ \neg x_3 \lor \neg x_{12} \lor x_{13} \]

\[ \neg x_3 \lor x_7 \lor \neg x_{13} \]

\[ x_8 \lor \neg x_7 \lor \neg x_{12} \]

**Partial Model**

\[ \text{Lev. Lit. Back?} \]

\[ 1 \quad \neg x_1 \quad (d) \]

\[ + \quad x_4 \]

\[ * \quad \neg x_3 \]

*Now, \( \neg x_3 \) is not a decision*
A very simple procedure?

Very simple to write a backtrack search but...

Where to branch?

- Mistakes at the top of the tree are dramatic!
- (almost) As many nodes where to decide than where to explore
- A perfect branching scheme is NP-Hard
Today’s Itinerary

Introduction

Propositional Logic: simple and complex

DP & DPLL

Applications

CDCL

Understanding CDCL as complex systems

Conclusion
Bounded Model Checking at a glance

We have a system to verify, modeled by an automaton, encoding its state transitions

**Correctness: No bugs** A special state ”error” is used in the model. The problem is about its reachability.

**Liveness: No infinite loop** Any state must be reachable from any other state, in any future.

**Notice:**
- Closely related to temporal logic;
- Before SAT, BDD were used to solve these problems
(Bounded) Model Checking at a glance

We fix a *bound* $k$, and increment it as we need.

- The automaton is represented by the propositional logic function $T$ that encodes the characteristics function of the reachable states.

Example (2-bits 1-adder):

$$(a' \iff \neg a) \land (b' \iff a \oplus b)$$

$$
\begin{array}{c}
\text{a} \\
\downarrow \\
+1 \\
\uparrow \\
\text{b} \\
\end{array}
\quad
\begin{array}{c}
\text{a'} \\
\downarrow \\
\text{b'} \\
\uparrow \\
\end{array}
$$

$$(0, 0) \rightarrow (1, 0) \rightarrow (0, 1) \rightarrow (1, 1) \rightarrow (0, 0) \rightarrow \ldots$$
We fix a *bound* $k$, and increment it as we need

- The automaton is represented by the propositional logic function $T$ that encodes the characteristics function of the reachable states. Example (2-bits 1-adder): $(a' \iff \neg a) \land (b' \iff a \oplus b)$
- The property to check is (for instance): $a \land b$ (is the state $(11)$ reachable?)
- The initial state is an assignment of variables at time step 0
BMC: Unrolling loops

Let us check whether the state (11) is reachable in 2 iterations

\[ l(s_0) = \neg a_0 \land \neg b_0 \]
\[ T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0) \]
\[ T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1) \]
\[ p(s_2) = a_2 \land b_2 \]
\[ p(s_0) = a_0 \land b_0 \]
\[ p(s_1) = a_1 \land b_1 \]

Finally, is the formula

\[ (\neg a_0 \land \neg b_0) \land ((a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0)) \land ((a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1)) \land (a_2 \land b_2) \] satisfiable?
BMC: Unrolling loops

Let us check whether the state (11) is reachable in 2 iterations

\[ I(s_0) = \neg a_0 \land \neg b_0 \]

\[ T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0) \]

\[ T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1) \]

\[ p(s_2) = a_2 \land b_2 \]

\[ p(s_0) = a_0 \land b_0 \]

\[ p(s_1) = a_1 \land b_1 \]

**Finally, is the formula**

\[ (\neg a_0 \land \neg b_0) \land ((a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0)) \land ((a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1)) \land (a_2 \land b_2) \] satisfiable?
Let us check whether the state (11) is reachable in 2 iterations

\[ l(s_0) = \neg a_0 \land \neg b_0 \]

\[ T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0) \]

\[ T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1) \]

\[ p(s_2) = a_2 \land b_2 \]

\[ p(s_0) = a_0 \land b_0 \]

\[ p(s_1) = a_1 \land b_1 \]

**Finally**, is the formula

\[ (\neg a_0 \land \neg b_0) \land ((a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0)) \land ((a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1)) \land (a_2 \land b_2) \text{ satisfiable?} \]
Unbounded Model Checking

\[ I \land T_1 \land T_2 \land \ldots \land T_k \land BUG_k \]

How to ensure that \( BUG \) is unreachable?

**Idea**: find an invariant \( Inv \) s.t. \( BUG \) is not reachable in \( k > 0 \) steps

- \( Inv \) characterizes an over approximation of the reachable states in \( j \) steps:

\[ I \land T_1 \land \cdots \land T_j \rightarrow Inv \]

- \( Inv \) is an inductive property:

\[ Inv \land T_1 \rightarrow Inv_1 \]

- \( BUG \) is not reachable from \( Inv \) in \( k \) steps:

\[ Inv \land T_1 \land T_2 \land \ldots \land T_k \land BUG_k \equiv \bot \]

- Incremental SAT Solving / Proof Analysis
The Erdös Discrepancy Problem (1932)

- Infinite series of +1 and -1: $< -1, 1, 1, -1 - 1, 1, 1, \ldots >$
- $\forall C \exists k, d \text{ t.q.} \left| \sum_{i=1}^{k} x_i d \right| \geq C$

\[ + - - + + - - - + + + -1 \]
\[ - + - - + -1 \]
\[ - - + -1 \]
\[ + - 0 \]
\[ + + +2 \]

- Proven in 2014 for $C = 2$ ($k=1161$)
- The proof: UNSAT certificate (trace) from Glucose (13 Gb)$^3$
- General case proven two years later by Terence Tao (previous proof considered as the biggest mathematical proof ever by T. Tao).

---

$^3$cgi.csc.liv.ac.uk/~konev/SAT14/
Solution for $C=2$, 1160 steps

(For $C=3$, maximum solution is not yet known)
The "biggest proof" in the world

**Boolean Pythagorean triples problem**

Is it possible to colorize the \( n \) integers \( \leq n \) in two colors s.t. no triplet \((a, b, c)\) is \( a^2 + b^2 = c^2 \) monochromatic?
The "biggest proof" in the world

No solution for n=7825
- Open question since 20 years
- $10^{2300}$ possible candidates
- SAT encoding
- Original problem splitted in 1,000,000 subproblems
- 800 CPUs

Proof is 200Tb long (Glucose’s output)
- In practice the proof is not really kept
Let’s generate this 200Tb proof! (or not)

Follow the PythagoreanTriplets Python Notebook!
Today’s Itinerary

Introduction

Propositional Logic: simple and complex

DP & DPLL

Applications

CDCL

Understanding CDCL as complex systems

Conclusion
From LookAhead to Lookback

All solvers are now turned to lazily detect Unit Propagation

No way to maintain counters for “smart” branching

Look ahead heuristics were “easy” to understand

Look back heuristics are very hard to study
1999, on the way to the revolution

Huge problems are coming from the real-world: Planning & Bounded Model Checking

- Planning as Satisfiability. [Kautz and Selman, 92]
- Symbolic Model Checking using SAT procedures instead of BDDs. [Biere & al. 99]
- SAT solvers can’t cope with those huge formulas without specialized data structures Quasi-incapacité des solveurs SAT à gérer autant de clauses

DPLL extinction...

- GRASP: learning clauses in SAT solvers
- DLIS: very simple heuristic
- SATO: lazy data structure to detect unary clauses

Algorithms ingredients for the upcoming revolution
BMC, GRASP, DLIS, SATO
Huge problems?

Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf
Huge problems?

Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf
Huge problems?
Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf

BMC instances
Huge problems?
Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf

BMC instances
Huge problems?
Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf
Data structures are stronger than algorithms!

Until now:

- Heuristics were used to *mimic* what a human will do (when picking a variable to branch on)
- Data structures were implemented to support algorithms

**Seminal paper**

« *Chaff: Engineering an Efficient SAT Solver* » [Moskewicz & al. ’01]

“Simply” optimize known algorithms

**Highest priority**: BCP (Boolean Constraint Propagation)
Let’s play…to be the laziest

Detect when you have only 1 non called card in your hand

1. count the non called cards

2  ♠️

King ♠️

5 ♦️

3 ♣️

5 ♥️

Jack ♥️

6
Let’s play... to be the laziest

Detect when you have only 1 non called card in your hand

1. **count the non called cards**
Let’s play… to be the laziest

Detect when you have only 1 non called card in your hand

1. **count the non called cards**

```plaintext
2  

K

3

5

J
```

5
Let’s play...to be the laziest

Detect when you have only 1 non called card in your hand

1. **count the non called cards**

![Count the non called cards image]

4

![Count the non called cards image 2]

4
Let’s play...to be the laziest

Detect when you have only 1 non called card in your hand

1. count the non called cards

![Playing cards](image)
Let’s play…to be the laziest

Detect when you have only 1 non called card in your hand

1. count the non called cards

2️⃣ ♠️

3️⃣ ♣️

3️⃣ ♠️

3️⃣ ♠️

5️⃣ ♠️

6️⃣ ♠️

5️⃣ ♠️

10️⃣ ♠️
Let’s play...to be the laziest

Detect when you have only 1 non called card in your hand

1. **count the non called cards**
Let’s play… to be the laziest

Detect when you have only 1 non called card in your hand

1. count the non called cards
Let’s play…to be the laziest

Detect when you have only 1 non called card in your hand

1. count the non called cards

Unary Clause!!
Let's play...to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails
Let’s play…to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails
Let’s play... to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails

![Card images]
Let’s play…to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails
Let’s play…to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails

![Playing cards to illustrate the concept of detecting one non-called card in a hand.](image-url)
Let’s play...to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails

![Playing cards](image-url)
Let’s play…to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails
Let’s play... to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails
Let’s play…to be the laziest – 2

Detect when you have only 1 non called card in your hand

1. Watch heads and tails

Unary Clause!!
Let’s play…to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play…to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. **Keep only two uncalled cards (witnesses) (Chaff)**
Let’s play... to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play...to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. **Keep only two uncalled cards (witnesses) (Chaff)**

![Playing cards](image)
Let’s play...to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play…to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play...to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play...to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play…to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play... to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play…to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. Keep only two uncalled cards (witnesses) (Chaff)
Let’s play…to be the laziest – 3

Detect when you have only 1 non called card in your hand

3. **Keep only two uncalled cards (witnesses) (Chaff)**

Nothing to do if want to replay!!
Only 2 watches per clauses
How to update it?

Let’s say the current assignment is $a, b, c, d$

We branch on $e$

Clauses that can be unary/empty are all the clauses watched by $\neg e$

$\neg e \lor a \lor f \lor g$ 
Watched by $\neg e, a$

$\neg e \lor f \lor b$ 
Watched by $\neg e, f$

$f \lor \neg e \lor \neg a \lor \neg g$ 
Watched by $\neg e, f$

$\neg e \lor f \lor \neg a \lor \neg d$ 
Watched by $\neg e, f$

Example of a clause containing $\neg e$ but not watched by it:
$f \lor \neg e$
After deciding on $e$

(current assignment is $a, b, c, d$)

$\neg e \lor a \lor f \lor g$

$\neg e, a$

Watched by $\neg e, a$

$b \lor f \lor \neg e$

Watched by $\neg e, f$

$f \lor \neg e \lor \neg a \lor \neg g$

Watched by $\neg e, f$

$\neg e \lor f \lor \neg a \lor \neg d$

Watched by $\neg e, f$

Deciding on $e$:

$\neg e \lor a \lor f \lor g$

Still watched by $\neg e, a$

$\neg e, a$

Watched by $b, f$

$b \lor f \lor \neg e$

Watched by $\neg g, f$

$f \lor \neg g \lor \neg a \lor \neg e$

Watched by $\neg e, f$... But **UNARY !!**

$\neg e \lor f \lor \neg a \lor \neg d$

Watched by $\neg e, f$... But **UNARY !!**
We have to cope with this lazy data structure

The state of the formula is unknown!

- How many reduced clauses? How many satisfied clauses?
- Some variables may be pure?

Only one guarantee: all unary and empty clauses are detected

Find a model? All the variables are assigned without any conflict

Heuristique: How to choose a variable to branch on? We are blind!
- We need to use the past, not the current state of the formula
- The heuristics will be heavily related to the clause learning mechanism
Ingredients of an efficient SAT solver
All ingredients are designed to cope with huge problems
CDCL principles at a glance
Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x}_3 \lor \overline{x}_7 \lor x_{13} \]
\[ \phi_6 = \overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x}_7 \lor \overline{x}_9 \]
CDCL principles at a glance

Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]
CDCL principles at a glance
Decisions – Propagations

\( \phi_1 = x_1 \lor x_4 \)
\( \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \)
\( \phi_3 = x_1 \lor x_8 \lor x_{12} \)
\( \phi_4 = x_2 \lor x_{11} \)
\( \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \)
\( \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \)
\( \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \)
CDCL principles at a glance

Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]
CDCL principles at a glance

Decisions – Propagations

\[
\begin{align*}
\phi_1 &= x_1 \lor x_4 \\
\phi_2 &= x_1 \lor \overline{x_3} \lor \overline{x_8} \\
\phi_3 &= x_1 \lor x_8 \lor x_{12} \\
\phi_4 &= x_2 \lor x_{11} \\
\phi_5 &= \overline{x_3} \lor \overline{x_7} \lor x_{13} \\
\phi_6 &= \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \\
\phi_7 &= x_8 \lor \overline{x_7} \lor \overline{x_9}
\end{align*}
\]
CDCL principles at a glance

Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]

DL 1: \( \overline{x_1} \rightarrow x_1, \overline{x_4}[\phi_1] \)

DL 2: \( x_3 \rightarrow x_3, \overline{x_8}[\phi_2], x_{12}[\phi_3] \)
CDCL principles at a glance

Decisions – Propagations

\begin{align*}
\phi_1 &= x_1 \lor \overline{x_4} \\
\phi_2 &= x_1 \lor \overline{x_3} \lor \overline{x_8} \\
\phi_3 &= x_1 \lor x_8 \lor x_{12} \\
\phi_4 &= x_2 \lor x_{11} \\
\phi_5 &= \overline{x_3} \lor \overline{x_7} \lor x_{13} \\
\phi_6 &= \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \\
\phi_7 &= x_8 \lor \overline{x_7} \lor \overline{x_9}
\end{align*}

\begin{align*}
\text{DL 1:} & \quad \overline{x_1} \quad \Rightarrow \quad \overline{x_1}, x_{4}[\phi_1] \\
\text{DL 2:} & \quad x_3 \quad \Rightarrow \quad x_3, \overline{x_8}[\phi_2], x_{12}[\phi_3] \\
\text{DL 3:} & \quad \overline{x_2} \quad \Rightarrow \quad \overline{x_2}, x_{11}[\phi_4]
\end{align*}
CDCL principles at a glance
Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]
CDCL principles at a glance

Conflict Analysis

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x}_3 \lor \overline{x}_7 \lor x_{13} \]
\[ \phi_6 = \overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x}_7 \lor \overline{x}_9 \]
CDCL principles at a glance
Conflict Analysis

\[ \beta_1 = \text{res}(x_9, \phi_7, \phi_6) = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \lor \overline{x}_9 \]

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x}_3 \lor \overline{x}_7 \lor x_{13} \]
\[ \phi_6 = \overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x}_7 \lor \overline{x}_9 \]
CDCL principles at a glance

Conflic Analysis

\[ \beta_1 = \text{res}(x_9, \phi_7, \phi_6) = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \lor \overline{x}_{13} \]
\[ \beta = \text{res}(x_{13}, \beta_1, \phi_5) = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \]

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x}_3 \lor \overline{x}_7 \lor x_{13} \]
\[ \phi_6 = \overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x}_7 \lor \overline{x}_9 \]
CDCL principles at a glance

Conflict Analysis

\[ \beta_1 = \text{res}(x_9, \phi_7, \phi_6) = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \lor \overline{x}_{13} \]
\[ \beta = \text{res}(x_{13}, \beta_1, \phi_5) = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \]

- Stops as soon as the resolvent has a unique literal from the last decision level (FUIP).
- \( \beta \) is added to the clauses databases (ensure a systematic search)
CDCL principles at a glance
Non Chronological Backtrackings

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = \overline{x_8} \lor \overline{x_7} \lor \overline{x_9} \]

\[ \beta = \overline{x_3} \lor x_8 \lor \overline{x_7} \]
**CDCL principles at a glance**

**Non Chronological Backtrackings**

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = \overline{x_8} \lor \overline{x_7} \lor \overline{x_9} \]

\[ \beta = \overline{x_3} \lor x_8 \lor \overline{x_7} \]
CDCL principles at a glance
Non Chronological Backtrackings

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x}_3 \lor \overline{x}_7 \lor x_{13} \]
\[ \phi_6 = \overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9 \]
\[ \phi_7 = \overline{x}_8 \lor \overline{x}_7 \lor \overline{x}_9 \]

\[ \beta = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \]
CDCL Principles, graphical point of view
CDCL Principles, graphical point of view
CDCL Principles, graphical point of view

- **Reason** side
  - \(x_1 \lor x_2\)
  - \(\neg x_2 \lor \neg x_3\)
  - \(x_3 \lor x_5 \lor x_6\)
  - \(\neg x_4 \lor x_8 \lor x_9\)

- **Conflict** side
  - \(x_{10} \lor \neg x_9 \lor x_{11}\)
  - \(\neg x_{11} \lor x_8 \lor \neg x_{12}\)
  - \(x_{12} \lor \neg x_{13}\)
  - \(x_7 \lor x_{12} \lor x_{14}\)
CDCL Principles, graphical point of view
CDCL Principles, graphical point of view
CDCL Principles, graphical point of view

CDCL Principles, graphical point of view

x₁ \lor x₂
\neg x₂ \lor \neg x₃
x₃ \lor x₅ \lor x₆
\neg x₄ \lor x₈ \lor x₉
x₁₀ \lor \neg x₉ \lor x₁₁
\neg x₁₁ \lor x₈ \lor \neg x₁₂
x₁₂ \lor \neg x₁₃
x₇ \lor x₁₂ \lor x₁₄

"conflict" side
CDCL Principles, graphical point of view

\[ \neg x_1 \quad \neg x_4 \quad \neg x_7 \quad \neg x_{10} \quad \neg x_{13} \quad \neg x_{16} \]

\[ x_2 \quad \neg x_5 \quad \neg x_8 \quad x_9 \quad x_{11} \quad x_{12} \quad x_{14} \quad x_{15} \quad x_{16} \]

1. \[ x_1 \lor x_2 \]
2. \[ \neg x_2 \lor \neg x_3 \]
3. \[ x_3 \lor x_5 \lor x_6 \]
4. \[ \neg x_4 \lor x_8 \lor x_9 \]

\[ \neg x_2 \lor \neg x_4 \lor \neg x_5 \lor x_7 \lor \neg x_6 \lor \neg x_8 \]
\[ x_{10} \lor \neg x_9 \lor x_{11} \]
\[ \neg x_{11} \lor x_8 \lor \neg x_{12} \]
\[ x_{12} \lor \neg x_{13} \]
\[ \neg x_6 \lor x_{12} \lor x_{15} \]
\[ x_{13} \lor \neg x_{14} \lor \neg x_{16} \]
\[ \neg x_{15} \lor \neg x_{14} \lor x_{16} \]

“conflict" side
\[ \neg x_{16} \lor x_{16} \]
CDCL Principles, graphical point of view

CDCL Principles, graphical point of view:

- The CDCL principles are illustrated graphically.
- Variables and their negations are represented by nodes in the graph.
- The graph shows the propagation of conflicts and the resolution process.
- The "reason" side versus the "conflict" side is highlighted.

CDCL Principles:

1. $x_1 \lor x_2$
2. $\neg x_2 \lor \neg x_3$
3. $x_3 \lor x_5 \lor x_6$
4. $\neg x_4 \lor x_8 \lor x_9$
5. $x_7 \lor \neg x_5 \lor x_6 \lor \neg x_8$
6. $\neg x_6 \lor x_{12} \lor \neg x_{13}$
7. $x_{10} \lor \neg x_9 \lor \neg x_{11}$
8. $x_{11} \lor \neg x_8 \lor \neg x_{12}$
9. $\neg x_6 \lor x_{12} \lor x_{15}$
10. $x_{13} \lor \neg x_4 \lor \neg x_5$
11. $x_{14} \lor \neg x_{14} \lor \neg x_{16}$
12. $\neg x_{15} \lor \neg x_{14} \lor x_{16}$
13. $x_{16} \lor \neg x_{14} \lor \neg x_{15}$

"conflict" side:

- $\neg x_{16} \lor x_{16}$
CDCL Principles, graphical point of view
CDCL Principles, graphical point of view

Coupure «FUIP»

x_1 \lor x_2
\neg x_2 \lor \neg x_3
x_3 \lor x_5 \lor x_6
\neg x_4 \lor x_8 \lor x_9
x_{10} \lor \neg x_9 \lor x_{11}
\neg x_11 \lor x_8 \lor \neg x_{12}
x_{12} \lor \neg x_{13}
\neg x_6 \lor x_{12} \lor x_{14}
\neg x_6 \lor x_{12} \lor x_{15}
x_{13} \lor \neg x_{14} \lor \neg x_{15}
\neg x_{15} \lor \neg x_{14} \lor x_{16}

"conflict" side

\neg x_{16} \lor x_{16}
CDCL Principles, graphical point of view

\[ \neg x_6 \lor x_7 \lor x_8 \lor \neg x_{11} \]

\[ \neg x_6 \lor x_7 \lor x_{12} \]

\[ x_{13} \lor \neg x_{14} \lor \neg x_{15} \]

\[ \neg x_{16} \lor x_{16} \]

Coupure « FUIP »
CDCL Principles, graphical point of view

“reason” side
\[ x_1 \lor \neg x_4 \lor x_7 \lor x_{10} \]

“conflict” side
\[ \neg x_6 \lor x_7 \lor x_8 \lor \neg x_{11} \]

Coupure « FUIP »
\[ x_{13} \lor \neg x_{14} \lor \neg x_{15} \]

\[ x_1 \lor x_2 \]
\[ \neg x_2 \lor \neg x_3 \]
\[ x_3 \lor x_5 \lor x_6 \]
\[ \neg x_4 \lor x_8 \lor x_9 \]
\[ x_{10} \lor \neg x_9 \lor x_{11} \]
\[ \neg x_6 \lor x_{12} \lor x_{15} \]
\[ x_{13} \lor \neg x_{14} \lor \neg x_{16} \]
\[ x_{12} \lor \neg x_{13} \]
\[ x_{14} \lor \neg x_{15} \lor x_{16} \]
CDCL Principles, graphical point of view
CDCL Principles, graphical point of view

\[\begin{align*}
\neg x_1 \lor x_2 \\
\neg x_2 \lor \neg x_3 \\
x_3 \lor x_5 \lor x_6 \\
\neg x_4 \lor x_8 \lor x_9 \\
x_10 \lor \neg x_9 \lor x_{11} \\
x_{10} \lor \neg x_6 \lor x_{11} \\
x_{11} \lor \neg x_8 \lor x_{12} \\
x_{12} \lor \neg x_{13} \\
x_7 \lor x_{12} \lor x_{14} \\
\neg x_6 \lor x_{12} \lor x_{15} \\
x_{13} \lor \neg x_{14} \lor \neg x_{16} \\
\neg x_{15} \lor \neg x_{14} \lor x_{16}
\end{align*}\]
CDCL Principles, graphical point of view
**Conflict when propagating**

\[ x_{16} \lor \neg x_{15} \lor \neg x_{14} \]

Let's do the conflict analysis.

\[
\begin{align*}
& x_{16} \lor \neg x_{15} \lor \neg x_{14} \\
& \neg x_{15} \lor \neg x_{14} \lor x_{13} \\
& \neg x_{14} \lor x_{13} \lor x_{12} \lor \neg x_{6} \\
& x_{13} \lor x_{12} \lor x_{7} \lor \neg x_{6} \\
& x_{12} \lor x_{7} \lor \neg x_{6} 
\end{align*}
\]

Simply follows the order of the trail!
VSIDS : the killing heuristics
Variable State Independant Decaying Sum

Idea : The heuristics is state independant

Award variables (not literals) view

- in the most recent learnt clauses
- during the last conflict analysis

Well, yes...But where to branch at the beginning?

The heuristics is **highly dynamic (really, really)**

- in 1/30s (not in Python) a variable can fall from the top to the abysses

Side effect: the solver is very hard to understand/predict
VSIDS dynamicity

Exponential increasing of the increment

After each conflict, it is multiplied by $1/v$ ($v = 0.95$)
VSIDS and the heap

Some tricks:
- Uses a heap for sorting variables
- When a variable is chosen by the heuristics, remove it from the heap (free)
- When a variable is propagated, don’t remove it from the heap (let’s be lazy)
- Thus, picks variables from the heap until it is unassigned
- Push back variables to the heap when backtracking
Restarts, common idea

Restarts are commonly used in local search algorithms. **Idea:** when the search failed, look at another part of the search space (looking for a solution).

**Heavy Tailed Phenomenon Observed on Search Trees**

![Graph showing sample mean of backtrack search vs. standard distribution](image)
Restarts, common idea

Restarts are commonly used in local search algorithms. 

**Idea:** when the search failed, look at another part of the search space (looking for a solution).

---

**Heavy Tailed Phenomenon Observed on Search Trees**

![Graph showing erratic behavior of the sample mean of backtrack search for completing a quasi-group (order 11, 30% pre-assignment) vs. stabilized behavior of the sample mean for a standard distribution (gamma).](image)

In this case, we see that the sample mean converges rapidly to a constant value with increasing sample size. On the other hand, the heavy-tailed distribution in Figure 1(a) shows a highly erratic behavior of the mean that does not stabilize with increasing sample size.

Previous authors have discovered the related – and quite surprising – phenomenon of so-called exceptionally hard SAT problems in fixed problem distributions [20, 62]. For these instances, we further observed that when a small amount of

The median, not shown here, stabilizes rather quickly at the value 1.
Restarts, common idea

Restarts are commonly used in local search algorithms. **Idea**: when the search failed, look at another part of the search space (looking for a solution).

Restarts are introduced to fight this phenomenon

**Note**: Learning allows frequent restarts. The completeness is still guarantee.
Now: Ultra-Rapid Restarts

**Last progress** Luby-law restarts

\[(1,2,1,2,4,1,2,1,2,4,8,1,2,1,2,4,1,2,1,2,4,8,16,...)\]

Multiplied by a constant (32, 64, 128, 512).

**Other strategies** decision levels / LBD based / agility

**Idea** When the solver does not make any progress, restart

**Important thing**: restart a lot (optimal: Luby 6), use in conjunction with phase-saving

**Restarts do not restart search** Heuristics are maintained during a restarts. The solver directly goes to the same search space, but with a distinct path.
Now: Ultra-Rapid Restarts

**Last progress** Luby-law restarts

(1,2,1,2,4,1,2,1,2,4,8,1,2,1,2,4,1,2,1,2,4,8,16,...)

Multiplied by a constant (32, 64, 128, 512).

**Other strategies** decision levels / LBD based / agility

**Idea** When the solver does not make any progress, restart

**Important thing:** restart a lot (optimal: Luby 6), use in conjunction with phase-saving

**Restarts do not restart search** Heuristics are maintained during a restarts. The solver directly goes to the same search space, but with a distinct path.
Many other ingredients

Conflict Clause Minimization

Phase Saving

Blocking Literals

Pre-processing

In-Processing

...
Clause Database Management

In the early 2000, Chaff was dying because of memory (very quickly) (some) learnt clauses had to be removed

- Regularly remove unseen clauses during last conflict analysis
- Extends the notion of VSIDS to clauses

Keeping as many clauses as possible was believed to be crucial

- Essential for completeness!
- Especially if we restart a lot!
Glucose: the last SAT surprize

We can remove 95% of learnt clauses and get better!

Instead of removing clauses when needed, we eagerly remove them very often

Remove half of the clause at each point of a dynamic strategy

Clause activity is not relevant. Use the notion of LBD instead
Literal Block Distance (LBD) – initial idea (2009)

- One decision often creates a lot of propagated literals ("blocks")
  - Those variables will probably be propagated together again and again
- Reducing decisions? Adds dependencies between independent blocks
- How? Add the strongest possible constraints between them

**LBD of a learnt clause: number of prop. blocks of literals**

- Small LBD scores are better
- The importance of "Glue Clauses" (LBD=2)
  - Only one literal from the last decision level (the assertive one)
  - This literal will be **glued** to the other block
  - Kept forever in glucose

The restart policy is based on LBD too
What about parallel CDCL?

**Bad idea:** Try to parallelize the BCP engine

**A very difficult task:** each engine is eagerly using the memory

Even sharing a clause is not a good idea

**The 2-Watched literal scheme is a nightmare for sharing the clause** in a single core engine the first 2 literals are a shared information between the 2 watched list.

So, the state of the art is to **duplicate** clauses
Many Cores

- Multi-core architecture, cloud \( \Rightarrow \) design of parallel solvers
- Different strategies
  - Divide and conquer
    - Explicitly splits search w.r.t. assignments
  - Portfolio algorithms
    - Each thread: its own solver and the whole formula
    - Rely on orthogonal searches
  - Parallel solver
    - communication: learnt clauses
    - All threads working on the same proof
Many Cores, Many more problems

Sharing clauses in parallel has many drawbacks

- Imported clauses can be bad (noise, wrong way, ...)
- Imported clauses can be subsumed / useless
- Imported clauses can dominate learnt clauses
- Each thread has to manage many more clauses
- Many side effects on all core components

**Currently:** Clauses are sent as soon as they are learnt, plus:

- **MANYSAT 1.0:** size ≤ 8
- **MANYSAT 1.1:** dynamically adjust the threshold
- **PLINGELING:** size ≤ 30 and LBD ≤ 8
- **Penelope:** LBD ≤ 8 (PSM allows more clauses exchanges)
Many *useless clauses* even in single engine solvers

![Graph showing number of conflicts vs. number of useless clauses in final proof](image)

- **x-axis**: Number of conflicts
- **y-axis**: Useless clauses in final proof (UNSAT formulas)
How many times clauses are seen in conflicts

- x-axis: Number of conflicts
- y-axis: Number of clauses seen at least Z times

- L.R. Seen 1 time (m=0.91)
- L.R. Seen 2 times (m=0.34)
- L.R. Seen 3 times (m=0.22)
- L.R. Seen 4 times (m=0.17)
Today’s Itinerary

Introduction

Propositional Logic: simple and complex

DP & DPLL

Applications

CDCL

Understanding CDCL as complex systems

Conclusion
Do we really understand what we have built?

We know how to build an efficient (single engine) SAT solver, but:

- CDCL is not DPLL
  - because of ultra-rapid restarts and aggressive clause DB cleaning
- Learning can be bad
  - we’ll see that keeping all clauses is not a winning strategy
- Restarting is not restarting
  - directly go to the same search space, by another path
- Luby-based restarts are dangerous
  - rare but very large windows are following a fixed restart strategy
- What is the phase?
  - good to reach a solution or a contradiction?
- “good” variables: top or bottom of the tree?
  - splitting on top, resolving on bottom variables
- glucose is not complete!
  - Too many restarts and forgetting
CDCL solvers are complex systems

We have a lot of open problems around these questions:

"Understand what we have implemented"

It’s ok if we don’t fully “understand” our code

- Very fast and unpredictable
- Work well on real-world instances, but how to define such a structure?
- All components are tightly connected, side effects are everywhere
- There is no “one-simple reason” explaining their performance (supposition)
- At least we know that we don’t know

Idea behind glucose

A real experimental study of CDCL solvers
CDCL solvers are complex systems – Illustration

Example of a real conflict analysis:

- Many resolutions at each conflict
- Very reactive VSIDS (1/10s lifetime)

But: A clear structure behind!

A part of the research in SAT is hunting for the structure behind CDCL mechanisms
How to understand this (trivial) proof?

- Less than 70,000 conflicts
- Solved in a few seconds (not in Python!)
- A very dense proof
- Hard to understand
Outliers everywhere!

We try to understand CDCL solvers but all problems are distincts!

Number of decisions

stats after 10,000 conflicts on 1164 “not easy” problems from all previous contests
Outliers everywhere!

We try to understand CDCL solvers but all problems are distincts!

Decisions per conflicts

stats after 10,000 conflicts on 1164 “not easy” problems from all previous contests
Outliers everywhere!

We try to understand CDCL solvers but all problems are distincts!

“True” (non binary) glue clauses

stats after 10,000 conflicts on 1164 “not easy” problems from all previous contests
Outliers everywhere!

We try to understand CDCL solvers but all problems are distincts!

Successive conflicts

stats after 10,000 conflicts on 1164 “not easy” problems from all previous contests
Outliers everywhere!

We try to understand CDCL solvers but all problems are distincts!

Propagations

stats after 10,000 conflicts on 1164 “not easy” problems from all previous contests
When experimenting suggests that CDCL solvers are inefficient

Most of solver’s time is spent in unit propagation

But, on UNSAT instances:
- Only 50% of generated clauses are useful for deriving the final contradiction
- Only 20% of unit propagation are used during conflict analysis

Only 10% of solver’s time is useful for deriving the contradiction!
Today's Itinerary

Introduction

Propositional Logic: simple and complex

DP & DPLL

Applications

CDCL

Understanding CDCL as complex systems

Conclusion
(Possible) Illustration of learnt clause mechanism
(Possible) Illustration of learnt clause mechanism
(Possible) Illustration of learnt clause mechanism
(Possible) Illustration of learnt clause mechanism
(Possible) Illustration of learnt clause mechanism
(Possible) Illustration of learnt clause mechanism
(Possible) Illustration of learnt clause mechanism
We now how to built an efficient SAT Solver but we can hardly explain their power.

Experimenting is essential now in SAT research.

LBD are related to communities (but may be more “semantic”)

We (may) need to see CDCL has clauses producers, not branching algorithms
(much more difficult to handle, but closer to reality)

What about modelizing SAT solvers?