State-of-the-art SAT Solving

Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

Summer School on Formal Techniques, May 22, 2017
Dress Code as Satisfiability Problem

The SAT problem: Can a formula in propositional logic be satisfied?

Propositional logic

- Boolean variables: tie and shirt (for the example below)
- Logic symbols: ¬ (not), ∨ (or), ∧ (and)
- Literals: tie, ¬tie, shirt, and ¬shirt

Three conditions / clauses:
- not wearing a tie nor a shirt is impolite  \((\text{tie} \lor \text{shirt})\)
- clearly one should not wear a tie without a shirt  \((\neg \text{tie} \lor \text{shirt})\)
- wearing a tie and a shirt is overkill  \((\neg (\text{tie} \land \text{shirt})) \equiv (\neg \text{tie} \lor \neg \text{shirt})\)

Is the formula \((\text{tie} \lor \text{shirt}) \land (\neg \text{tie} \lor \text{shirt}) \land (\neg \text{tie} \lor \neg \text{shirt})\) satisfiable?
The Satisfiability (SAT) problem

\[(x_5 \lor x_8 \lor \bar{x}_2) \land (x_2 \lor \bar{x}_1 \lor \bar{x}_3) \land (\bar{x}_8 \lor \bar{x}_3 \lor \bar{x}_7) \land (\bar{x}_5 \lor x_3 \lor x_8) \land \\
(\bar{x}_6 \lor \bar{x}_1 \lor \bar{x}_5) \land (x_8 \lor \bar{x}_9 \lor x_3) \land (x_2 \lor x_1 \lor x_3) \land (\bar{x}_1 \lor x_8 \lor x_4) \land \\
(\bar{x}_9 \lor \bar{x}_6 \lor x_8) \land (x_8 \lor x_3 \lor \bar{x}_9) \land (x_9 \lor \bar{x}_3 \lor x_8) \land (x_6 \lor \bar{x}_9 \lor x_5) \land \\
(x_2 \lor \bar{x}_3 \lor \bar{x}_8) \land (x_8 \lor \bar{x}_6 \lor \bar{x}_3) \land (x_8 \lor x_3 \lor \bar{x}_1) \land (\bar{x}_8 \lor x_6 \lor \bar{x}_2) \land \\
(x_7 \lor x_9 \lor \bar{x}_2) \land (x_8 \lor \bar{x}_9 \lor x_2) \land (\bar{x}_1 \lor \bar{x}_9 \lor x_4) \land (x_8 \lor x_1 \lor \bar{x}_2) \land \\
(x_3 \lor \bar{x}_4 \lor \bar{x}_6) \land (\bar{x}_1 \lor \bar{x}_7 \lor x_5) \land (\bar{x}_7 \lor x_1 \lor x_6) \land (\bar{x}_5 \lor x_4 \lor \bar{x}_6) \land \\
(\bar{x}_4 \lor x_9 \lor \bar{x}_8) \land (x_2 \lor x_9 \lor x_1) \land (x_5 \lor \bar{x}_7 \lor x_1) \land (\bar{x}_7 \lor \bar{x}_9 \lor \bar{x}_6) \land \\
(x_2 \lor x_5 \lor x_4) \land (x_8 \lor \bar{x}_4 \lor x_5) \land (x_5 \lor x_9 \lor x_3) \land (\bar{x}_5 \lor \bar{x}_7 \lor x_9) \land \\
(x_2 \lor \bar{x}_8 \lor x_1) \land (\bar{x}_7 \lor x_1 \lor x_5) \land (x_1 \lor x_4 \lor x_3) \land (x_1 \lor \bar{x}_9 \lor \bar{x}_4) \land \\
(x_3 \lor x_5 \lor x_6) \land (\bar{x}_6 \lor x_3 \lor \bar{x}_9) \land (\bar{x}_7 \lor x_5 \lor x_9) \land (x_7 \lor \bar{x}_5 \lor \bar{x}_2) \land \\
(x_4 \lor x_7 \lor x_3) \land (x_4 \lor \bar{x}_9 \lor \bar{x}_7) \land (x_5 \lor \bar{x}_1 \lor x_7) \land (x_5 \lor \bar{x}_1 \lor x_7) \land \\
(x_6 \lor x_7 \lor \bar{x}_3) \land (\bar{x}_8 \lor \bar{x}_6 \lor \bar{x}_7) \land (x_6 \lor x_2 \lor x_3) \land (\bar{x}_8 \lor x_2 \lor x_5) \land \\
\]

Does there exist an assignment satisfying all clauses?
Search for a satisfying assignment (or proof none exists)
$\mathcal{NP}$-Complete: Good or Bad News?

SAT is the first $\mathcal{NP}$-complete problem  \[\text{[Cook’71]}\]
**NP-Complete: Good or Bad News?**

SAT is the first $\text{NP}$-complete problem \cite{Cook71}.

**Bad?**

- Only exponential time solving algorithms are known
- Probably no polynomial time algorithm exists ($\text{P} \neq \text{NP}$)
**NP-Complete: Good or Bad News?**

SAT is the first $\mathcal{NP}$-complete problem \[\text{[Cook’71]}\]

**Bad?**
- Only exponential time solving algorithms are known
- Probably no polynomial time algorithm exists ($\mathcal{P} \neq \mathcal{NP}$)

**Good!**
- SAT solvers are powerful tools for real world problems
- All problems in $\mathcal{NP}$ can be translated into SAT in polynomial time
Motivation

From 100 variables, 200 constraints (early 90s) to 1,000,000 vars. and 20,000,000 cls. in 20 years.
Motivation

From 100 variables, 200 constraints (early 90s) to **1,000,000** vars. and **20,000,000** cls. in 20 years.

Applications:
Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.
Motivation

From 100 variables, 200 constraints (early 90s) to 1,000,000 vars. and 20,000,000 cls. in 20 years.

Applications:
Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.

SAT used to solve many other problems!
Capitalise on the performance of SAT solvers

- formal verification
- graph theory
- bioinformatics
- train safety
- timetabling
- number theory
- cryptography
- rewriting termination

Encode

SAT solver

Decode
Overview

Search for Lemmas (Today)
- Learning Lemmas
- Data-structures
- Heuristics

Search for Simplification (Tomorrow)
- Variable elimination
- Blocked clause elimination
- Unhiding redundancy
Conflict-driven SAT solvers: Search and Analysis

\[(x_1 \lor x_4) \land (x_3 \lor \neg x_4 \lor \neg x_5) \land (\neg x_3 \lor \neg x_2 \lor \neg x_4) \land F_{\text{extra}} \]
Conflict-driven SAT solvers: Search and Analysis

\[(x_1 \lor x_4) \land (x_3 \lor \overline{x_4} \lor \overline{x_5}) \land (\overline{x_3} \lor \overline{x_2} \lor \overline{x_4}) \land F_{\text{extra}}\]

\(x_5 = 1\)
Conflict-driven SAT solvers: Search and Analysis

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land F_{\text{extra}}\]

\[x_5 = 1\]
\[x_2 = 1\]
Conflict-driven SAT solvers: Search and Analysis

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \mathcal{F}_{\text{extra}}\]
Conflict-driven SAT solvers: Search and Analysis

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land F_{extra}\]
Conflict-driven SAT solvers: Search and Analysis

\[(x_1 \lor x_4) \land (x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land (\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land F_{\text{extra}}\]
Conflict-driven SAT solvers: Search and Analysis

\[
\begin{align*}
(x_1 \lor x_4) \land \\
(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land \\
(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \\
F_{\text{extra}}
\end{align*}
\]
Conflict-driven SAT solvers: Search and Analysis

\[ (x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \mathcal{F}_{\text{extra}} \]

Diagram:
- Root node 0
- Node 1: \( x_5 = 1 \)
- Node 2: \( x_2 = 1 \)
- Node 6: \( x_1 = 0 \), \( x_4 = 1 \)
- Node 7: \( x_3 = 1 \)
- Node 7: \( x_3 = 0 \)

Assignments:
- \( x_1 = 0 \)
- \( x_2 = 1 \)
- \( x_3 = 1 \)
- \( x_3 = 0 \)
- \( x_4 = 1 \)
- \( x_5 = 1 \)
Conflict-driven SAT solvers: Search and Analysis

\[ \big( x_1 \lor x_4 \big) \land \\
\big( x_3 \lor \bar{x}_4 \lor \bar{x}_5 \big) \land \\
\big( \bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4 \big) \land \\
\mathcal{F}_{\text{extra}} \]

\[ (\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5) \]
Conflict-driven SAT solvers: Search and Analysis

\[(x_1 \lor x_4) \land (x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land (\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \mathcal{F}_{\text{extra}}\]
Conflict-driven SAT solvers: Search and Analysis

\[
\begin{align*}
F_{\text{extra}} &= (x_1 \lor x_4) \land (x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land \\
&\quad (\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \\
&\quad (\overline{x}_2 \lor \overline{x}_4 \lor \overline{x}_5)
\end{align*}
\]
Conflict-driven SAT solvers: Search and Analysis

\[
(x_1 \lor x_4) \land \\
(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land \\
(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \\
F_{\text{extra}}
\]

\[
\begin{align*}
(x_1 = 0) & \quad (x_4 = 1) \\
(x_2 = 1) & \quad (x_3 = 0) \\
(x_5 = 1) & \\
& \quad (\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)
\end{align*}
\]
Conflict-driven SAT solvers: Pseudo-code

1: while TRUE do
2: \( l_{\text{decision}} := \text{GetDecisionLiteral}() \)
3: If no \( l_{\text{decision}} \) then return satisfiable
4: \( \mathcal{F} := \text{Simplify}( \mathcal{F}(l_{\text{decision}} \leftarrow 1) ) \)
5: while \( \mathcal{F} \) contains \( C_{\text{falsified}} \) do
6: \( C_{\text{conflict}} := \text{AnalyzeConflict}( C_{\text{falsified}} ) \)
7: If \( C_{\text{conflict}} = \emptyset \) then return unsatisfiable
8: \( \text{BackTrack}( C_{\text{conflict}} ) \)
9: \( \mathcal{F} := \text{Simplify}( \mathcal{F} \cup \{ C_{\text{conflict}} \} ) \)
10: end while
11: end while
Learning conflict clauses [Marques-SilvaSakallah’96]
Learning conflict clauses

\[ (\neg x_1 \lor \neg x_3 \lor x_5 \lor x_{17} \lor \neg x_{19}) \]

tri-asserting clause
Learning conflict clauses

\[(x_{10} \lor \neg x_{17} \lor x_{19})\]

first unique implication point
Learning conflict clauses

$[\text{Marques-SilvaSakallah’96}]$

$\frac{x_{12}=0}{x_6=0}$

$\frac{x_4=1}{x_{10}=0}$

$\frac{x_1=1}{x_{18}=1}$

$\frac{x_{17}=0}{x_{18}=0}$

$\frac{x_3=1}{x_5=0}$

$\frac{x_8=1}{x_9=1}$

$\left(x_2 \lor \neg x_4 \lor \neg x_8 \lor x_{17} \lor \neg x_{19}\right)$

second unique implication point
Average Learned Clause Length
Data-structures

Watch pointers
Simple data structure for unit propagation
Conflict-driven: Watch pointers (1) [MoskewiczMZZM’01]

\[ \varphi = \{ x_1 = *, x_2 = *, x_3 = *, x_4 = *, x_5 = *, x_6 = * \} \]
\[ \varphi = \{ x_1 = *, x_2 = *, x_3 = *, x_4 = *, x_5 = 1, x_6 = * \} \]
Conflict-driven: Watch pointers (1) [MoskewiczMZZM’01]

\[ \varphi = \{ \neg x_1 = *, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = * \} \]
Conflict-driven: Watch pointers (1) [MoskewiczMZZM’01]

\[ \varphi = \{ x_1 = *, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = * \} \]
\[ \varphi = \{ x_1 = 1, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = * \} \]
Conflict-driven: Watch pointers (1) [MoskewiczMZZM’01]

\[ \varphi = \{ x_1 = 1, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = * \} \]
$\varphi = \{ x_1 = 1, x_2 = *, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = * \}$

\[\begin{array}{cccccc}
x_6 & & x_2 & & x_3 & & x_5 & & x_1 \\
\end{array}\]

\[\begin{array}{cccccc}
x_1 & & x_4 & & x_3 & & x_5 & & x_6 \\
\end{array}\]
Conflict-driven: Watch pointers (1) [MoskewiczMZZM’01]

\[ \varphi = \{ x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = * \} \]
Conflict-driven: Watch pointers (1) [MoskewiczMZZM’01]

\[ \varphi = \{ x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 1 \} \]
\[ \varphi = \{ x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 1 \} \]
Conflict-driven: Watch pointers (2) [MoskewiczMZZM’01]

Only examine (get in the cache) a clause when both

- a watch pointer gets falsified
- the other one is not satisfied

While backjumping, just unassign variables

Conflict clauses → watch pointers

No detailed information available

Not used for binary clauses
Average Number Clauses Visited Per Propagation
Percentage visited clauses with other watched literal true
Heuristics
Most important CDCL heuristics

Variable selection heuristics

- aim: minimize the search space
- plus: could compensate a bad value selection
Most important CDCL heuristics

Variable selection heuristics
- aim: minimize the search space
- plus: could compensate a bad value selection

Value selection heuristics
- aim: guide search towards a solution (or conflict)
- plus: could compensate a bad variable selection, cache solutions of subproblems [PipatsrisawatDarwiche’07]
Most important CDCL heuristics

Variable selection heuristics

- aim: minimize the search space
- plus: could compensate a bad value selection

Value selection heuristics

- aim: guide search towards a solution (or conflict)
- plus: could compensate a bad variable selection, cache solutions of subproblems [PipatsrisawatDarwiche’07]

Restart strategies

- aim: avoid heavy-tail behavior [GomesSelmanCrato’97]
- plus: focus search on recent conflicts when combined with dynamic heuristics
Variable selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers
Variable selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Variable State Independent Decaying Sum (VSIDS)

- original idea (zChaff): for each conflict, increase the score of involved variables by 1, half all scores each 256 conflicts
  
  [MoskewiczMZZM’01]

- improvement (MiniSAT): for each conflict, increase the score of involved variables by $\delta$ and increase $\delta := 1.05\delta$

  [EenSörensson’03]
Visualization of VSIDS in PicoSAT

http://www.youtube.com/watch?v=M0jhFywLre8
Value selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers
Value selection heuristics

Based on the occurrences in the (reduced) formula
  - examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
  - not practical for CDCL solver due to watch pointers

Based on the encoding / consequently
  - negative branching (early MiniSAT) [EenSörensson’03]
Value selection heuristics

Based on the occurrences in the (reduced) formula
  ▶ examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
  ▶ not practical for CDCL solver due to watch pointers

Based on the encoding / consequently
  ▶ negative branching (early MiniSAT) [EenSörensson’03]

Based on the last implied value (phase-saving)
  ▶ introduced to CDCL [PipatsrisawatDarwiche’07]
  ▶ already used in local search [HirschKojevnikov’01]
Heuristics: Phase-saving

Selecting the last implied value remembers solved components

negative branching  phase-saving

[PipatsrisawatDarwiche’07]
Restarts

Restarts in CDCL solvers:

- Counter heavy-tail behavior [GomesSelmanCrato’97]
- Unassign all variables but keep the (dynamic) heuristics
Restarts

Restarts in CDCL solvers:
- Counter heavy-tail behavior \[\text{[GomesSelmanCrato'97]}\]
- Unassign all variables but keep the (dynamic) heuristics

Restart strategies: \[\text{[Walsh'99, LubySinclairZuckerman'93]}\]
- Geometrical restart: e.g. 100, 150, 225, 333, 500, 750, . . .
- Luby sequence: e.g. 100, 100, 200, 100, 100, 200, 400, . . .
Restarts

Restarts in CDCL solvers:
- Counter heavy-tail behavior \[\text{[GomesSelmanCrato’97]}\]
- Unassign all variables but keep the (dynamic) heuristics

Restart strategies: [Walsh’99, LubySinclairZuckerman’93]
- Geometrical restart: e.g. 100, 150, 225, 333, 500, 750, \ldots
- Luby sequence: e.g. 100, 100, 200, 100, 100, 200, 400, \ldots

Rapid restarts by reusing trail: [vanderTakHeuleRamos’11]
- Partial restart same effect as full restart
- Optimal strategy Luby-1: 1, 1, 2, 1, 1, 2, 4, \ldots
Conflict-Clause Minimization
Self-Subsumption

Use self-subsumption to shorten conflict clauses

\[
\frac{C \lor I \quad D \lor \bar{I}}{D} \quad C \subseteq D
\]

\[
\frac{(a \lor b \lor I) \quad (a \lor b \lor c \lor \bar{I})}{(a \lor b \lor c)}
\]

Conflict clause minimization is an important optimization.
Use self-subsumption to shorten conflict clauses

\[
\frac{C \lor l}{D \lor \overline{l}} \quad D \subseteq C \quad \frac{(a \lor b \lor l)(a \lor b \lor c \lor \overline{l})}{(a \lor b \lor c)}
\]

Conflict clause minimization is an important optimization.

Use implication chains to further minimization:

\[
\ldots (\overline{a} \lor b)(\overline{b} \lor c)(a \lor c \lor d) \ldots \quad \Rightarrow \quad \ldots (\overline{a} \lor b)(\overline{b} \lor c)(c \lor d) \ldots
\]
Conflict-clause minimization

[SörenssonBiere’09]

\[
\begin{align*}
x_1 &= 0 & x_2 &= 1 & x_3 &= 0 \\
x_4 &= 1 & x_5 &= 0 & x_6 &= 1 & x_7 &= 0 \\
x_8 &= 1 & x_9 &= 0 & x_{10} &= 1 & x_{11} &= 0 & x_{12} &= 1 & x_{13} &= 0 \\
x_{14} &= 1 & x_{15} &= 0 & x_{13} &= 1
\end{align*}
\]
Conflict-clause minimization

x₁ = 0
x₄ = 1
x₈ = 1
x₁₀ = 1

(¬x₂ ∨ x₅ ∨ ¬x₆ ∨ x₇ ∨ x₁₁)

first unique implication point

x₂ = 1
x₅ = 0
x₉ = 0
x₁₁ = 0
x₁₄ = 1

x₃ = 0
x₆ = 1
x₁₂ = 1
x₁₅ = 0

x₇ = 0
x₁₃ = 0

[SörenssonBiere'09]
Conflict-clause minimization

\[ (x_1 \lor \overline{x}_4 \lor \overline{x}_8 \lor \overline{x}_{10}) \]

last unique implication point
Conflict-clause minimization

\[
\begin{align*}
 x_1 &= 0 & x_2 &= 1 & x_3 &= 0 \\
 x_4 &= 1 & x_5 &= 0 & x_6 &= 1 & x_7 &= 0 \\
 x_8 &= 1 & x_9 &= 0 & x_{11} &= 0 & x_{12} &= 1 & x_{13} &= 0 \\
 x_{10} &= 1 & x_{14} &= 1 & x_{15} &= 0 & x_{13} &= 1
\end{align*}
\]

\[\neg x_2 \lor x_5 \lor \neg x_6 \lor x_{11}\]

reduced conflict clause

[SörenssonBiere’09]
Conflict-clause minimization

\[
\begin{align*}
\bar{x}_2 \lor x_5 \lor \lor x_{11}
\end{align*}
\]

minimized conflict clause

[SörenssonBiere’09]
Conclusions: state-of-the-art CDCL solver

Key contributions to CDCL solvers:

- concept of conflict clauses (grasp) [Marques-SilvaSakallah’96]
- restart strategies [GomesSC’97, LubySZ’93]
- 2-watch pointers and VSIDS (zChaff) [MoskewiczMZZM’01]
- efficient implementation (Minisat) [EenSörensson’03]
- phase-saving (Rsat) [PipatsrisawatDarwiche’07]
- conflict-clause minimization [SörenssonBiere’09]

+ Pre- and in-processing techniques