From SAT To SMT: Part I

Vijay Ganesh
MIT
Software Engineering & SMT Solvers
An Indispensable Tactic for Any Strategy

Formal Methods
Program Analysis
Automatic Testing
Program Synthesis

SAT/SMT Solvers
Foundation of Software Engineering
Logic Abstractions of Computation

- Formal Methods
- Program Analysis
- Program Reasoning
- Automatic Testing
- Program Synthesis
Foundation of Software Engineering

Logic Abstractions of Computation

- Formal Methods
- Program Analysis
- Automatic Testing
- Program Synthesis

Logics (Boolean,...)
Software Engineering using Solvers
Engineering, Usability, Novelty

Program Reasoning Tool

Program Specification

Logic Formulas

SAT/SMT Solver

SAT/UNSAT

Program is Correct?
or Generate Counterexamples (Test cases)
SAT/SMT Solver Research Story
A 1000x Improvement

- Solver-based programming languages
- Compiler optimizations using solvers
- Solver-based debuggers
- Solver-based type systems
- Solver-based concurrency bugfinding
- Solver-based synthesis
- Bio & Optimization

- Concolic Testing
- Program Analysis
- Equivalence Checking
- Auto Configuration

- Bounded MC
- Program Analysis
- AI

1,000,000 Constraints
100,000 Constraints
10,000 Constraints
1,000 Constraints

The SAT/SMT Problem

- Rich logics (Modular arithmetic, Arrays, Strings,...)
- NP-complete, PSPACE-complete,...
- Practical, scalable, usable, automatic
- Enable novel software reliability approaches
Topics Covered

- Motivation for SAT/SMT solvers in software engineering
- High-level description of the SAT/SMT problem & logics

Rest of the lecture

- Modern SAT solver architecture & techniques
- Modern SMT solver architecture & techniques
- My own contributions: STP & HAMPI
- SAT/SMT-based applications
- Future of SAT/SMT solvers
- Some history (who, when,...) and references sprinkled throughout the talk
The Boolean SAT Problem

Basic Definitions and Format

A literal $p$ is a Boolean variable $x$ or its negation $\neg x$.

A clause $C$ is a disjunction of literals: $x_2 \lor \neg x_4 \lor x_{15}$

A CNF is a conjunction of clauses: $(x_2 \lor \neg x_1 \lor x_5) \land (x_6 \lor \neg x_2) \land (x_3 \lor \neg x_4 \lor \neg x_6)$

All Boolean formulas assumed to be in CNF

Assignment is a mapping (binding) from variables to Boolean values (True, False).

A unit clause $C$ is a clause with a single unbound literal

The SAT-problem is:

Find an assignment s.t. each input clause has a true literal (aka input formula has a solution or is SAT)

OR establish input formula has no solution (aka input formula is UNSAT)

The Input formula is represented in DIMACS Format:

```
c DIMACS
p cnf 6 3
2 -1 5 0
6 -2 0
3 -4 -6 0
```
DPLL SAT Solver Architecture
The Basic Solver

DPLL(Θ_{cnf}, assign) {
  Propagate unit clauses;
  if ”conflict”: return FALSE;
  if ”complete assign”: return TRUE;
  ”pick decision variable x”;
  return DPLL(Θ_{cnf} | x=0, assign[x=0]) || DPLL(Θ_{cnf} | x=1, assign[x=1]);
}

• Propagate (Boolean Constant Propagation):
  • Propagate inferences due to unit clauses
  • Most time in solving goes into this

• Detect Conflict:
  • Conflict: partial assignment is not satisfying

• Decide (Branch):
  • Choose a variable & assign some value

• Backtracking:
  • Implicitly done by the recursion
Modern CDCL SAT Solver Architecture

Key Steps and Data-structures

Key steps

- Decide()
- Propagate()
  (BCP: Boolean constraint propagation)
- Conflict analysis and learning()
- Backjump()
- Forget()
- Restart()

CDCL: Conflict-Driven Clause-Learning

- Conflict analysis is a key step
- Results in learning a conflict clause
- Prunes the search space

Key data-structures (State):

- Stack or trail of partial assignments (AT)
- Input clause database
- Conflict clause database
- Conflict graph
- Decision level (DL) of a variable

Input SAT Instance

Propagate() (BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

Return SAT

Decide()

TopLevel Conflict?

BackJump()

Return UNSAT
Input SAT Instance

Propagate()
(BCP)

No Conflict?

All Vars Assigned?

Conflicts Analysis()

TopLevel Conflict?

Return SAT

Decide()

Return UNSAT

BackJump()

• Propagate (Boolean Constant Propagation):
  • Propagate inferences due to unit clauses
  • Most time in solving goes into this

• Decide:
  • Choose a variable & assign some value (decision)
  • Each decision is a decision level
  • Imposes dynamic variable order

• Conflict analysis and clause learning:
  • Analyze the reason (learn conflict clause)
  • Conflict clause blocks the non-satisfying & a large set of other 'no-good' assignments

• BackJump:
  • Undo the decision(s) that caused no-good assignment
  • Assign 'decision variables' different values
  • Go back several decision levels

Marques-Silva & Sakallah (1999)

Marques-Silva & Sakallah (1999)

Backtrack (Davis, Putnam, Loveland, Logemann 1962)
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

- Propagate (Boolean Constant Propagation):
  - Propagate inferences due to unit clauses
  - Most time in solving goes into this

- Decide:
  - Choose a variable & assign some value (decision)
  - Each decision is a decision level
  - Imposes dynamic variable order

- Conflict analysis and clause learning:
  - Analyze the reason (learn conflict clause)
  - Conflict clause blocks the non-satisfying & a large set of other 'no-good' assignments

- BackJump:
  - Undo the decision(s) that caused no-good assignment
  - Assign 'decision variables' different values
  - Go back several decision levels

- Backtrack (Davis, Putnam, Loveland, Logemann 1962)

Input SAT Instance

Propagate() (BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

Return SAT

Decide()

TopLevel Conflict?

Return UNSAT

BackJump()
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

- **Propagate (Boolean Constant Propagation):**
  - Propagate inferences due to unit clauses
  - Most time in solving goes into this

- **Detect Conflict?**
  - Conflict: partial assignment is not satisfying

- **Conflict Analysis:**
  - Analyze the reason (learn conflict clause)
  - Conflict clause blocks the non-satisfying & a large set of other 'no-good' assignments

- **BackJump:**
  - Undo the decision(s) that caused no-good assignment
  - Assign 'decision variables' different values
  - Go back several decision levels

Input SAT Instance

- Propagate() (BCP)
- No Conflict?
- All Vars Assigned?
- Conflict Analysis()
- TopLevel Conflict?
- Decide()
- Return SAT
- BackJump()
- Return UNSAT
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

Input SAT Instance

Propagate() (BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

Return SAT

Decide()

TopLevel Conflict?

Return UNSAT

BackJump()

• Propagate (Boolean Constant Propagation):
  • Propagate inferences due to unit clauses
  • Most time in solving goes into this

• Detect Conflict?
  • Conflict: partial assignment is not satisfying

• Conflict clause blocks the non-satisfying & a large set of other 'no-good' assignments

• BackJump:
  • Undo the decision(s) that caused no-good assignment
  • Assign 'decision variables' different values
  • Go back several decision levels

• Backtrack (Davis, Putnam, Loveland, Logemann 1962)

Vijay Ganesh, Dagstuhl, Aug 8-12, 2011
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

- Propagate (Boolean Constant Propagation):
  - Propagate inferences due to unit clauses
  - Most time in solving goes into this

- Detect Conflict?
  - Conflict: partial assignment is not satisfying

- Decide (Branch):
  - Choose a variable & assign some value (decision)
  - Basic mechanism to do search
  - Imposes dynamic variable order
  - Decision Level (DL): variable \( \Rightarrow \) natural number

- Conflict analysis and clause learning:
  - Analyze the reason (learn conflict clause)
  - Conflict clause blocks the non-satisfying & a large set of other 'no-good' assignments

- BackJump:
  - Undo the decision(s) that caused no-good assignment
  - Assign 'decision variables' different values
  - Go back several decision levels

Vijay Ganesh, Dagstuhl, Aug 8-12, 2011
Input SAT Instance

Propagate()
(BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

Return SAT

Decide()

TopLevel Conflict?

Return UNSAT

BackJump()

• Propagate (Boolean Constant Propagation):
  • Propagate inferences due to unit clauses
  • Most time in solving goes into this

• Detect Conflict?
  • Conflict: partial assignment is not satisfying

• Decide (Branch):
  • Choose a variable & assign some value (decision)
  • Basic mechanism to do search
  • Imposes dynamic variable order
  • Decision Level (DL): variable \( \Rightarrow \) natural number

Marques-Silva & Sakallah (1999)

Backtrack (Davis, Putnam, Loveland, Logemann 1962)
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

Input SAT Instance

- Propagate():
  - Propagate inferences due to unit clauses
  - Most time in solving goes into this

- Detect Conflict:
  - Conflict: partial assignment is not satisfying

- Decide (Branch):
  - Choose a variable & assign some value (decision)
  - Each decision is a decision level
  - Imposes dynamic variable order
  - Decision Level (DL): variable ⇒ natural number

- Conflict analysis and clause learning:
  - Compute assignments that lead to conflict (analysis)
  - Construct conflict clause blocks the non-satisfying & a large set of other ‘no-good’ assignments (learning)
  - Marques-Silva & Sakallah (1996)
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

- **Propagate:**
  - Propagate inferences due to unit clauses
  - Most time in solving goes into this

- **Detect Conflict?**
  - Conflict: partial assignment is not satisfying

- **Decide (Branch):**
  - Choose a variable & assign some value *(decision)*
  - Each decision is a decision level
  - Imposes dynamic variable order
  - Decision Level *(DL)*: variable $\Rightarrow$ natural number

- **Conflict analysis and clause learning:**
  - Compute assignments that lead to conflict *(analysis)*
  - Construct conflict clause blocks the non-satisfying & a large set of other ‘no-good’ assignments *(learning)*
  - Marques-Silva & Sakallah (1996)
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

Input SAT Instance

Propagate() (BCP)

No Conflict?

All Vars Assigned?

Return SAT

Conflict Analysis()

Decide()

TopLevel Conflict?

Return UNSAT

BackJump()
Modern CDCL SAT Solver Architecture

**Propagate(), Decide(), Analyze/Learn(), BackJump()**

- **Input SAT Instance**

**Propagate()**

- (BCP)
- No Conflict?
- Propagate inferences due to unit clauses
- Most time in solving goes into this

**Detect Conflict?**

- Conflict: partial assignment is not satisfying

**Decide()**

- Choose a variable & assign some value (decision)
- Each decision is a decision level
- Imposes dynamic variable order
- Decision Level (DL): variable ⇒ natural number

**Conflict Analysis()**

- Compute assignments that lead to conflict (analysis)
- Construct conflict clause blocks the non-satisfying & a large set of other ‘no-good’ assignments (learning)
- Marques-Silva & Sakallah (1996)

- **Conflict-driven BackJump**
  - Undo the decision(s) that caused no-good assignment
  - Assign ‘decision variables’ different values
  - Go back several decision levels
  - Backjump: Marques-Silva, Sakallah (1999)
  - Backtrack: Davis, Putnam, Loveland, Logemann (1962)

- **BackJump()**

- **Return UNSAT**

- **Return SAT**
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

Input SAT Instance

Propagate()

(BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

Return SAT

Decide()

TopLevel Conflict?

Return UNSAT

BackJump()

{3, 6, -7, 8}
{1, 4, 7}
{-8, 4}
{-1, -3, 8}
{-3, -4, -8}
{-1, -2, 3, 4, -6}

Unit clause (BCP)

Decide

{3, 6, -7, 8}
{1, 4, 7}
{-8, 4}
{-1, -3, 8}
{-3, -4, -8}
{-1, -2, 3, 4, -6}

Another unit clause (more BCP)

CONFLICT!
(Trigger to analyze & backjump)

{3, 6, -7, 8}
{1, 4, 7}
{-8, 4}
{-1, -3, 8}
{-3, -4, -8}
{-1, -2, 3, 4, -6}
**Modern CDCL SAT Solver Architecture**

**Decide() Details: VSIDS Heuristic**

- **Input SAT Instance**

  - Propagate() (BCP)

  - **No Conflict?**

  - All Vars Assigned?

  - Conflict Analysis()

  - **TopLevel Conflict?**

    - **Decide()**

    - Return SAT

    - Return UNSAT

    - BackJump()

- **Decide() or Branching():**
  - Choose a variable & assign some value (decision)
  - Imposes dynamic variable order (Malik et al. 2001)

- **How to choose a variable:**
  - VSIDS heuristics
  - Each variable has an activity
  - Activity is bumped additively, if variable occurs in conflict clause
  - Activity of all variables is decayed by multiplying by const < 1
  - Next decision variable is the variable with highest activity
  - Over time, truly important variables get high activity
  - This is pure magic, and seems to work for many problems
Modern CDCL SAT Solver Architecture

Propagate() Details: Two-watched Literal Scheme

Input SAT Instance → **Propagate() (BCP)** → No Conflict?

- All Vars Assigned?
  - Conflict Analysis()
  - **Decide()**
  - **Return SAT**

- TopLevel Conflict?
  - **BackJump()**
  - **Return UNSAT**

- **Unit clause (BCP)**

```
{3, 6, -7, 8}
{1, 4, 7}
{-8, 4}
{-1, -3, 8}
{-3, -4, -8}
{-1, -2, 3, 4, -6}

{3, 6, -7, 8}
{1, 4, 7}
{-8, 4}
{-1, -3, 8}
{-3, -4, -8}
{-1, -2, 3, 4, -6}
```

Notice: The constraint propagates 8

---

**Table:**

<table>
<thead>
<tr>
<th>Watched Literal</th>
<th>Watcher List</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>{-1, -3, 8},...</td>
</tr>
<tr>
<td>-3</td>
<td>{-1, -3, 8},...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Watched Literal</th>
<th>Watcher List</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>{-1, -3, 8},...</td>
</tr>
<tr>
<td>-3</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>{-1, -3, 8},...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

Input SAT Instance

Propagate() (BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

Return SAT

Decide()

TopLevel Conflict?

BackJump()

Return UNSAT

Decide

{3, 6, -7, 8}
{1, 4, 7}
{-8, 4}
{-1, -3, 8}
{-3, -4, -8}
{-1, -2, 3, 4, -6}

{3, 6, -7, 8}
{1, 4, 7}
{-8, 4}
{-1, -3, 8}
{-3, -4, -8}
{-1, -2, 3, 4, -6}

Another unit clause (more BCP)

CONFLICT! (Trigger to analyze & backjump)

Basic Backtracking Search

- Flip the last decision
- Try setting 1 to False
- Highly inefficient
- No learning from mistakes
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details

Input SAT Instance → Propagate() (BCP)

No Conflict?

All Vars Assigned?

Conflicting Analysis()

Return SAT

Decide()

TopLevel Conflict?

Return UNSAT

BackJump()

Some Definitions

- **Decision Level (DL)**
  - Map from Boolean variables in input to natural numbers
  - All unit clauses in input & resultant propagations get DL = 0
  - Every decision var gets a DL in increasing order >= 1
  - All propagations due to decision var at DL=x get the DL=x

- **Conflict Graph (CG) or Implication Graph**
  - Directed Graph that records decisions & propagations
  - Vertices: literals, Edge: unit clauses

- **Conflict Clause (CC)**
  - Clause returned by Conflict Analysis(), added to conflict DB
  - Implied by the input formula
  - A cut in the CG
  - Prunes the search

- **Assignment Trail (AT)**
  - A stack of partial assignment to literals, with DL info
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{ X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots \} \\
Current decision: \{ X_1 = 1@6 \}

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}

\begin{align*}
X_9 &= 0@1 \\
X_{10} &= 0@3 \\
X_{11} &= 0@3
\end{align*}
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, ...\}

Current decision: \{X_1 = 1@6\}

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

Clause DB

\[ W_1 = (\neg X_1 + X_2) \]
\[ W_2 = (\neg X_1 + X_3 + X_9) \]
\[ W_3 = (\neg X_2 + \neg X_3 + X_4) \]
\[ W_4 = (\neg X_4 + X_5 + X_{10}) \]
\[ W_5 = (\neg X_4 + X_6 + X_{11}) \]
\[ W_6 = (\neg X_5 + \neg X_6) \]
\[ W_7 = (X_1 + X_7 + \neg X_{12}) \]
\[ W_8 = (X_1 + X_8) \]
\[ W_9 = (\neg X_7 + \neg X_8 + \neg X_{13}) \]
Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

Clause DB

\[ W_1 = (\neg X_1 + X_2) \]
\[ W_2 = (\neg X_1 + X_3 + X_9) \]
\[ W_3 = (\neg X_2 + \neg X_3 + X_4) \]
\[ W_4 = (\neg X_4 + X_5 + X_{10}) \]
\[ W_5 = (\neg X_4 + X_6 + X_{11}) \]
\[ W_6 = (\neg X_5 + \neg X_6) \]
\[ W_7 = (X_1 + X_7 + \neg X_{12}) \]
\[ W_8 = (X_1 + X_8) \]
\[ W_9 = (\neg X_7 + \neg X_8 + \neg X_{13}) \]
Modern CDCL SAT Solver Architecture
Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}
Current decision: \{X_1 = 1@6\}

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_10) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

<table>
<thead>
<tr>
<th>Clause DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>W_1 = (\neg X_1 + X_2)</td>
</tr>
<tr>
<td>W_2 = (\neg X_1 + X_3 + X_9)</td>
</tr>
<tr>
<td>W_3 = (\neg X_2 + \neg X_3 + X_4)</td>
</tr>
<tr>
<td>W_4 = (\neg X_4 + X_5 + X_{10})</td>
</tr>
<tr>
<td>W_5 = (\neg X_4 + X_6 + X_{11})</td>
</tr>
<tr>
<td>W_6 = (\neg X_5 + \neg X_6)</td>
</tr>
<tr>
<td>W_7 = (X_1 + X_7 + \neg X_{12})</td>
</tr>
<tr>
<td>W_8 = (X_1 + X_8)</td>
</tr>
<tr>
<td>W_9 = (\neg X_7 + \neg X_8 + \neg X_{13})</td>
</tr>
</tbody>
</table>

\[X_1 = 1@6\]
\[W_2\]
\[X_9 = 0@1\]
\[W_2\]
\[X_{10} = 0@3\]
\[W_1\]
\[X_2 = 1@6\]
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, ...\}

Current decision: \{X_1 = 1@6\}

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}

\begin{figure}
\centering
\begin{tikzpicture}
  \node[fill=green!50] (X1) at (0,0) {$X_1 = 1@6$};
  \node[fill=green!50] (X2) at (1,1) {$X_2 = 1@6$};
  \node[fill=green!50] (X3) at (2,2) {$X_3 = 1@6$};
  \node[fill=green!50] (X9) at (-2,0) {$X_9 = 0@1$};
  \node[fill=green!50] (X10) at (2,3) {$X_{10} = 0@3$};
  \node[fill=green!50] (X11) at (3,4) {$X_{11} = 0@3$};

  \draw[->, thick] (X1) -- (X2);
  \draw[->, thick] (X2) -- (X3);
  \draw[->, thick] (X1) -- (X9);
  \draw[->, thick] (X9) -- (X2);
  \draw[->, thick] (X3) -- (X10);
  \draw[->, thick] (X10) -- (X11);
\end{tikzpicture}
\end{figure}
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \( \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots \} \)

Current decision: \( \{X_1 = 1@6\} \)

Clause DB

- \( W_1 = (\neg X_1 + X_2) \)
- \( W_2 = (\neg X_1 + X_3 + X_9) \)
- \( W_3 = (\neg X_2 + \neg X_3 + X_4) \)
- \( W_4 = (\neg X_4 + X_5 + X_{10}) \)
- \( W_5 = (\neg X_4 + X_6 + X_{11}) \)
- \( W_6 = (\neg X_5 + \neg X_6) \)
- \( W_7 = (X_1 + X_7 + \neg X_{12}) \)
- \( W_8 = (X_1 + X_8) \)
- \( W_9 = (\neg X_7 + \neg X_8 + \neg X_{13}) \)
Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

Clause DB

\[W_1 = (\neg X_1 + X_2)\]
\[W_2 = (\neg X_1 + X_3 + X_9)\]
\[W_3 = (\neg X_2 + \neg X_3 + X_4)\]
\[W_4 = (\neg X_4 + X_5 + X_{10})\]
\[W_5 = (\neg X_4 + X_6 + X_{11})\]
\[W_6 = (\neg X_5 + \neg X_6)\]
\[W_7 = (X_1 + X_7 + \neg X_{12})\]
\[W_8 = (X_1 + X_8)\]
\[W_9 = (\neg X_7 + \neg X_8 + \neg X_{13})\]
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

Clause DB

- \(W_1 = \neg X_1 + X_2\)
- \(W_2 = \neg X_1 + X_3 + X_9\)
- \(W_3 = \neg X_2 + \neg X_3 + X_4\)
- \(W_4 = \neg X_4 + X_5 + X_{10}\)
- \(W_5 = \neg X_4 + X_6 + X_{11}\)
- \(W_6 = \neg X_5 + \neg X_6\)
- \(W_7 = X_1 + X_7 + \neg X_{12}\)
- \(W_8 = (X_1 + X_8)\)
- \(W_9 = (\neg X_7 + \neg X_8 + \neg X_{13})\)
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, ...\}

Current decision: \{X_1 = 1@6\}

Clause DB

- \(W_1 = (\neg X_1 + X_2)\)
- \(W_2 = (\neg X_1 + X_3 + X_9)\)
- \(W_3 = (\neg X_2 + \neg X_3 + X_4)\)
- \(W_4 = (\neg X_4 + X_5 + X_{10})\)
- \(W_5 = (\neg X_4 + X_6 + X_{11})\)
- \(W_6 = (\neg X_5 + \neg X_6)\)
- \(W_7 = (X_1 + X_7 + \neg X_{12})\)
- \(W_8 = (X_1 + X_8)\)
- \(W_9 = (\neg X_7 + \neg X_8 + \neg X_{13})\)
Current Assignment Trail: \(\{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}\)

Current decision: \(\{X_1 = 1@6\}\)

Clause DB

\[
\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
\]
Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
**Modern CDCL SAT Solver Architecture**

**Conflict Analysis/Learn() Details: Implication Graph**

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, ...\}

Current decision: \{X_1 = 1@6\}

**Clause DB**

\[
\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
\]
Modern CDCL SAT Solver Architecture
Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, ...\}

Current decision: \{X_1 = 1@6\}

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}

Clause DB Diagram
Conflict Analysis/Learn() Details: Implication Graph

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\} \\
Current decision: \{X_1 = 1@6\}

**Clause DB**

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
**Modern CDCL SAT Solver Architecture**

**Conflict Analysis/Learn() Details: Implication Graph**

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, ...\}

Current decision: \{X_1 = 1@6\}

<table>
<thead>
<tr>
<th>Clause DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>W_1 = (¬X_1 + X_2)</td>
</tr>
<tr>
<td>W_2 = (¬X_1 + X_3 + X_9)</td>
</tr>
<tr>
<td>W_3 = (¬X_2 + ¬X_3 + X_4)</td>
</tr>
<tr>
<td>W_4 = (¬X_4 + X_5 + X_{10})</td>
</tr>
<tr>
<td>W_5 = (¬X_4 + X_6 + X_{11})</td>
</tr>
<tr>
<td>W_6 = (¬X_5 + ¬X_6)</td>
</tr>
<tr>
<td>W_7 = (X_1 + X_7 + ¬X_{12})</td>
</tr>
<tr>
<td>W_8 = (X_1 + X_8)</td>
</tr>
<tr>
<td>W_9 = (¬X_7 + ¬X_8 + ¬X_{13})</td>
</tr>
</tbody>
</table>
Current Assignment Trail: \(\{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@3, \ldots\}\)

Current decision: \(\{X_1 = 1@6\}\)

**Clause DB**

\[
\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}
\]
Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}\n
Current Decision: \{X_1 = 1@6\}

Simplest strategy is to traverse the conflict graph backwards until decision variables:
conflict clause includes only decision variables (¬X_1 + X_9 + X_{10} + X_{11})

<table>
<thead>
<tr>
<th>Clause DB</th>
<th>CONFLICT GRAPH</th>
</tr>
</thead>
<tbody>
<tr>
<td>W_1 = (¬X_1 + X_2)</td>
<td><img src="image" alt="Conflict Graph" /></td>
</tr>
<tr>
<td>W_2 = (¬X_1 + X_3 + X_9)</td>
<td></td>
</tr>
<tr>
<td>W_3 = (¬X_2 + ¬X_3 + X_4)</td>
<td></td>
</tr>
<tr>
<td>W_4 = (¬X_4 + X_5 + X_{10})</td>
<td></td>
</tr>
<tr>
<td>W_5 = (¬X_4 + X_6 + X_{11})</td>
<td></td>
</tr>
<tr>
<td>W_6 = (¬X_5 + ¬X_6)</td>
<td></td>
</tr>
<tr>
<td>W_7 = (X_1 + X_7 + ¬X_{12})</td>
<td></td>
</tr>
<tr>
<td>W_8 = (X_1 + X_8)</td>
<td></td>
</tr>
<tr>
<td>W_9 = (¬X_7 + ¬X_8 + ¬X_{13})</td>
<td></td>
</tr>
</tbody>
</table>
Modern CDCL SAT Solver Architecture

Conflict Analysis/Learn() Details: Conflict Clause

Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current Decision: \{X_1 = 1@6\}

Another strategy is to use First Unique Implicant Point (UIP):
Traverse graph backwards in breadth-first, expand literals of conflict, stop at first UIP

Clause DB

\begin{align*}
W_1 &= (\neg X_1 + X_2) \\
W_2 &= (\neg X_1 + X_3 + X_9) \\
W_3 &= (\neg X_2 + \neg X_3 + X_4) \\
W_4 &= (\neg X_4 + X_5 + X_{10}) \\
W_5 &= (\neg X_4 + X_6 + X_{11}) \\
W_6 &= (\neg X_5 + \neg X_6) \\
W_7 &= (X_1 + X_7 + \neg X_{12}) \\
W_8 &= (X_1 + X_8) \\
W_9 &= (\neg X_7 + \neg X_8 + \neg X_{13})
\end{align*}

CONFLICT GRAPH

\begin{align*}
X_1 &= 1@6 \\
X_2 &= 1@6 \\
X_3 &= 1@6 \\
X_4 &= 1@6 \\
X_5 &= 1@6 \\
X_6 &= 1@6 \\
X_7 &= 0@1 \\
X_8 &= 0@3 \\
X_9 &= 0@1 \\
X_{10} &= 0@3 \\
X_{11} &= 0@3 \\
\end{align*}
Current Assignment Trail: \{X_9 = 0@1, X_{10} = 0@3, X_{11} = 0@3, X_{12} = 1@2, X_{13} = 1@2, \ldots\}

Current decision: \{X_1 = 1@6\}

Strategy: Closest decision level (DL) ≤ current DL for which conflict clause is unit. Undo \{X_1 = 1@6\}
Conflict clause: \((X_9 + X_{10} + X_{11} + \neg X_{12} + \neg X_{13})\)

BackJump strategy: Closest decision level (DL) \(\leq\) current DL for which conflict clause is unit. Undo \(\{X_{10} = 0@3\}\)

<table>
<thead>
<tr>
<th>CLAUSE DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>(W_1 = \neg X_1 + X_2)</td>
</tr>
<tr>
<td>(W_2 = \neg X_1 + X_3 + X_9)</td>
</tr>
<tr>
<td>(W_3 = \neg X_2 + \neg X_3 + X_4)</td>
</tr>
<tr>
<td>(W_4 = \neg X_4 + X_5 + X_{10})</td>
</tr>
<tr>
<td>(W_5 = \neg X_4 + X_6 + X_{11})</td>
</tr>
<tr>
<td>(W_6 = \neg X_5 + \neg X_6)</td>
</tr>
<tr>
<td>(W_7 = X_1 + X_7 + \neg X_{12})</td>
</tr>
<tr>
<td>(W_8 = X_1 + X_8)</td>
</tr>
<tr>
<td>(W_9 = \neg X_7 + \neg X_8 + \neg X_{13})</td>
</tr>
</tbody>
</table>

| CONFLICT GRAPH |

<table>
<thead>
<tr>
<th>TRAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>BackJump</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
Modern CDCL SAT Solver Architecture

Restarts and Forget

Input SAT Instance

Propagate() (BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

Return SAT

Decide()

TopLevel Conflict?

Return UNSAT

BackJump()

- Restarts
  - Clear the Trail and start again
  - Start searching with a different variable order
  - Only Conflict Clause (CC) database is retained

- Forget: throw away less active learnt conflict clauses routinely
  - Routinely throw away very large CC
  - Logically CC are implied
  - Hence no loss in soundness/completeness
  - Time Savings: smaller DB means less work in propagation
  - Space savings
Modern CDCL SAT Solver Architecture

Why is SAT efficient?

- VSIDS branching heuristic and propagate (BCP)
- Conflict-Driven Clause-Learning (CDCL)
- Forget conflict clauses if DB goes too big
- BackJump
- Restarts
- All the above elements are needed for efficiency
- Deeper understanding lacking
- No predictive theory

Input SAT Instance

Propagate() (BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

TopLevel Conflict?

Return SAT

Decide()

Return UNSAT

BackJump()
Modern CDCL SAT Solver Architecture

Propagate(), Decide(), Analyze/Learn(), BackJump()

Input SAT Instance

Propagate() (BCP)

No Conflict?

All Vars Assigned?

Conflict Analysis()

Conflict

Return SAT

Decide()

TopLevel Conflict?

Return UNSAT

BackJump()

- Conflict-Driven Clause-Learning (CDCL) (Marques-Silva & Sakallah 1996)
- Decide/branch and propagate (BCP) (Malik et al. 2001, Zabih & McAllester 1988)
- BackJump (McAllester 1980, Marques-Silva & Sakallah 1999)
- Restarts (Selman & Gomes 2001)
- Follows MiniSAT (Een & Sorensson 2003)
A solver is said to be sound, if, for any input formula F, the solver terminates and produces a solution, then F is indeed SAT.

**Proof:** (Easy) SAT is returned only when all vars have been assigned a value (True, False) by Decide or BCP, and solver checks the solution.
Completeness: A solver is said to be complete, if, for any input formula F that is SAT, the solver terminates and produces a solution (i.e., solver does not miss solutions)

Proof: (Harder)
- Backtracking + BCP + decide is complete (easy)
- Conflict clause is implied by input formula (easy)
- Only need to see backjumping does not skip assignments
  - Observe backjumping occurs only when conflict clause (CC) vars < decision level (DL) of conflicting var
  - Backjumping to max(DL of vars in CC)
  - Decision tree rooted at max(DL of vars in CC)+1 is guaranteed to not satisfy CC
  - Hence, backjumping will not skip assignments
Termination: Some measure decreases every iteration

Proof Sketch:

- Loop guarantees either conflict clause (CC) added OR assign extended

- CC added. What stops CC addition looping forever?
  - Recall that CC is remembered
  - No CC duplication possible
  - CC blocks UNSAT assign exploration in decision tree. No duplicate UNSAT assign exploration possible
  - Size of decision tree explored decreases for each CC add
References & Important SAT Solvers


7. zChaff SAT Solver by Lintao Zhang 2002.


9. MiniSAT Solver by Niklas Een and Niklas Sorenson 2005 - present

10. SAT Live: http://www.satlive.org/

11. SAT Competition: http://www.satcompetition.org/

Modern CDCL SAT Solver Architecture

Important Ideas and Conclusions

1. SAT solvers are crucial for software engineering

2. Huge impact in formal methods, program analysis and testing

3. Key ideas that make SAT efficient
   1. Conflict-driven clause learning
   2. VSIDS (or similar) variable selection heuristics
   3. Backjumping
   4. Restarts

4. Techniques I didn’t discuss
   1. Survey propagation (belief propagation) by Selman & Gomes
   2. Works well for randomized SAT, not yet for industrial instances
   3. Physics-inspired
   4. Combining CDCL with survey propagation (?)