SAT Solvers
A Brief Introduction

Marcelo Finger

Department of Computer Science
Instituto de Matemática e Estatística
Universidade de São Paulo
TOPICS

1. THE PROBLEM
2. A BRIEF HISTORY OF SAT SOLVERS
3. THE DPLL ALGORITHM
4. DPLL AND RESOLUTION
5. WATCHED LITERALS
6. CONCLUSION
SAT is a central problem in Computer Science, with both theoretical and practical interests.
SAT is a central problem in Computer Science, with both theoretical and practical interests.

SAT was the 1st NP-complete problem.
The Problem

The Centrality of SAT

- SAT is a central problem in Computer Science, with both theoretical and practical interests
- SAT was the 1st NP-complete problem
- SAT received a lot of attention [1960-now]
The Problem
The Centrality of SAT

- SAT is a central problem in Computer Science, with both theoretical and practical interests
- SAT was the 1st NP-complete problem
- SAT received a lot of attention [1960-now]
- SAT has very efficient implementations
SAT is a central problem in Computer Science, with both theoretical and practical interests.

- SAT was the 1st NP-complete problem
- SAT received a lot of attention [1960-now]
- SAT has very efficient implementations
- SAT has become the “assembly language” of hard-problems
The Problem
The Centrality of SAT

- SAT is a central problem in Computer Science, with both theoretical and practical interests
- SAT was the 1st NP-complete problem
- SAT received a lot of attention [1960-now]
- SAT has very efficient implementations
- SAT has become the “assembly language” of hard-problems
- SAT is logic
The Setting: the language

- Atoms: $\mathcal{P} = \{p_1, \ldots, p_n\}$
- Literals: $p_i$ and $\neg p_j$
- $\bar{p} = \neg p$, $\overline{\neg p} = p$
- A clause is a set of literals. Ex: $\{p, \bar{q}, r\}$ or $p \lor \bar{q} \lor r$
- A formula $C$ is a set of clauses
The Setting: semantics

- Valuation for atoms $\nu : \mathcal{P} \rightarrow \{0, 1\}$
The Setting: semantics

- Valuation for atoms $\nu : \mathcal{P} \rightarrow \{0, 1\}$
- An atom $p$ is satisfied if $\nu(p) = 1$
Valuation for atoms $\nu : \mathcal{P} \rightarrow \{0, 1\}$

- An atom $p$ is satisfied if $\nu(p) = 1$

- Valuations are extended to all formulas
**The Setting: Semantics**

- Valuation for atoms $\nu : \mathcal{P} \to \{0, 1\}$
- An atom $p$ is **satisfied** if $\nu(p) = 1$
- Valuations are extended to all formulas
- $\nu(\bar{\lambda}) = 1 \iff \nu(\lambda) = 0$
The Setting: semantics

- Valuation for atoms $\nu : \mathcal{P} \rightarrow \{0, 1\}$
- An atom $p$ is satisfied if $\nu(p) = 1$
- Valuations are extended to all formulas
  - $\nu(\overline{\lambda}) = 1 \iff \nu(\lambda) = 0$
- A clause $c$ is satisfied ($\nu(c) = 1$) if some literal $\lambda \in c$ is satisfied
The Setting: semantics

- Valuation for atoms $\nu: \mathcal{P} \rightarrow \{0, 1\}$
- An atom $p$ is satisfied if $\nu(p) = 1$
- Valuations are extended to all formulas
  - $\nu(\bar{\lambda}) = 1 \iff \nu(\lambda) = 0$
- A clause $c$ is satisfied ($\nu(c) = 1$) if some literal $\lambda \in c$ is satisfied
- A formula $C$ is satisfied ($\nu(C) = 1$) if all clauses in $C$ are satisfied
THE PROBLEM

- A formula $C$ is **satisfiable** if exits $v$, $v(C) = 1$.
- Otherwise, $C$ is **unsatisfiable**
A formula \( C \) is **satisfiable** if exists \( \nu, \nu(C) = 1 \).

Otherwise, \( C \) is **unsatisfiable**.
An NP Algorithm for SAT

**NP-SAT(C)**

**INPUT:** \(C\), a formula in clausal form

**OUTPUT:** \(v\), if \(v(C) = 1\); no, otherwise.

1: Guess a \(v\)
2: Show, in polynomial time, that \(v(C) = 1\)
3: return \(v\)
4: if no such \(v\) is guessable then
5: return no
6: end if
# A Naïve SAT Solver

**NaiveSAT**($C$)

<table>
<thead>
<tr>
<th>Input:</th>
<th>$C$, a formula in clausal form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>$v$, if $v(C) = 1$; no, otherwise.</td>
</tr>
</tbody>
</table>

1: **for** every valuation $v$ over $p_1, \ldots, p_n$ **do**
2: **if** $v(C) = 1$ **then**
3: **return** $v$
4: **end if**
5: **end for**
6: **return** no
A BRIEF HISTORY OF SAT SOLVERS

[Davis & Putnam, 1960; Davis, Longemann & Loveland, 1962] The DPLL Algorithm, a complete SAT Solver
A Brief History of SAT Solvers

- [Davis & Putnam, 1960; Davis, Longemann & Loveland, 1962] The DPLL Algorithm, a complete SAT Solver
- [Tseitin, 1966] DPLL has exponential lower bound
A Brief History of SAT Solvers

- [Davis & Putnam, 1960; Davis, Longemann & Loveland, 1962] The DPLL Algorithm, a complete SAT Solver
- [Tseitin, 1966] DPLL has exponential lower bound
- [Cook 1971] SAT is NP-complete
Incomplete SAT methods

Incomplete methods compute valuation if $C$ is SAT; if $C$ is unSAT, no answer.

- [Selman, Levesque & Mitchell, 1992] GSAT, a local search algorithm for SAT
Incomplete SAT methods compute valuation if $C$ is SAT; if $C$ is unSAT, no answer.

- [Selman, Levesque & Mitchell, 1992] GSAT, a local search algorithm for SAT
Incomplete SAT methods compute valuation if $C$ is SAT; if $C$ is unSAT, no answer.

- [Selman, Levesque & Mitchell, 1992] GSAT, a local search algorithm for SAT
- [Kautz & Selman, 1992] SAT planning
Incomplete SAT methods

Incomplete methods compute valuation if $C$ is SAT; if $C$ is unSAT, no answer.

- [Selman, Levesque & Mitchell, 1992] GSAT, a local search algorithm for SAT
- [Kautz & Selman, 1992] SAT planning
Incomplete SAT methods compute valuation if \( C \) is SAT; if \( C \) is unSAT, no answer.

- [Selman, Levesque & Mitchell, 1992] GSAT, a local search algorithm for SAT
- [Kautz & Selman, 1992] SAT planning
- [Gent & Walsh, 1994] SAT phase transition
Incomplete SAT methods compute valuation if $C$ is SAT; if $C$ is unSAT, no answer.

- [Selman, Levesque & Mitchell, 1992] GSAT, a local search algorithm for SAT
- [Kautz & Selman, 1992] SAT planning
- [Gent & Walsh, 1994] SAT phase transition
DPLL: Second Generation

- SAT competitions since 2002:
  http://www.satcompetition.org/
DPLL: Second Generation

- SAT competitions since 2002:
  - http://www.satcompetition.org/
- Aggregation of several techniques to SAT, such as learning, unlearning, backjumping, watched literal, special heuristics.
DPLL: Second Generation

- SAT competitions since 2002:
  - http://www.satcompetition.org/
- Aggregation of several techniques to SAT, such as learning, unlearning, backjumping, watched literal, special heuristics.
- Very competitive SAT solvers: Chaff [2001], BerkMin [2002], zChaff [2004].
DPLL: Second Generation

- SAT competitions since 2002:
  http://www.satcompetition.org/
- Aggregation of several techniques to SAT, such as learning, unlearning, backjumping, watched literal, special heuristics.
- Very competitive SAT solvers: Chaff [2001], BerkMin [2002], zChaff [2004].
- Applications to planning, microprocessor test and verification, software design and verification, AI search, games, etc.
DPLL: Second Generation

- SAT competitions since 2002: http://www.satcompetition.org/
- Aggregation of several techniques to SAT, such as learning, unlearning, backjumping, watched literal, special heuristics.
- Very competitive SAT solvers: Chaff [2001], BerkMin [2002], zChaff [2004].
- Applications to planning, microprocessor test and verification, software design and verification, AI search, games, etc.
- Some non-DPLL SAT solvers incorporate all those techniques: [Dixon 2004]
DPLL Through Examples

\[ p \lor q \]
\[ p \lor \bar{q} \]
\[ \bar{p} \lor t \lor s \]
\[ \bar{p} \lor \bar{t} \lor s \]
\[ \bar{p} \lor \bar{s} \]
\[ \bar{p} \lor s \lor \bar{a} \]

Marcelo Finger
IME-USP
SAT Solvers
Delete all clauses that contain $\lambda$, if $\overline{\lambda}$ does not occur.

\begin{align*}
p \lor q \\
p \lor \overline{q} \\
\overline{p} \lor t \lor s \\
\overline{p} \lor \overline{t} \lor s \\
\overline{p} \lor \overline{s} \\
\overline{p} \lor \overline{t} \lor \overline{s} \\
\overline{p} \lor \overline{q} \lor \overline{t} \lor \overline{s} \\
\overline{p} \lor \overline{q} \lor \overline{t} \lor \overline{s} \lor \overline{a}
\end{align*}
CONSTRUCTION OF A PARTIAL VALUATION

Choose a literal: $s$. $V = \{s\}$

Propagate choice: Delete clauses containing $s$. Delete $\overline{s}$ from other clauses.

\[
\begin{align*}
p \lor q \\
p \lor \overline{q} \\
\overline{p} \lor \overline{q} \lor s \\
\overline{p} \lor \overline{q} \lor \overline{s} \\
\overline{p} \lor \overline{s} \\
\end{align*}
\]
Unit Propagation

Enlarge the partial valuation with unit clauses.  
\[ V = \{ s, \bar{p} \} \]

Propagate unit clauses as before.

\[ \begin{align*}
\bar{p} & \implies q \\
\bar{p} & \implies \bar{q} \\
\bar{p} & 
\end{align*} \]

Another propagation step leads to  
\[ V = \{ s, \bar{p}, q, \bar{q} \} \]
Unit propagation may lead to contradictory valuation:
\[ V = \{s, \bar{p}, q, \bar{q}\} \]
Backtrack to the previous choice, and propagate: \[ V = \{\bar{s}\} \]
When propagation finishes, a new choice is made: \( p \).
\[ V = \{ \bar{s}, p \} . \]
This leads to an inconsistent valuation: \( V = \{ \bar{s}, p, t, \bar{t} \} \)
Backtrack to last choice: \( V = \{ \bar{s}, \bar{p} \} \)

Propagation leads to another contradiction: \( V = \{ \bar{s}, \bar{p}, q, \bar{q} \} \)
The Formula is UnSAT

There is nowhere to backtrack to now!
The formula is unsatisfiable, with a proof sketched below.
The Resolution Inference For Clauses

Usual Resolution

\[
\frac{C \lor \lambda \quad \bar{\lambda} \lor D}{C \lor D}
\]

Clauses as Sets

\[
\frac{\Gamma \cup \{\lambda\} \quad \{\bar{\lambda}\} \cup \Delta}{\Gamma \cup \Delta}
\]

Note that, as clauses are sets

\[
\frac{\Gamma \cup \{\mu, \lambda\} \quad \{\bar{\lambda}, \mu\} \cup \Delta}{\Gamma \cup \Delta \cup \{\mu\}}
\]
DPLL PROOFS AND RESOLUTION

\[ (\neg p \lor \neg s) \]
\[ (p \lor q) \]
\[ (p \lor \neg q) \]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
DPLL Proofs and Resolution

\[
\begin{align*}
s & \quad (\bar{p} \lor \bar{s}) \\
\bar{p} & \quad p \\
p & \quad (p \lor \bar{q}) \\
\bar{p} & \quad \bar{q} \\
\bar{q} & \quad (p \lor \bar{q}) \\
p & \quad (p \lor \bar{q}) \\
\bar{q} & \quad (p \lor \bar{q}) \\
\end{align*}
\]
DPLL PROOFS AND RESOLUTION

\[
\begin{align*}
\bar{s} \quad & p \\
p \quad & p \lor q \\
p \lor \bar{q} \\
\bar{p} \lor \bar{s} \\
p \lor \bar{q} \\
\bar{p} \lor \bar{s} \\
p \lor q \\
\bar{t} \lor \bar{q} \\
\bar{t} \lor \bar{q} \\
\times \\
\times \\
\end{align*}
\]
DPLL Proofs and Resolution
DPLL Proofs and Resolution

\[ \bar{s} \]
\[ p \]
\[ p \lor q \]
\[ p \lor \bar{q} \]
\[ \bar{p} \lor \bar{s} \]
\[ \bar{p} \lor s \]
\[ \bar{p} \lor \bar{t} \lor s \]
\[ \bar{p} \lor \bar{t} \lor \bar{s} \]
\[ \bar{p} \lor \bar{q} \]
\[ \bar{p} \lor q \]
\[ \bar{p} \lor \bar{t} \lor s \]
\[ \bar{p} \lor \bar{t} \lor \bar{s} \]
\[ \bar{p} \lor \bar{q} \]
\[ \bar{p} \lor q \]

Marcelo Finger

SAT Solvers
DPLL Proofs and Resolution

Marcelo Finger
IME-USP
SAT Solvers
**CONCLUSION**

- DPLL is *isomorphic* to (a restricted form of) resolution
CONCLUSION

- DPLL is *isomorphic* to (a restricted form of) resolution
- DPLL inherits all properties of this (restricted form of resolution)
CONCLUSION

- DPLL is isomorphic to (a restricted form of) resolution
- DPLL inherits all properties of this (restricted form of resolution)
- In particular, DPLL inherits the exponential lower bounds
Enhancing DPLL

For the reasons discussed, DPLL needs to be improved to achieve better efficiency. Several techniques have been applied:

- Learning
- Unlearning
- Backjumping
- Watched literals
- Heuristics for choosing literals
Enhancing DPLL

For the reasons discussed, DPLL needs to be improved to achieve better efficiency. Several techniques have been applied:

- Learning
- Unlearning
- Backjumping
- Watched literals
- Heuristics for choosing literals
Watched Literals
Empirical measures show that 80% of time DPLL is doing Unit Propagation.

Propagation is the main target for optimization.

**CHAFF** introduced the technique of **Watched Literals**:
- Unit Propagation speed up
- No need to delete literals or clauses
- No need to watch all literals in a clause
- Constant time backtracking (very fast)
DPLL AND 3-VALUED LOGIC

- DPLL underlying logic is 3-valued
- Given a partial valuation

\[ V = \{\lambda_1, \ldots, \lambda_k\} \]

- Let \( \lambda \) be any literal.

\[ V(\lambda) = \begin{cases} 
1 \text{ (true)} & \text{if } \lambda \in V \\
0 \text{ (false)} & \text{if } \lambda \notin V \\
* \text{ (undefined)} & \text{otherwise}
\end{cases} \]
The Watched Literal Data Structure

- Every clause \( c \) has two selected literals: \( \lambda_{c1}, \lambda_{c2} \)
- For each \( c \), \( \lambda_{c1}, \lambda_{c2} \) are dynamically chosen and varies with time
- \( \lambda_{c1}, \lambda_{c2} \) are properly watched under partial valuation \( V \) if:
  - they are both undefined; or
  - at least one of them is true
Dynamics of Watched Literals

- Initially, $V = \emptyset$
- A pair of watched literals is chosen for each clause. It is proper.
- Literal choice and unit propagation expand $V$
- One or both watched literals may be falsified
- If $\lambda_{c1}, \lambda_{c2}$ become improper then
  - The falsified watched literal is changed
- If no proper pair of watched literals can be found, two things may occur to alter $V$
  - Unit propagation ($V$ is expanded)
  - Backtracking ($V$ is reduced)
The Problem History DPLL Resolution WatchLit Conclusion

Example

<table>
<thead>
<tr>
<th>clause</th>
<th>$\lambda_{c1}$</th>
<th>$\lambda_{c2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p \lor q \lor r$</td>
<td>$p = \ast$</td>
<td>$q = \ast$</td>
</tr>
<tr>
<td>$p \lor \bar{q} \lor s$</td>
<td>$p = \ast$</td>
<td>$\bar{q} = \ast$</td>
</tr>
<tr>
<td>$p \lor r \lor \bar{s}$</td>
<td>$p = \ast$</td>
<td>$r = \ast$</td>
</tr>
</tbody>
</table>

Initially $V = \emptyset$

A pair of literals was elected for each clause

All are undefined, all pairs are proper
\( p \) IS CHosen

\[ V = \{ \overline{p} \} \]

All watched literals become \((0,*),\) improper

New literals are chosen to be watched

<table>
<thead>
<tr>
<th>clause</th>
<th>( \lambda_{c1} )</th>
<th>( \lambda_{c2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p \lor q \lor r )</td>
<td>( r = * )</td>
<td>( q = * )</td>
</tr>
<tr>
<td>( p \lor \overline{q} \lor s )</td>
<td>( s = * )</td>
<td>( \overline{q} = * )</td>
</tr>
<tr>
<td>( p \lor \overline{r} \lor \overline{s} )</td>
<td>( \overline{s} = * )</td>
<td>( r = * )</td>
</tr>
</tbody>
</table>
\( \vec{r} \) IS CHOSEN

\[ V = \{ \bar{p}, \bar{r} \} \]
WL in clauses 1,3 become improper
No other \(*\)- or 1-literal to be chosen
Unit propagation: \( q, \bar{s} \) become true

<table>
<thead>
<tr>
<th>clause</th>
<th>( \lambda_{c1} )</th>
<th>( \lambda_{c2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p \lor q \lor r )</td>
<td>( r = 0 )</td>
<td>( q = \neq 1 )</td>
</tr>
<tr>
<td>( p \lor \bar{q} \lor s )</td>
<td>( s = \neq )</td>
<td>( \bar{q} = \neq )</td>
</tr>
<tr>
<td>( p \lor \bar{r} \lor \bar{s} )</td>
<td>( \bar{s} = \neq 1 )</td>
<td>( r = 0 )</td>
</tr>
</tbody>
</table>
UNIT PROPAGATION LEADS TO BACKTRACKING

\[ V = \{ \overline{p}, \overline{r}, q, \overline{s} \} \]

WL in clause 2 becomes improper
No other *- or 1-literal to be chosen
No unit propagation is possible: clause 2 is false

<table>
<thead>
<tr>
<th>clause</th>
<th>( \lambda_{c1} )</th>
<th>( \lambda_{c2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p \lor q \lor r )</td>
<td>( r = 0 )</td>
<td>( q = 1 )</td>
</tr>
<tr>
<td>( p \lor \overline{q} \lor s )</td>
<td>( s = 0 )</td>
<td>( \overline{q} = 0 )</td>
</tr>
<tr>
<td>( p \lor r \lor \overline{s} )</td>
<td>( \overline{s} = 1 )</td>
<td>( r = 0 )</td>
</tr>
</tbody>
</table>
Fast Backtracking

$V$ is contracted to last choice point

$V = \{ \overline{p}, \overline{r}, q, \bar{s} \} \cup \{ \bar{p}, r \}$

<table>
<thead>
<tr>
<th>clause</th>
<th>$\lambda_{c1}$</th>
<th>$\lambda_{c2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p \lor q \lor r$</td>
<td>$r = 1$</td>
<td>$q = *$</td>
</tr>
<tr>
<td>$p \lor \bar{q} \lor s$</td>
<td>$s = *$</td>
<td>$\bar{q} = *$</td>
</tr>
<tr>
<td>$p \lor r \lor \bar{s}$</td>
<td>$\bar{s} = *$</td>
<td>$r = 1$</td>
</tr>
</tbody>
</table>

Only affected WLs had to be recomputed
No need to reestablish previous context from a stack of contexts
Very quick backtracking
CONCLUSION OF THE TALK

- DPLL is > 40 years old, but still the most used strategy for SAT solvers
CONCLUSION OF THE TALK

- DPLL is > 40 years old, but still the most used strategy for SAT solvers
- Use of smart techniques have improved DPLL’s performance: \( N = 15 \rightarrow N = 10\ 000 \)
DPLL is > 40 years old, but still the most used strategy for SAT solvers

- Use of smart techniques have improved DPLL’s performance: \( N = 15 \rightarrow N = 10000 \)
- There are still very hard formulas that make DPLL exponential
Conclusion of the Talk

- DPLL is > 40 years old, but still the most used strategy for SAT solvers
- Use of smart techniques have improved DPLL’s performance: \( N = 15 \rightarrow N = 10000 \)
- There are still very hard formulas that make DPLL exponential
- Experiments show that these formulas do occur in practice
**Conclusion of the Talk**

- DPLL is > 40 years old, but still the most used strategy for SAT solvers
- Use of smart techniques have improved DPLL’s performance: \( N = 15 \rightarrow N = 10000 \)
- There are still very hard formulas that make DPLL exponential
- Experiments show that these formulas do occur in practice
- The future of SAT solvers lies in non-DPLL, non-clausal methods
**Conclusion of the Talk**

- DPLL is > 40 years old, but still the most used strategy for SAT solvers
- Use of smart techniques have improved DPLL’s performance: $N = 15 \rightarrow N = 10\,000$
- There are still very hard formulas that make DPLL exponential
- Experiments show that these formulas do occur in practice
- The future of SAT solvers lies in non-DPLL, non-clausal methods
- But the techniques learned from DPLL are incorporated in new techniques