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

The Problem
The Centrality of SAT

- SAT is a central problem in Computer Science, with both theoretical and practical interests

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

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

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

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 $p \vee \bar{q} \vee r$
- A formula C is a set of clauses

The Setting: semantics

- Valuation for atoms $v : \mathcal{P} \longrightarrow \{0, 1\}$

The Setting: semantics

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

The Setting: semantics

- Valuation for atoms $v : \mathcal{P} \longrightarrow \{0, 1\}$
- An atom p is satisfied if $v(p) = 1$
- Valuations are extended to all formulas

The Setting: semantics

- Valuation for atoms $v : \mathcal{P} \longrightarrow \{0, 1\}$
- An atom p is satisfied if $v(p) = 1$
- Valuations are extended to all formulas
- $v(\bar{\lambda}) = 1 \Leftrightarrow v(\lambda) = 0$

The Setting: semantics

- Valuation for atoms $v : \mathcal{P} \longrightarrow \{0, 1\}$
- An atom p is satisfied if $v(p) = 1$
- Valuations are extended to all formulas
- $v(\bar{\lambda}) = 1 \Leftrightarrow v(\lambda) = 0$
- A clause c is satisfied ($v(c) = 1$) if some literal $\lambda \in c$ is satisfied

The Setting: semantics

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

The Problem

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

The Problem

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

The SAT Problem

Given a formula C, decide if C is satisfiable.

Witnesses: If C is satisfiable, provide a v such that $v(C) = 1$; otherwise, give a proof that 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)

Input: C, a formula in clausal form

Output: v, if $v(C) = 1$; no, otherwise.

for every valuation v over $p_1, \ldots, p_n$ **do**
    if $v(C) = 1$ **then**
        return v
    end if
end for
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

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
- [Mitchell, Levesque & Selman, 1992] Hard and easy SAT problems

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
- [Mitchell, Levesque & Selman, 1992] Hard and easy SAT problems
- [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
- [Mitchell, Levesque & Selman, 1992] Hard and easy SAT problems
- [Kautz & Selman, 1992] SAT planning
- [Kautz & Selman, 1993] WalkSAT Algorithm

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
- [Mitchell, Levesque & Selman, 1992] Hard and easy SAT problems
- [Kautz & Selman, 1992] SAT planning
- [Kautz & Selman, 1993] WalkSAT Algorithm
- [Gent & Walsh, 1994] SAT phase transition

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
- [Mitchell, Levesque & Selman, 1992] Hard and easy SAT problems
- [Kautz & Selman, 1992] SAT planning
- [Kautz & Selman, 1993] WalkSAT Algorithm
- [Gent & Walsh, 1994] SAT phase transition
- [Shang & Wah, 1998] Discrete Lagrangian Method (DLM)

DPLL: Second Generation

- Second Generation of DPLL SAT Solvers: Posit [1995], SATO [1997], GRASP [1999]. Heuristics but no learning.

DPLL: Second Generation

- Second Generation of DPLL SAT Solvers: Posit [1995], SATO [1997], GRASP [1999]. Heuristics but no learning.
- SAT competitions since 2002: `http://www.satcompetition.org/`

DPLL: Second Generation

- Second Generation of DPLL SAT Solvers: Posit [1995], SATO [1997], GRASP [1999]. Heuristics but no learning.
- 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

- Second Generation of DPLL SAT Solvers: Posit [1995], SATO [1997], GRASP [1999]. Heuristics but no learning.
- 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

- Second Generation of DPLL SAT Solvers: Posit [1995], SATO [1997], GRASP [1999]. Heuristics but no learning.
- 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

- Second Generation of DPLL SAT Solvers: Posit [1995], SATO [1997], GRASP [1999]. Heuristics but no learning.
- 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

$$p \vee q$$
$$p \vee \bar{q}$$
$$\bar{p} \vee t \vee s$$
$$\bar{p} \vee \bar{t} \vee s$$
$$\bar{p} \vee \bar{s}$$
$$\bar{p} \vee s \vee \bar{a}$$

Initial Simplifications

Delete all clauses that contain λ, if $\bar{\lambda}$ does not occur.

$$
\begin{aligned}
& p \vee q \\
& p \vee \bar{q} \\
& \bar{p} \vee t \vee s \\
& \bar{p} \vee \bar{t} \vee s \\
& \bar{p} \vee \bar{s} \\
& \cancel{\bar{p} \vee s \vee \bar{a}}
\end{aligned}
$$

Construction of a Partial Valuation

Choose a literal: s. $V = \{\mathbf{s}\}$
Propagate choice: Delete clauses containing s. Delete $\bar{s}$ from other clauses.

$p \vee q$

$p \vee \bar{q}$

~~$\bar{p} \vee t \vee s$~~

~~$\bar{p} \vee \bar{t} \vee s$~~

$\bar{p}$ ~~$\vee \bar{s}$~~

Unit Propagation

Enlarge the partial valuation with unit clauses.

$V = \{\mathbf{s}, \bar{p}\}$

Propagate unit clauses as before.

$$\cancel{p \vee}\ q$$

$$\cancel{p \vee}\ \bar{q}$$

$$\cancel{\bar{p}}$$

Another propagation step leads to $V = \{\mathbf{s}, \bar{p}, q, \bar{q}\}$

Backtracking

Unit propagation may lead to contradictory valuation:
$V = \{\mathbf{s}, \bar{p}, q, \bar{q}\}$
Backtrack to the previous choice, and propagate: $V = \{\bar{s}\}$

$p \vee q$
$p \vee \bar{q}$
$\bar{p} \vee t$ ~~$\vee s$~~
$\bar{p} \vee \bar{t}$ ~~$\vee s$~~
~~$\bar{p} \vee \bar{s}$~~

New Choice

When propagation finishes, a new choice is made: p.

$V = \{\bar{s}, \mathbf{p}\}$.

This leads to an inconsistent valuation: $V = \{\bar{s}, \mathbf{p}, t, \bar{t}\}$

Backtrack to last choice: $V = \{\bar{s}, \bar{p}\}$

$$\not{p} \not\vee q$$

$$\not{p} \not\vee \bar{q}$$

$$\not{\bar{p}} \not\vee \not{t}$$

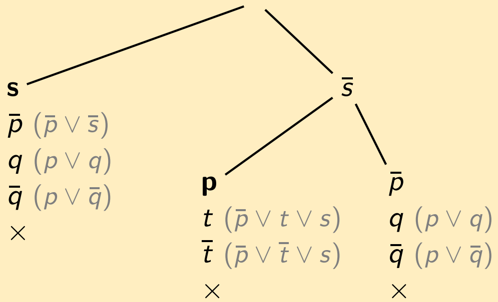
$$\not{\bar{p}} \not\vee \not{\bar{t}}$$

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

$$\frac{C \vee \lambda \quad \bar{\lambda} \vee D}{C \vee D}$$

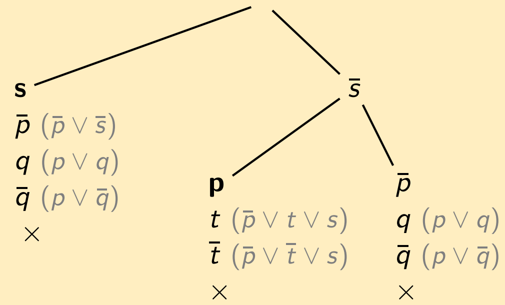
Clauses as Sets

$$\frac{\Gamma \cup \{\lambda\} \qquad \{\bar{\lambda}\} \cup \Delta}{\Gamma \cup \Delta}$$

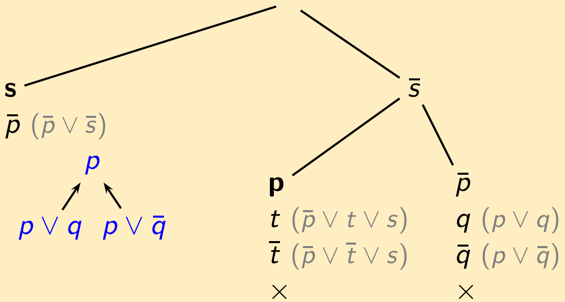
Note that, as clauses are sets

$$\frac{\Gamma \cup \{\mu, \lambda\} \qquad \{\bar{\lambda}, \mu\} \cup \Delta}{\Gamma \cup \Delta \cup \{\mu\}}$$

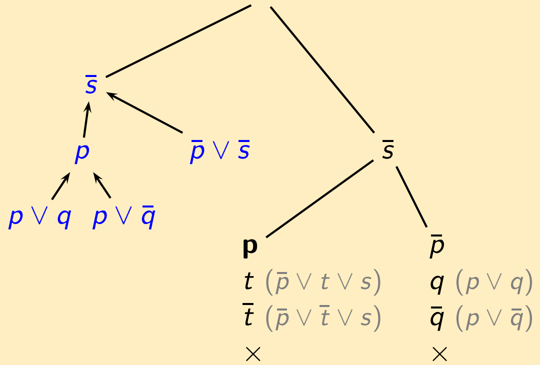
DPLL Proofs and Resolution



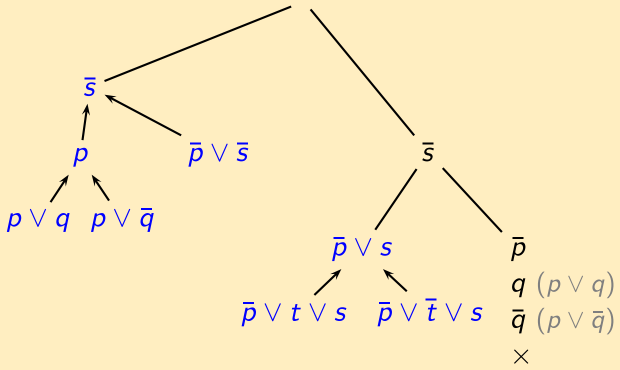
DPLL Proofs and Resolution



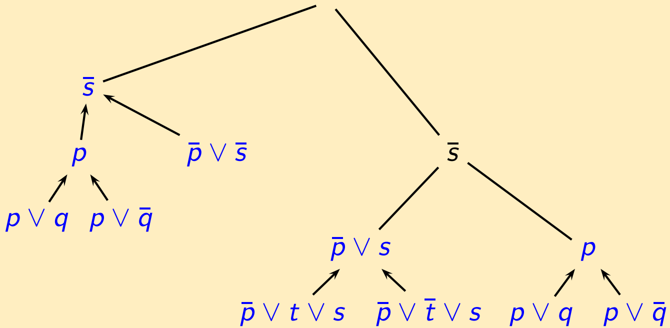
DPLL Proofs and Resolution



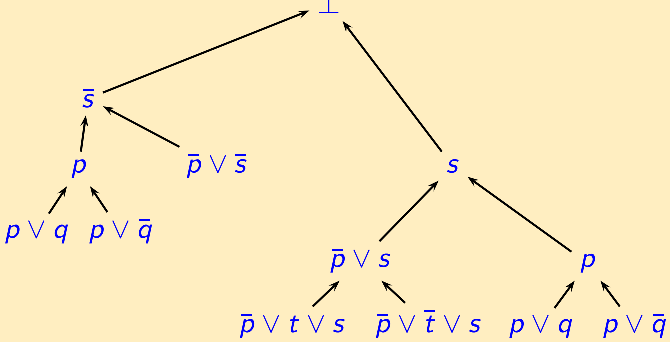
DPLL Proofs and Resolution



DPLL Proofs and Resolution



DPLL Proofs and Resolution



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

The Cost of Unit Propagation

- 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 λ 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)

Example

clause	λ_{c1}	λ_{c2}
$p \vee q \vee r$	$p = *$	$q = *$
$p \vee \bar{q} \vee s$	$p = *$	$\bar{q} = *$
$p \vee r \vee \bar{s}$	$p = *$	$r = *$

Initially $V = \emptyset$

A pair of literals was elected for each clause

All are undefined, all pairs are proper

$\bar{p}$ is chosen

$V = \{\bar{\mathbf{p}}\}$

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

New literals are chosen to be watched

clause	λ_{c1}	λ_{c2}
$p \vee q \vee r$	$r = *$	$q = *$
$p \vee \bar{q} \vee s$	$s = *$	$\bar{q} = *$
$p \vee \bar{r} \vee \bar{s}$	$\bar{s} = *$	$r = *$

$\bar{r}$ is chosen

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

clause	λ_{c1}	λ_{c2}
$p \vee q \vee r$	$r = 0$	$q = \not{*}\ 1$
$p \vee \bar{q} \vee s$	$s = *$	$\bar{q} = *$
$p \vee \bar{r} \vee \bar{s}$	$\bar{s} = \not{*}\ 1$	$r = 0$

Unit propagation leads to backtracking

$V = \{\bar{\mathbf{p}}, \bar{\mathbf{r}}, q, \bar{s}\}$
WL in clause 2 becomes improper
No other *- or 1-literal to be chosen
No unit propagation is possible: clause 2 is false

clause	λ_{c1}	λ_{c2}
$p \vee q \vee r$	$r = 0$	$q = 1$
$p \vee \bar{q} \vee s$	$s = 0$	$\bar{q} = 0$
$p \vee r \vee \bar{s}$	$\bar{s} = 1$	$r = 0$

Fast Backtracking

V is contracted to last choice point

$V = \sout{\{\bar{\mathbf{p}}, \bar{r}, q, \bar{s}\}}\ \{\bar{\mathbf{p}}, r\}$

clause	λ_{c1}	λ_{c2}
$p \vee q \vee r$	$r = 1$	$q = *$
$p \vee \bar{q} \vee s$	$s = *$	$\bar{q} = *$
$p \vee r \vee \bar{s}$	$\bar{s} = *$	$r = 1$

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: $N = 15 \longrightarrow N = 10\,000$

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: $N = 15 \longrightarrow N = 10\,000$
- 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: $N = 15 \longrightarrow N = 10\,000$
- 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: $N = 15 \longrightarrow N = 10\,000$
- 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

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: $N = 15 \longrightarrow N = 10\,000$
- 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