Introduction to Automated Reasoning and Satisfiability

Marijn J.H. Heule

Carnegie Mellon University

http://www.cs.cmu.edu/~mheule/15816-f23/

Automated Reasoning and Satisfiability
August 28, 2023

To Start...

Marijn Heule

Instructor

Ruben Martins

Instructor

Let's start by shortly introducing ourselves

To Start...

Marijn Heule

Instructor

Ruben Martins

Instructor

Let's start by shortly introducing ourselves

Everyone is expect to attend the lectures

- Email us prior to a lecture if you can't attend.

Automated Reasoning Has Many Applications

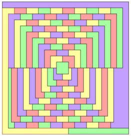
formal verification

security

bioinformatics

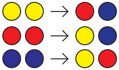
planning and scheduling

train safety



automated theorem proving

exploit generation



term rewriting termination

encode → **automated reasoning** → **decode**

Automated Reasoning Has Many Applications

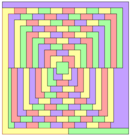
formal verification

security

bioinformatics

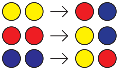
planning and scheduling

train safety



automated theorem proving

exploit generation



term rewriting termination

encode

automated reasoning

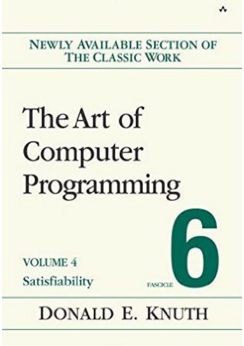
decode

Microsoft IBM intel aws

Breakthrough in SAT Solving in the Last 20 Years

Satisfiability (SAT) problem: Can a Boolean formula be satisfied?

mid '90s: formulas solvable with thousands of variables and clauses
now: formulas solvable with millions of variables and clauses



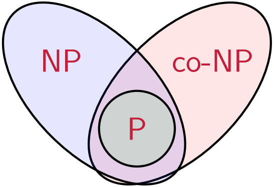
Edmund Clarke: *"a key technology of the 21st century"* [Biere, Heule, vanMaaren, and Walsh '09]

Donald Knuth: *"evidently a killer app, because it is key to the solution of so many other problems"* [Knuth '15]

Satisfiability and Complexity

Complexity classes of decision problems:

P : efficiently computable answers.
NP : efficiently checkable yes-answers.
co-NP : efficiently checkable no-answers.



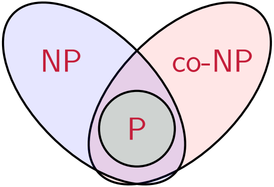
Cook-Levin Theorem [1971]: SAT is NP-complete.

Solving the P $\stackrel{?}{=}$ NP question is worth \$1,000,000 [Clay MI '00].

Satisfiability and Complexity

Complexity classes of decision problems:

P : efficiently computable answers.
NP : efficiently checkable yes-answers.
co-NP : efficiently checkable no-answers.



Cook-Levin Theorem [1971]: SAT is NP-complete.

Solving the P $\stackrel{?}{=}$ NP question is worth $1,000,000 [Clay MI '00].

The effectiveness of SAT solving: fast solutions in practice.

The beauty of NP: guaranteed short solutions.

"NP is the new P!"

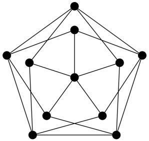
Course Overview

date	*topic*	*slides*	*video*	*notes*
08/28/2023	Introduction to Automated Reasoning	pdf (F22)	link (F20)	
08/30/2023	Applications for Automated Reasoning	pdf (F22)	link (F20)	
09/06/2023	Representations for Automated Reasoning	pdf (F22)	link (F20)	
09/11/2023	SAT and SMT Solvers in Practice	pdf (F22)	link (F20)	*Homework 1 assigned*
09/13/2023	Conflict-Driven Clause Learning	pdf (F22)	link (F20)	
09/18/2023	Preprocessing Techniques	pdf (F22)	link (F20)	*Homework 1 due*
09/20/2023	Proof Systems and Proof Complexity	pdf (F22)	link (F20)	*Homework 2 assigned*
09/25/2023	Binary Decision Diagrams	pdf, pdf (F22)	link (F20)	
09/27/2023	Local Search and Lookahead Techniques	pdf, pdf (F22)	link (F20)	*Homework 2 due*
10/02/2023	Maximum Satisfiability	pdf (F22)	link (F20)	*Homework 3 assigned*
10/04/2023	Synthesis	pdf (F20)	link (F20)	
10/09/2023	Verifying Automated Reasoning Results	pdf (F21)	link (F20)	*Homework 3 due*
10/11/2023	Parallel Automated Reasoning			
10/16/2023	**Select topic for final project and form groups**			
12/14/2023	**Project presentations**			

Course Reports (I)

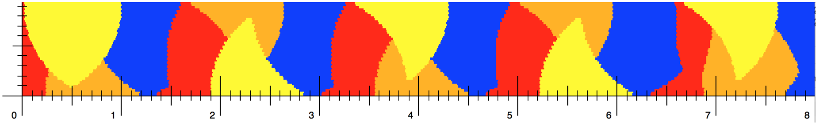
The second half of the course consists of a project

- A group of 2 (or 1) students work on a research question
- The results will be presented in a scientific report
- Several have been published in journals and at conferences

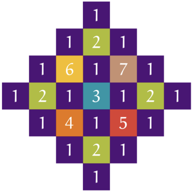


Emre Yolcu, Xinyu Wu, and Marijn J. H. Heule
Mycielski graphs and PR proofs (2020). In Theory and Practice of Satisfiability Testing - SAT 2020, Lecture Notes in Computer Science 12178, pp. 201-217.
Best student paper award

Peter Oostema, Ruben Martins, and Marijn J. H. Heule.
Coloring Unit-Distance Strips using SAT (2020).
In Logic for Programming, Artificial Intelligence and Reasoning, EPiC Series in Computing 73, pp. 373-389.

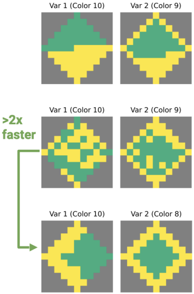


Course Reports (II)



Bernardo Subercaseaux and Marijn Heule.
The Packing Chromatic Number of the Infinite Square Grid is 15. Tools and Algorithms for the Construction and Analysis of Systems 2023, pp. 389–406.
In Quanta Magazine and The New York Times

Andrew Haberlandt, Harrison Green, and Marijn Heule.
Effective Auxiliary Variables via Structured Reencoding
In Theory and Practice of Satisfiability Testing 2023, LIPIcs 271, pp. 11:1–11:19.
The solver won SAT Competition 2023



Diplomacy Problem

"You are chief of protocol for the embassy ball. The crown prince instructs you either to invite *Peru* or to exclude *Qatar*. The queen asks you to invite either *Qatar* or *Romania* or both. The king, in a spiteful mood, wants to snub either *Romania* or *Peru* or both. Is there a guest list that will satisfy the whims of the entire royal family?"

Diplomacy Problem

"You are chief of protocol for the embassy ball. The crown prince instructs you either to invite *Peru* or to exclude *Qatar*. The queen asks you to invite either *Qatar* or *Romania* or both. The king, in a spiteful mood, wants to snub either *Romania* or *Peru* or both. Is there a guest list that will satisfy the whims of the entire royal family?"

$$(p \vee \overline{q}) \wedge (q \vee r) \wedge (\overline{r} \vee \overline{p})$$

Truth Table

$$F := (p \vee \overline{q}) \wedge (q \vee r) \wedge (\overline{r} \vee \overline{p})$$

p	q	r	falsifies	eval(F)
0	0	0	$(q \vee r)$	0
0	0	1	—	1
0	1	0	$(p \vee \overline{q})$	0
0	1	1	$(p \vee \overline{q})$	0
1	0	0	$(q \vee r)$	0
1	0	1	$(\overline{r} \vee \overline{p})$	0
1	1	0	—	1
1	1	1	$(\overline{r} \vee \overline{p})$	0

Slightly Harder Example

Slightly Harder Example 1

What are the solutions for the following formula?

$$
\begin{aligned}
&(a \lor b \lor \overline{c}) \land \\
&(\overline{a} \lor \overline{b} \lor c) \land \\
&(b \lor c \lor \overline{d}) \land \\
&(\overline{b} \lor \overline{c} \lor d) \land \\
&(a \lor c \lor d) \land \\
&(\overline{a} \lor \overline{c} \lor \overline{d}) \land \\
&(\overline{a} \lor b \lor d)
\end{aligned}
$$

Slightly Harder Example

Slightly Harder Example 1

What are the solutions for the following formula?

$$(a \lor b \lor \overline{c}) \land (\overline{a} \lor \overline{b} \lor c) \land (b \lor c \lor \overline{d}) \land (\overline{b} \lor \overline{c} \lor d) \land (a \lor c \lor d) \land (\overline{a} \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor b \lor d)$$

a	b	c	d
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

a	b	c	d
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Pythagorean Triples Problem (I) [Ronald Graham, early 80's]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

$3^2 + 4^2 = 5^2$	$6^2 + 8^2 = 10^2$	$5^2 + 12^2 = 13^2$	$9^2 + 12^2 = 15^2$
$8^2 + 15^2 = 17^2$	$12^2 + 16^2 = 20^2$	$15^2 + 20^2 = 25^2$	$7^2 + 24^2 = 25^2$
$10^2 + 24^2 = 26^2$	$20^2 + 21^2 = 29^2$	$18^2 + 24^2 = 30^2$	$16^2 + 30^2 = 34^2$
$21^2 + 28^2 = 35^2$	$12^2 + 35^2 = 37^2$	$15^2 + 36^2 = 39^2$	$24^2 + 32^2 = 40^2$

Pythagorean Triples Problem (I) [Ronald Graham, early 80's]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

$$\begin{array}{llll} 3^2 + 4^2 = 5^2 & 6^2 + 8^2 = 10^2 & 5^2 + 12^2 = 13^2 & 9^2 + 12^2 = 15^2 \\ 8^2 + 15^2 = 17^2 & 12^2 + 16^2 = 20^2 & 15^2 + 20^2 = 25^2 & 7^2 + 24^2 = 25^2 \\ 10^2 + 24^2 = 26^2 & 20^2 + 21^2 = 29^2 & 18^2 + 24^2 = 30^2 & 16^2 + 30^2 = 34^2 \\ 21^2 + 28^2 = 35^2 & 12^2 + 35^2 = 37^2 & 15^2 + 36^2 = 39^2 & 24^2 + 32^2 = 40^2 \end{array}$$

Best lower bound: a bi-coloring of $[1, 7664]$ s.t. there is no monochromatic Pythagorean Triple [Cooper & Overstreet 2015].

Myers conjectures that the answer is No [PhD thesis, 2015].

Pythagorean Triples Problem (II) [Ronald Graham, early 80's]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

A bi-coloring of $[1, n]$ is encoded using Boolean variables x_i with $i \in \{1, 2, \ldots, n\}$ such that $x_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean Triple $a^2 + b^2 = c^2$, two clauses are added: $(x_a \vee x_b \vee x_c)$ and $(\overline{x}_a \vee \overline{x}_b \vee \overline{x}_c)$.

Pythagorean Triples Problem (II) [Ronald Graham, early 80's]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

A bi-coloring of $[1, n]$ is encoded using Boolean variables x_i with $i \in \{1, 2, \ldots, n\}$ such that $x_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean Triple $a^2 + b^2 = c^2$, two clauses are added: $(x_a \vee x_b \vee x_c)$ and $(\overline{x}_a \vee \overline{x}_b \vee \overline{x}_c)$.

Theorem ([Heule, Kullmann, and Marek (2016)])

$[1, 7824]$ *can be bi-colored s.t. there is no monochromatic Pythagorean Triple. This is impossible for* $[1, 7825]$.

Pythagorean Triples Problem (II) [Ronald Graham, early 80's]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

A bi-coloring of $[1, n]$ is encoded using Boolean variables x_i with $i \in \{1, 2, \ldots, n\}$ such that $x_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean Triple $a^2 + b^2 = c^2$, two clauses are added: $(x_a \vee x_b \vee x_c)$ and $(\overline{x}_a \vee \overline{x}_b \vee \overline{x}_c)$.

Theorem ([Heule, Kullmann, and Marek (2016)])

$[1, 7824]$ *can be bi-colored s.t. there is no monochromatic Pythagorean Triple. This is impossible for* $[1, 7825]$.

4 CPU years computation, but 2 days on cluster (800 cores)

Pythagorean Triples Problem (II) [Ronald Graham, early 80's]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

A bi-coloring of $[1, n]$ is encoded using Boolean variables x_i with $i \in \{1, 2, \dots, n\}$ such that $x_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean Triple $a^2 + b^2 = c^2$, two clauses are added: $(x_a \vee x_b \vee x_c)$ and $(\overline{x}_a \vee \overline{x}_b \vee \overline{x}_c)$.

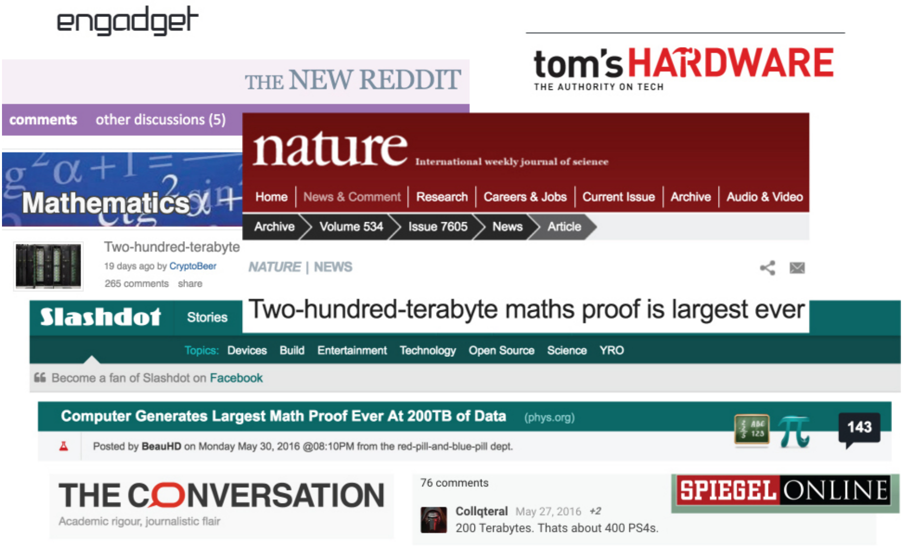
Theorem ([Heule, Kullmann, and Marek (2016)])

$[1, 7824]$ *can be bi-colored s.t. there is no monochromatic Pythagorean Triple. This is impossible for* $[1, 7825]$.

4 CPU years computation, but 2 days on cluster (800 cores)

200 terabytes proof, but validated with verified checker

Media: “The Largest Math Proof Ever”



Introduction

Terminology

Basic Solving Techniques

Solvers and Benchmarks

Given a *CNF formula*,
does there exist an *assignment*
to the *Boolean variables*
that satisfies all *clauses*?

Terminology: Variables and literals

Boolean variable x_i

- can be assigned the Boolean values 0 or 1

Literal

- refers either to x_i or its complement $\overline{x}_i$
- literals x_i are satisfied if variable x_i is assigned to 1 (true)
- literals $\overline{x}_i$ are satisfied if variable x_i is assigned to 0 (false)

Terminology: Clauses

Clause

- Disjunction of literals: E.g. $C_j = (l_1 \vee l_2 \vee l_3)$
- Can be falsified with only one assignment to its literals: All literals assigned to false
- Can be satisfied with $2^k - 1$ assignment to its k literals
- One special clause - the empty clause (denoted by $\perp$) - which is always falsified

Terminology: Formulae

Formula

- Conjunction of clauses: E.g. $F = C_1 \wedge C_2 \wedge C_3$
- Is satisfiable if there exists an assignment satisfying all clauses, otherwise unsatisfiable
- Formulae are defined in Conjunction Normal Form (CNF) and generally also stored as such - also learned information
- Any propositional formula can be efficiently transformed into CNF [Tseitin '70]

Terminology: Assignments

Assignment

- Mapping of the values 0 and 1 to the variables
- $\alpha \circ F$ results in a reduced formula F_{reduced}:
  - all satisfied clauses are removed
  - all falsified literals are removed
- satisfying assignment $\leftrightarrow$ F_{reduced} is empty
- falsifying assignment $\leftrightarrow$ F_{reduced} contains $\perp$
- partial assignment versus full assignment

Resolution

The most commonly used inference rule in propositional logic is the resolution rule (the operation is denoted by $\bowtie$)

$$\frac{C \vee x \qquad \bar{x} \vee D}{C \vee D}$$

Resolution

The most commonly used inference rule in propositional logic is the resolution rule (the operation is denoted by $\bowtie$)

$$\frac{C \lor x \qquad \bar{x} \lor D}{C \lor D}$$

Examples for $F := (p \lor \overline{q}) \land (q \lor r) \land (\bar{r} \lor \overline{p})$

- $(\overline{q} \lor p) \bowtie (\overline{p} \lor \bar{r}) = (\overline{q} \lor \bar{r})$
- $(p \lor \overline{q}) \bowtie (q \lor r) = (p \lor r)$
- $(q \lor r) \bowtie (\bar{r} \lor \overline{p}) = (q \lor \overline{p})$

Resolution

The most commonly used inference rule in propositional logic is the resolution rule (the operation is denoted by $\bowtie$)

$$\frac{C \lor x \qquad \bar{x} \lor D}{C \lor D}$$

Examples for $F := (p \lor \overline{q}) \land (q \lor r) \land (\overline{r} \lor \overline{p})$

- $(\overline{q} \lor p) \bowtie (\overline{p} \lor \overline{r}) = (\overline{q} \lor \overline{r})$
- $(p \lor \overline{q}) \bowtie (q \lor r) = (p \lor r)$
- $(q \lor r) \bowtie (\overline{r} \lor \overline{p}) = (q \lor \overline{p})$

Adding (non-redundant) resolvents until fixpoint, is a complete proof procedure. It produces the empty clause if and only if the formula is unsatisfiable

Tautology

A clause C is a tautology if it contains
for some variable x, both the literals x and $\overline{x}$.

Slightly Harder Example 2

Compute all non-tautological resolvents for:

$$\begin{array}{l}(a \vee b \vee \overline{c}) \wedge (\overline{a} \vee \overline{b} \vee c) \wedge \\ (b \vee c \vee \overline{d}) \wedge (\overline{b} \vee \overline{c} \vee d) \wedge \\ (a \vee c \vee d) \wedge (\overline{a} \vee \overline{c} \vee \overline{d}) \wedge \\ (\overline{a} \vee b \vee d)\end{array}$$

Which resolvents remain after removing the supersets?

Introduction

Terminology

Basic Solving Techniques

Solvers and Benchmarks

SAT solving: Unit propagation

A *unit clause* is a clause of size 1

UnitPropagation (α, F):

while $\bot \notin F$ **and** unit clause y exists **do**
    expand α by adding $y = 1$ and simplify F
end while
return α, F

Unit Propagation: Example

$$F_{\text{unit}} := (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge$$
$$(\bar{x}_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\bar{x}_1 \vee x_4 \vee \bar{x}_5) \wedge$$
$$(x_1 \vee \bar{x}_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \bar{x}_6)$$

Unit Propagation: Example

$$F_{\text{unit}} := (\bar{x}_1 \lor \bar{x}_3 \lor x_4) \land (\bar{x}_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor x_3 \lor x_6) \land (\bar{x}_1 \lor x_4 \lor \bar{x}_5) \land (x_1 \lor \bar{x}_6) \land (x_4 \lor x_5 \lor x_6) \land (x_5 \lor \bar{x}_6)$$

$$\alpha = \{x_1{=}1\}$$

Unit Propagation: Example

$$F_{\text{unit}} := (\overline{x}_1 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\overline{x}_1 \vee x_4 \vee \overline{x}_5) \wedge (x_1 \vee \overline{x}_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \overline{x}_6)$$

$$\alpha = \{x_1{=}1, x_2{=}1\}$$

Unit Propagation: Example

$$F_{\text{unit}} := (\bar{x}_1 \lor \bar{x}_3 \lor x_4) \land (\bar{x}_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor x_3 \lor x_6) \land (\bar{x}_1 \lor x_4 \lor \bar{x}_5) \land (x_1 \lor \bar{x}_6) \land (x_4 \lor x_5 \lor x_6) \land (x_5 \lor \bar{x}_6)$$

$$\alpha = \{x_1{=}1, x_2{=}1, x_3{=}1\}$$

Unit Propagation: Example

$$F_{\text{unit}} := (\overline{x}_1 \lor \overline{x}_3 \lor x_4) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_3) \land (\overline{x}_1 \lor x_2) \land (x_1 \lor x_3 \lor x_6) \land (\overline{x}_1 \lor x_4 \lor \overline{x}_5) \land (x_1 \lor \overline{x}_6) \land (x_4 \lor x_5 \lor x_6) \land (x_5 \lor \overline{x}_6)$$

$$\alpha = \{x_1{=}1, x_2{=}1, x_3{=}1, x_4{=}1\}$$

Reverse Unit Propagation

- *Unit propagation* (UP) satisfies unit clauses by assigning their literal to true (until fixpoint or a conflict).
- Let F be a formula. A clause C is implied by F via UP (denoted by $F \vdash_1 C$) if UP on $F \land \neg C$ results in a conflict.

Example

$$\begin{aligned} F = \ & (a \lor b \lor \bar{c}) \land (\bar{a} \lor \bar{b} \lor c) \land (b \lor c \lor \bar{d}) \land (\bar{b} \lor \bar{c} \lor d) \ \land \\ & (a \lor c \lor d) \land (\bar{a} \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor b \lor d) \land (a \lor \bar{b} \lor \bar{d}) \end{aligned}$$

Reverse Unit Propagation

- *Unit propagation* (UP) satisfies unit clauses by assigning their literal to true (until fixpoint or a conflict).
- Let F be a formula. A clause C is implied by F via UP (denoted by $F \vdash_1 C$) if UP on $F \land \neg C$ results in a conflict.

Example

$$F = (a \lor b \lor \bar{c}) \land (\bar{a} \lor \bar{b} \lor c) \land (b \lor c \lor \bar{d}) \land (\bar{b} \lor \bar{c} \lor d) \land (a \lor c \lor d) \land (\bar{a} \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor b \lor d) \land (a \lor \bar{b} \lor \bar{d})$$

clause	$(a \lor b)$
units	$\bar{a} \land \bar{b}$

Reverse Unit Propagation

- *Unit propagation* (UP) satisfies unit clauses by assigning their literal to true (until fixpoint or a conflict).
- Let F be a formula. A clause C is implied by F via UP (denoted by $F \vdash_1 C$) if UP on $F \land \neg C$ results in a conflict.

Example

$$F = (a \lor b \lor \overline{c}) \land (\overline{a} \lor \overline{b} \lor c) \land (b \lor c \lor \overline{d}) \land (\overline{b} \lor \overline{c} \lor d) \land (a \lor c \lor d) \land (\overline{a} \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor b \lor d) \land (a \lor \overline{b} \lor \overline{d})$$

clause	$(a \lor b)$	$(a \lor b \lor \overline{c})$
units	$\overline{a} \land \overline{b}$	$\overline{c}$

Reverse Unit Propagation

- *Unit propagation* (UP) satisfies unit clauses by assigning their literal to true (until fixpoint or a conflict).
- Let F be a formula. A clause C is implied by F via UP (denoted by $F \vdash_1 C$) if UP on $F \land \neg C$ results in a conflict.

Example

$$F = (a \lor b \lor \bar{c}) \land (\bar{a} \lor \bar{b} \lor c) \land (b \lor c \lor \bar{d}) \land (\bar{b} \lor \bar{c} \lor d) \land (a \lor c \lor d) \land (\bar{a} \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor b \lor d) \land (a \lor \bar{b} \lor \bar{d})$$

clause	$(a \lor b)$	$(a \lor b \lor \bar{c})$	$(b \lor c \lor \bar{d})$
units	$\bar{a} \land \bar{b}$	$\bar{c}$	$\bar{d}$

Reverse Unit Propagation

- *Unit propagation* (UP) satisfies unit clauses by assigning their literal to true (until fixpoint or a conflict).
- Let F be a formula. A clause C is implied by F via UP (denoted by $F \vdash_1 C$) if UP on $F \land \neg C$ results in a conflict.

Example

$$F = (a \lor b \lor \bar{c}) \land (\bar{a} \lor \bar{b} \lor c) \land (b \lor c \lor \bar{d}) \land (\bar{b} \lor \bar{c} \lor d) \land (a \lor c \lor d) \land (\bar{a} \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor b \lor d) \land (a \lor \bar{b} \lor \bar{d})$$

clause	$(a \lor b)$	$(a \lor b \lor \bar{c})$	$(b \lor c \lor \bar{d})$	$(a \lor c \lor d)$
units	$\bar{a} \land \bar{b}$	$\bar{c}$	$\bar{d}$	$\perp$

Reverse Unit Propagation

- *Unit propagation* (UP) satisfies unit clauses by assigning their literal to true (until fixpoint or a conflict).
- Let F be a formula. A clause C is implied by F via UP (denoted by $F \vdash_1 C$) if UP on $F \land \neg C$ results in a conflict.

Example

$$F = (a \lor b \lor \bar{c}) \land (\bar{a} \lor \bar{b} \lor c) \land (b \lor c \lor \bar{d}) \land (\bar{b} \lor \bar{c} \lor d) \land (a \lor c \lor d) \land (\bar{a} \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor b \lor d) \land (a \lor \bar{b} \lor \bar{d})$$

clause	$(a \lor b)$	$(a \lor b \lor \bar{c})$	$(b \lor c \lor \bar{d})$	$(a \lor c \lor d)$
units	$\bar{a} \land \bar{b}$	$\bar{c}$	$\bar{d}$	$\bot$

$$\dfrac{\dfrac{(a \lor c \lor d) \qquad (b \lor c \lor \bar{d})}{(a \lor b \lor c)} \qquad (a \lor b \lor \bar{c})}{(a \lor b)}$$

SAT Solving: DPLL

Davis Putnam Logemann Loveland [DP60,DLL62]

Recursive procedure that in each recursive call:

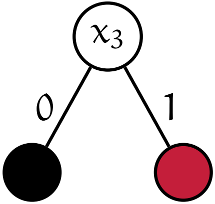
- Simplifies the formula (using unit propagation)
- Splits the formula into two subformulas
  - Variable selection heuristics (which variable to split on)
  - Direction heuristics (which subformula to explore first)

DPLL: Example

$$F_{\text{DPLL}} := (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (x_1 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_3)$$

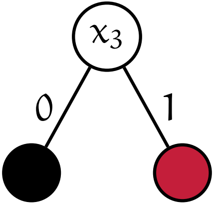
DPLL: Example

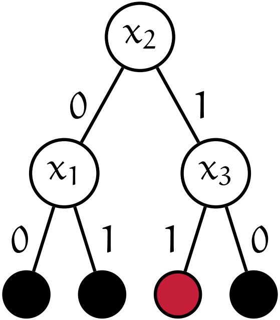
$$F_{\text{DPLL}} := (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_3) \land (x_1 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_3)$$



DPLL: Example

$$F_{\text{DPLL}} := (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_3) \land (x_1 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_3)$$





DPLL: Slightly Harder Example

Slightly Harder Example 3

Construct a DPLL tree for:

$$
\begin{aligned}
&(a \lor b \lor \overline{c}) \land (\overline{a} \lor \overline{b} \lor c) \land \\
&(b \lor c \lor \overline{d}) \land (\overline{b} \lor \overline{c} \lor d) \land \\
&(a \lor c \lor d) \land (\overline{a} \lor \overline{c} \lor \overline{d}) \land \\
&(\overline{a} \lor b \lor d)
\end{aligned}
$$

SAT Solving: Decision and Implications

Decision variables

- Variable selection heuristics and direction heuristics
- Play a crucial role in performance

Implied variables

- Assigned by reasoning (e.g. unit propagation)
- Maximizing the number of implied variables is an important aspect of look-ahead SAT solvers

SAT Solving: Clauses $\leftrightarrow$ assignments

- A clause C represents a set of falsified assignments, i.e. those assignments that falsify all literals in C
- A falsifying assignment α for a given formula represents a set of clauses that follow from the formula
  - For instance with all decision variables
  - Important feature of conflict-driven SAT solvers

Introduction

Terminology

Basic Solving Techniques

Solvers and Benchmarks

SAT Solving Paradigms

Conflict-driven

- search for short refutation, complete
- examples: lingeling, glucose, CaDiCaL, kissat

Look-ahead

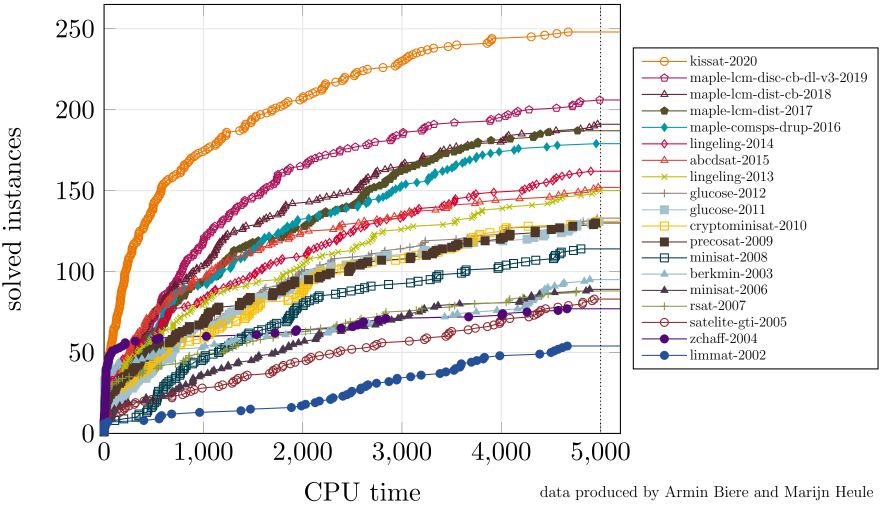
- extensive inference, complete
- examples: march, OKsolver, kcnfs

Local search

- local optimizations, incomplete
- examples: probSAT, UnitWalk, DDFW, Dimetheus

Progress of SAT Solvers

SAT Competition Winners on the SC2020 Benchmark Suite



data produced by Armin Biere and Marijn Heule

Applications: Industrial

- Model checking
  - Turing award '07 Clarke, Emerson, and Sifakis
- Software verification
- Hardware verification
- Equivalence checking
- Planning and scheduling
- Cryptography
- Car configuration
- Railway interlocking

Applications: Crafted

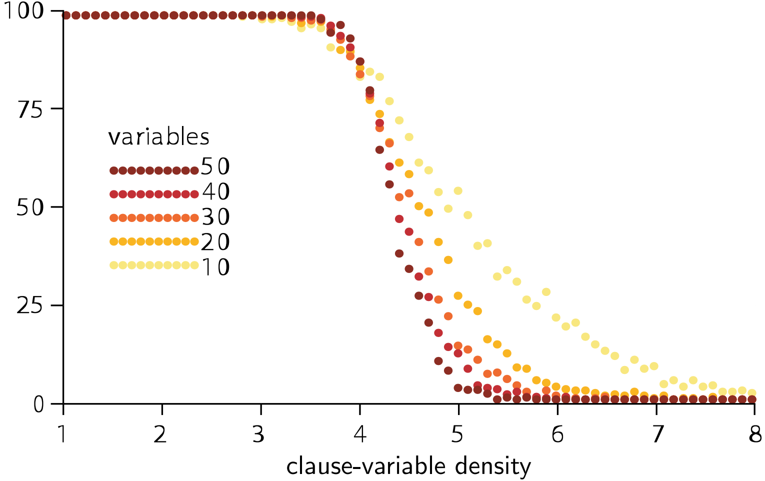
Combinatorial challenges and solver obstruction instances

- Pigeon-hole problems
- Tseitin problems
- Mutilated chessboard problems
- Sudoku
- Factorization problems
- Ramsey theory
- Rubik's cube puzzles

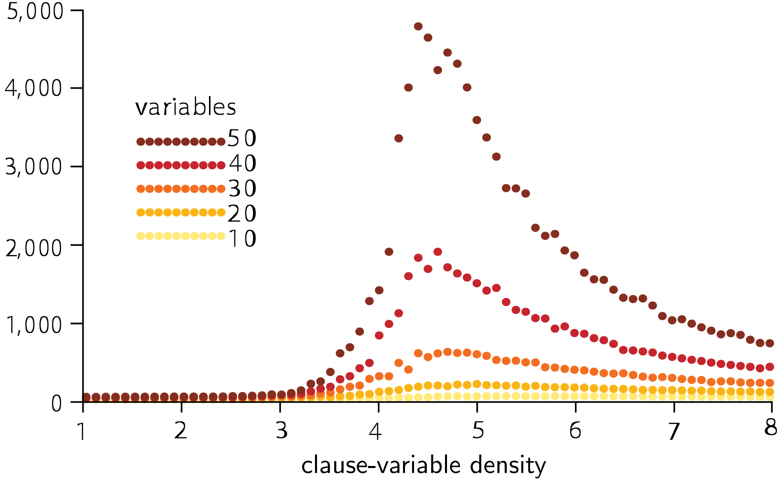
Random k-SAT: Introduction

- All clauses have length k
- Variables have the same probability to occur
- Each literal is negated with probability of 50%
- Density is ratio Clauses to Variables

Random 3-SAT: % satisfiable, the phase transition



Random 3-SAT: exponential runtime, the threshold



SAT Game

by Olivier Roussel

http://www.cs.utexas.edu/~marijn/game/