

Introduction to Automated Reasoning and Satisfiability

Marijn J.H. Heule

**Carnegie
Mellon
University**

<http://www.cs.cmu.edu/~mheule/15816-f21/>

Automated Reasoning and Satisfiability
September 8, 2021

To Start...

Let's start by shortly introducing ourselves

To Start...

Let's start by shortly introducing ourselves

Everyone is expect to attend the lectures

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

Office hours on Mondays at 3pm

- Zoom link provided via email

Automated Reasoning Has Many Applications



formal verification



security



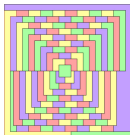
bioinformatics



planning and
scheduling



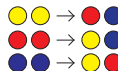
train safety



automated
theorem proving



exploit
generation



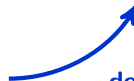
term rewriting
termination

encode



automated reasoning

decode



Automated Reasoning Has Many Applications



formal verification



security



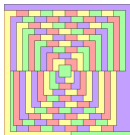
bioinformatics



planning and
scheduling



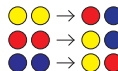
train safety



automated
theorem proving



exploit
generation



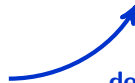
term rewriting
termination

encode



automated reasoning

decode

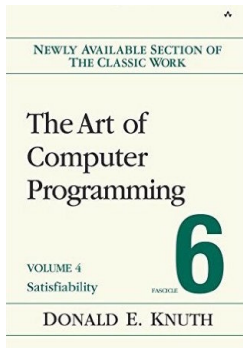
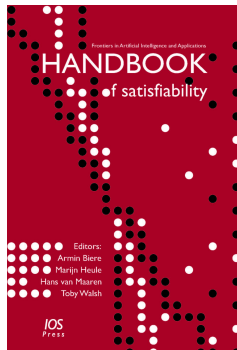


Breakthrough in SAT Solving in the Last 20 Years

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]

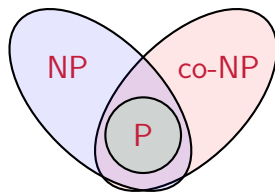
Satisfiability and Complexity

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].

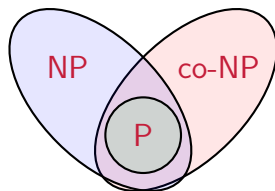
Satisfiability and Complexity

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

Classes start in the week of September 7, 2021. Lecture slides will be posted after each class meeting. The postfix (F20) denotes slides and links of last year.

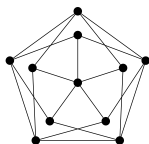
This schedule may change throughout the semester. Check back regularly for updates, including assignment deadlines and other important dates.

<i>date</i>	<i>topic</i>	<i>slides</i>	<i>video</i>	<i>notes</i>
9/8/21	Introduction to Automated Reasoning	pdf (F20)	link (F20)	
9/13/21	Applications for Automated Reasoning	pdf (F20)	link (F20)	
9/15/21	Representations for Automated Reasoning	pdf (F20)	link (F20)	
9/20/21	SAT and SMT Solvers in Practice	pdf (F20)	link (F20)	<i>Homework 1 assigned</i>
9/22/21	Conflict-Driven Clause Learning	pdf (F20)	link (F20)	
9/27/21	Preprocessing Techniques	pdf (F20)	link (F20)	<i>Homework 1 due</i>
9/29/21	Proof Systems and Proof Complexity	pdf (F20)	link (F20)	<i>Homework 2 assigned</i>
10/4/21	Maximum Satisfiability	pdf (F20)	link (F20)	
10/6/21	Local Search and Lookahead Techniques	pdf , pdf (F20)	link (F20)	<i>Homework 2 due</i>
10/11/21	Quantified Boolean Formulas	pdf (F20)	link (F20)	<i>Homework 3 assigned</i>
10/13/21	Binary Decision Diagrams	pdf (F20)	link (F20)	
10/18/21	Verifying Automated Reasoning Results	pdf (F20)	link (F20)	<i>Homework 3 due</i>
10/20/21	Select topic for final project and form groups			

Course Reports

The second half of the course consists of a project

- A group of 1 to 3 students work on a research question
- The results will be presented in a scientific report
- Several have been published in journals and at conferences



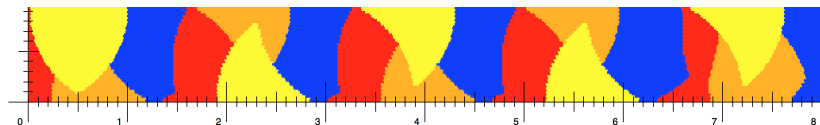
Emre Yolcu, Xinyu Wu, and Marijn J. H. Heule
Mycielski graphs and PR proofs (2020). In Theory and Practice of Satisfiability Testing - SAT 2020, Lecture Notes in Computer Science 12178, pp. 201-217.

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.



Introduction

Terminology

Basic Solving Techniques

Solvers and Benchmarks

Introduction

Terminology

Basic Solving Techniques

Solvers and Benchmarks

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?”

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

Truth Table

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

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

Slightly Harder Example

Slightly Harder Example 1

What are the solutions for the following formula?

$$(a \vee b \vee \bar{c}) \wedge$$

$$(\bar{a} \vee \bar{b} \vee c) \wedge$$

$$(b \vee c \vee \bar{d}) \wedge$$

$$(\bar{b} \vee \bar{c} \vee d) \wedge$$

$$(a \vee c \vee d) \wedge$$

$$(\bar{a} \vee \bar{c} \vee \bar{d}) \wedge$$

$$(\bar{a} \vee b \vee d)$$

Slightly Harder Example

Slightly Harder Example 1

What are the solutions for the following formula?

	a	b	c	d		a	b	c	d
$(a \vee b \vee \bar{c}) \wedge$	0	0	0	0		1	0	0	0
$(\bar{a} \vee \bar{b} \vee c) \wedge$	0	0	0	1		1	0	0	1
$(b \vee c \vee \bar{d}) \wedge$	0	0	1	0		1	0	1	0
$(\bar{b} \vee \bar{c} \vee d) \wedge$	0	0	1	1		1	0	1	1
$(a \vee c \vee d) \wedge$	0	1	0	0		1	1	0	0
$(\bar{a} \vee \bar{c} \vee \bar{d}) \wedge$	0	1	0	1		1	1	0	1
$(\bar{a} \vee b \vee d)$	0	1	1	0		1	1	1	0
	0	1	1	1		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 $a^2 + b^2 = c^2$?

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

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 $i \in \{1, 2, \dots, n\}$ 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 $(\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$.

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 $i \in \{1, 2, \dots, n\}$ 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 $(\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$.

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 $i \in \{1, 2, \dots, n\}$ 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 $(\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$.

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 $(\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$.

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”

engadget

THE NEW REDDIT

comments other discussions (5)

Mathematics

nature

International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video

Archive > Volume 534 > Issue 7605 > News > Article

Two-hundred-terabyte

19 days ago by [CryptoBeer](#)

265 comments share

NATURE | NEWS



Slashdot

Stories

Two-hundred-terabyte maths proof is largest ever

Topics: Devices Build Entertainment Technology Open Source Science YRO

Become a fan of Slashdot on Facebook

Computer Generates Largest Math Proof Ever At 200TB of Data (phys.org)



Posted by **BeauHD** on Monday May 30, 2016 @08:10PM from the red-pill-and-blue-pill dept.



143

THE CONVERSATION

Academic rigour, journalistic flair

76 comments



Collqteral May 27, 2016 +2

200 Terabytes. Thats about 400 PS4s.

SPIEGEL ONLINE

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 \bar{x}_i
- literals x_i are satisfied if variable x_i is assigned to 1 (true)
- literals \bar{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. $\mathcal{F} = C_1 \wedge C_2 \wedge C_3$
- 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
- $\alpha \circ \mathcal{F}$ results in a reduced formula $\mathcal{F}_{\text{reduced}}$:
 - all satisfied clauses are removed
 - all falsified literals are removed
- **satisfying assignment** $\leftrightarrow \mathcal{F}_{\text{reduced}}$ is empty
- **falsifying assignment** $\leftrightarrow \mathcal{F}_{\text{reduced}}$ 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)

$$\frac{C \vee \textcolor{blue}{x} \quad \textcolor{red}{\bar{x}} \vee D}{C \vee D}$$

Resolution

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

$$\frac{C \vee \textcolor{blue}{x} \quad \textcolor{red}{\bar{x}} \vee D}{C \vee D}$$

Examples for $F := (p \vee \bar{q}) \wedge (q \vee r) \wedge (\bar{r} \vee \bar{p})$

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

Resolution

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

$$\frac{C \vee \textcolor{blue}{x} \quad \textcolor{red}{\bar{x}} \vee D}{C \vee D}$$

Examples for $F := (p \vee \bar{q}) \wedge (q \vee r) \wedge (\bar{r} \vee \bar{p})$

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

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 \bar{x} .

Slightly Harder Example 2

Compute all non-tautological resolvents for:

$$\begin{aligned} & (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee \bar{b} \vee c) \wedge \\ & (b \vee c \vee \bar{d}) \wedge (\bar{b} \vee \bar{c} \vee d) \wedge \\ & (a \vee c \vee d) \wedge (\bar{a} \vee \bar{c} \vee \bar{d}) \wedge \\ & (\bar{a} \vee b \vee d) \end{aligned}$$

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, \mathcal{F}):

- 1: **while** $\perp \notin \mathcal{F}$ **and** unit clause y exists **do**
- 2: expand α by adding $y = 1$ and simplify \mathcal{F}
- 3: **end while**
- 4: **return** α, \mathcal{F}

Unit Propagation: Example

$$\begin{aligned}\mathcal{F}_{\text{unit}} := & (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge \\ & (\bar{x}_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\bar{x}_1 \vee x_4 \vee \bar{x}_5) \wedge \\ & (x_1 \vee \bar{x}_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \bar{x}_6)\end{aligned}$$

Unit Propagation: Example

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

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

Unit Propagation: Example

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

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

Unit Propagation: Example

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

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

Unit Propagation: Example

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

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

Davis Putnam Logemann Loveland [DP60,DLL62]

Recursive procedure that in each recursive call:

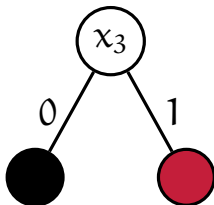
- Simplifies the formula (using unit propagation)
- Splits the formula into two subformulas
 - Variable selection heuristics (which variable to split on)
 - Direction heuristics (which subformula to explore first)

DPLL: Example

$$\mathcal{F}_{\text{DPLL}} := (\mathfrak{x}_1 \vee \mathfrak{x}_2 \vee \bar{\mathfrak{x}}_3) \wedge (\bar{\mathfrak{x}}_1 \vee \mathfrak{x}_2 \vee \mathfrak{x}_3) \wedge \\ (\bar{\mathfrak{x}}_1 \vee \bar{\mathfrak{x}}_2 \vee \mathfrak{x}_3) \wedge (\mathfrak{x}_1 \vee \mathfrak{x}_3) \wedge (\bar{\mathfrak{x}}_1 \vee \bar{\mathfrak{x}}_3)$$

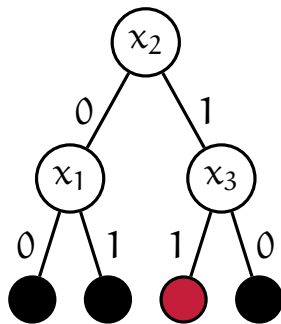
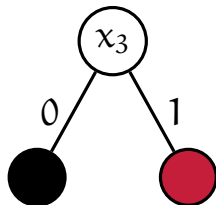
DPLL: Example

$$\mathcal{F}_{\text{DPLL}} := (\chi_1 \vee \chi_2 \vee \bar{\chi}_3) \wedge (\bar{\chi}_1 \vee \chi_2 \vee \chi_3) \wedge (\bar{\chi}_1 \vee \bar{\chi}_2 \vee \chi_3) \wedge (\chi_1 \vee \chi_3) \wedge (\bar{\chi}_1 \vee \bar{\chi}_3)$$



DPLL: Example

$$\mathcal{F}_{\text{DPLL}} := (\chi_1 \vee \chi_2 \vee \bar{\chi}_3) \wedge (\bar{\chi}_1 \vee \chi_2 \vee \chi_3) \wedge (\bar{\chi}_1 \vee \bar{\chi}_2 \vee \chi_3) \wedge (\chi_1 \vee \chi_3) \wedge (\bar{\chi}_1 \vee \bar{\chi}_3)$$



DPLL: Slightly Harder Example

Slightly Harder Example 3

Construct a DPLL tree for:

$$\begin{aligned} & (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee \bar{b} \vee c) \wedge \\ & (b \vee c \vee \bar{d}) \wedge (\bar{b} \vee \bar{c} \vee d) \wedge \\ & (a \vee c \vee d) \wedge (\bar{a} \vee \bar{c} \vee \bar{d}) \wedge \\ & (\bar{a} \vee b \vee d) \end{aligned}$$

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

Look-ahead

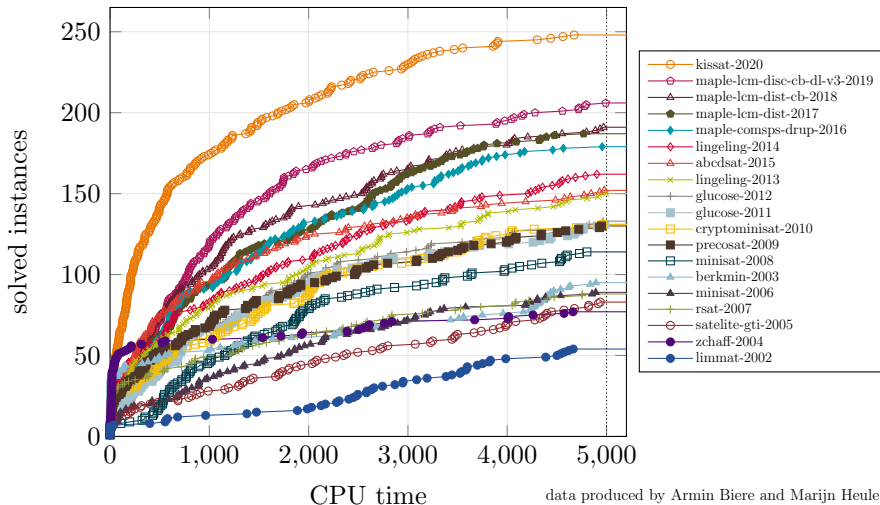
- extensive inference, complete
- examples: march, OKsolver, kcnfs

Local search

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

Progress of SAT Solvers

SAT Competition Winners on the SC2020 Benchmark Suite



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

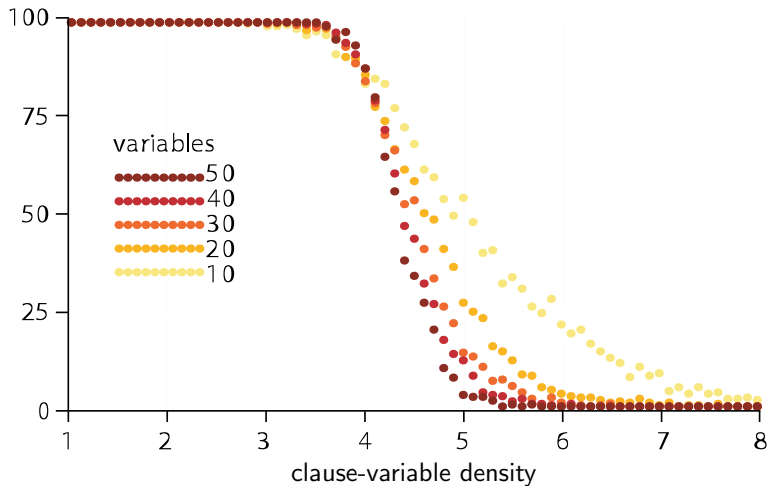
Combinatorial challenges and solver obstruction instances

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

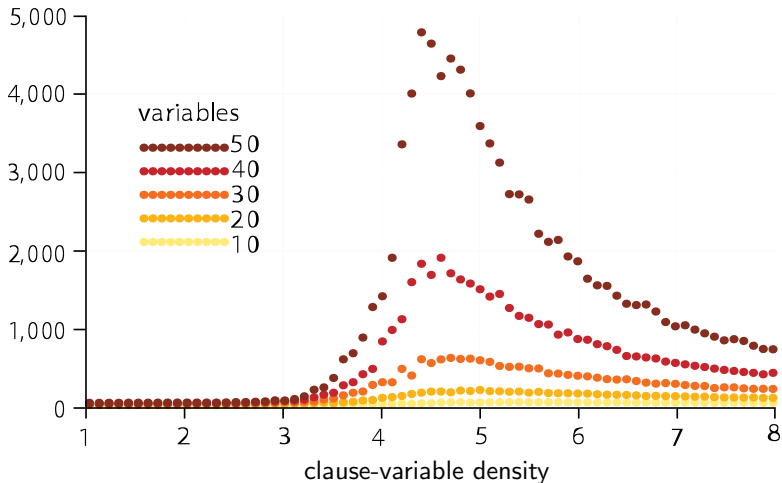
Random k -SAT: Introduction

- 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/>

Introduction to Automated Reasoning and Satisfiability

Marijn J.H. Heule

**Carnegie
Mellon
University**

<http://www.cs.cmu.edu/~mheule/15816-f21/>

Automated Reasoning and Satisfiability
September 8, 2021