Computer Assisted Mathematical Discovery

John Mackey

Carnegie Mellon University

CMUMC Colloquium

Carnegie Mellon University April 17, 2024

50 Years of Success in Computer-Assisted Mathematics

- 1976 Four-Color Theorem
- 1998 Kepler's Conjecture
- 2010 Largest number of moves to solve a Rubik's Cube is 20
- 2014 Erdős discrepancy problem ($C = 2$)
- 2016 2-Color Pythagorean triples problem
- 2018 Computation of Schur's fifth number
- 2019 Keller's Conjecture
- 2022 The Packing Number of the Infinite Square Grid is 15
- 2023 There is an Empty Hexagon in Every 30 Points

50 Years of Success in Computer-Assisted Mathematics

- 1976 Four-Color Theorem
- 1998 Kepler's Conjecture
- 2010 Largest number of moves to solve a Rubik's Cube is 20
- 2014 Erdős discrepancy problem ($C = 2$) SAT
- 2016 2-Color Pythagorean triples problem SAT
- 2018 Computation of Schur's fifth number SAT
- 2019 Keller's Conjecture SAT
- 2022 The Packing Number of the Infinite Square Grid is 15 SAT
- 2023 There is an Empty Hexagon in Every 30 Points SAT

What is SAT ?

What is SAT ?

SAT is the problem of determining whether the variables of a propositional formula can be assigned values in {TRUE, FALSE} in such a way to make the formula evaluate to TRUE.

What is SAT?

SAT is the problem of determining whether the variables of a propositional formula can be assigned values in {TRUE, FALSE} in such a way to make the formula evaluate to TRUE.

If such an assignment exists, then the formula is said to be *satisfiable*. Otherwise, the formula is said to be *unsatisfiable*.

What is SAT ?

SAT is the problem of determining whether the variables of a propositional formula can be assigned values in {TRUE, FALSE} in such a way to make the formula evaluate to TRUE.

If such an assignment exists, then the formula is said to be *satisfiable*. Otherwise, the formula is said to be *unsatisfiable*.

Consider, for example,

$$G := (p \vee \neg q) \wedge (q \vee r) \wedge (\neg r \vee \neg p).$$

What is SAT ?

SAT is the problem of determining whether the variables of a propositional formula can be assigned values in {TRUE, FALSE} in such a way to make the formula evaluate to TRUE.

If such an assignment exists, then the formula is said to be *satisfiable*. Otherwise, the formula is said to be *unsatisfiable*.

Consider, for example,

$$G := (p \vee \neg q) \wedge (q \vee r) \wedge (\neg r \vee \neg p).$$

How about $H := (\neg v \wedge (v \vee w)) \wedge (\neg w)$?

Naive SAT Solving via Truth Table

Naive SAT Solving via Truth Table

Recall $\quad G := (p \vee \neg q) \wedge (q \vee r) \wedge (\neg r \vee \neg p).$

Naive SAT Solving via Truth Table

Recall $G := (p \vee \neg q) \wedge (q \vee r) \wedge (\neg r \vee \neg p)$.

p	q	r	falsifies	evaluation
F	F	F	$(q \vee r)$	F
F	F	T	none	T
F	T	F	$(p \vee \neg q)$	F
F	T	T	$(p \vee \neg q)$	F
T	F	F	$(q \vee r)$	F
T	F	T	$(\neg r \vee \neg p)$	F
T	T	F	none	T
T	T	T	$(\neg r \vee \neg p)$	F

Doing better than Truth Tables

Doing better than Truth Tables

Given a propositional formula, we first express it in Conjunctive Normal Form (ANDs of ORs) as in the following example:

Doing better than Truth Tables

Given a propositional formula, we first express it in Conjunctive Normal Form (ANDs of ORs) as in the following example:

$$(p \Rightarrow q) \Rightarrow r$$

Doing better than Truth Tables

Given a propositional formula, we first express it in Conjunctive Normal Form (ANDs of ORs) as in the following example:

$$
\begin{aligned}
& (p \Rightarrow q) \Rightarrow r \\
\equiv \quad & (\neg p \vee q) \Rightarrow r
\end{aligned}
$$

Doing better than Truth Tables

Given a propositional formula, we first express it in Conjunctive Normal Form (ANDs of ORs) as in the following example:

$$
\begin{aligned}
& (p \Rightarrow q) \Rightarrow r \\
\equiv \quad & (\neg p \vee q) \Rightarrow r \\
\equiv \quad & \neg(\neg p \vee q) \vee r
\end{aligned}
$$

Doing better than Truth Tables

Given a propositional formula, we first express it in Conjunctive Normal Form (ANDs of ORs) as in the following example:

$$
\begin{array}{rl}
 & (p \Rightarrow q) \Rightarrow r \\
\equiv & (\neg p \vee q) \Rightarrow r \\
\equiv & \neg(\neg p \vee q) \vee r \\
\equiv & (\neg\neg p \wedge \neg q) \vee r
\end{array}
$$

Doing better than Truth Tables

Given a propositional formula, we first express it in Conjunctive Normal Form (ANDs of ORs) as in the following example:

$$
\begin{aligned}
& (p \Rightarrow q) \Rightarrow r \\
\equiv\quad & (\neg p \vee q) \Rightarrow r \\
\equiv\quad & \neg(\neg p \vee q) \vee r \\
\equiv\quad & (\neg\neg p \wedge \neg q) \vee r \\
\equiv\quad & (p \wedge \neg q) \vee r
\end{aligned}
$$

Doing better than Truth Tables

Given a propositional formula, we first express it in Conjunctive Normal Form (ANDs of ORs) as in the following example:

$$
\begin{aligned}
& (p \Rightarrow q) \Rightarrow r \\
\equiv \quad & (\neg p \vee q) \Rightarrow r \\
\equiv \quad & \neg(\neg p \vee q) \vee r \\
\equiv \quad & (\neg\neg p \wedge \neg q) \vee r \\
\equiv \quad & (p \wedge \neg q) \vee r \\
\equiv \quad & (p \vee r) \wedge (\neg q \vee r)
\end{aligned}
$$

Learning Clauses with Resolution Rules

Learning Clauses with Resolution Rules

Here is an example of a resolution rule:

$$(p \vee C_1) \wedge (\neg p \vee C_2) \quad \Rightarrow \quad C_1 \vee C_2$$

Learning Clauses with Resolution Rules

Here is an example of a resolution rule:

$$(p \vee C_1) \wedge (\neg p \vee C_2) \quad \Rightarrow \quad C_1 \vee C_2$$

We say that $C_1 \vee C_2$ is the *resolvent* of $(p \vee C_1)$ and $(\neg p \vee C_2)$ over p.

Learning Clauses with Resolution Rules

Here is an example of a resolution rule:

$$(p \vee C_1) \wedge (\neg p \vee C_2) \quad \Rightarrow \quad C_1 \vee C_2$$

We say that $C_1 \vee C_2$ is the *resolvent* of $(p \vee C_1)$ and $(\neg p \vee C_2)$ over p.

Davis-Putnam Algorithm: Given a Conjunctive Normal Form (CNF) formula, repeatedly select a variable, add all resolvents over that variable, and then delete all clauses containing that variable. If you derive a contradiction, then the original formula is unsatisfiable, otherwise a satisfying assignment can be found.

An example of the Davis-Putnam Algorithm

An example of the Davis-Putnam Algorithm

Consider the following CNF formula (here overline means negation):

$$(x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge$$
$$(\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

An example of the Davis-Putnam Algorithm

Consider the following CNF formula (here overline means negation):

$$(x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge$$
$$(\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

Resolving over the variable x_1 yields:

$$(x_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge$$
$$(\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

An example of the Davis-Putnam Algorithm

Consider the following CNF formula (here overline means negation):

$$(x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge$$
$$(\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

Resolving over the variable x_1 yields:

$$(x_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge$$
$$(\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

Resolving over the variable x_2 yields:

$$(\overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (x_3 \vee x_4)$$

An example of the Davis-Putnam Algorithm

Consider the following CNF formula (here overline means negation):

$$(x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge$$
$$(\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

Resolving over the variable x_1 yields:

$$(x_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge$$
$$(\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

Resolving over the variable x_2 yields:

$$(\overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (x_3 \vee x_4)$$

Resolving over the variable x_3 yields:

$$x_4 \wedge \overline{x}_4$$

An example of the Davis-Putnam Algorithm

Consider the following CNF formula (here overline means negation):

$$(x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge$$
$$(\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

Resolving over the variable x_1 yields:

$$(x_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge$$
$$(\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

Resolving over the variable x_2 yields:

$$(\overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (x_3 \vee x_4)$$

Resolving over the variable x_3 yields:

$$x_4 \wedge \overline{x}_4$$

Thus, the original formula is unsatisfiable.

Breakthrough in SAT Solving in the Last 25 Years

Breakthrough in SAT Solving in the Last 25 Years

Mid '90s: Formulas with thousands of variables and clauses were solvable.

Breakthrough in SAT Solving in the Last 25 Years

Mid '90s: Formulas with thousands of variables and clauses were solvable.
Now: Formulas with millions of variables and clauses are solvable.

Breakthrough in SAT Solving in the Last 25 Years

Mid '90s: Formulas with thousands of variables and clauses were solvable.
Now: Formulas with millions of variables and clauses are solvable.

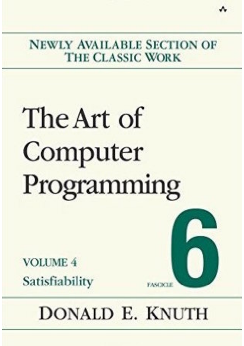
Edmund Clarke: *"a key technology of the 21st century"*

[Biere, Heule, vanMaaren, and Walsh '09]

Breakthrough in SAT Solving in the Last 25 Years

Mid '90s: Formulas with thousands of variables and clauses were solvable.
Now: Formulas with millions of variables and clauses are solvable.

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]

A Combinatorial Problem of Schur

A Combinatorial Problem of Schur

Is it possible to color the integers from 1 to n using colors from $\{$red, blue, green, orange$\}$ so that whenever $a + b = c$, the integers a, b and c don't all have the same color?

A Combinatorial Problem of Schur

Is it possible to color the integers from 1 to n using colors from $\{$red, blue, green, orange$\}$ so that whenever $a + b = c$, the integers a, b and c don't all have the same color?

For small values of n it is possible. Consider, for example, $\{1, 2, 3, 4, 5, 6\}$ which doesn't even use orange.

A Combinatorial Problem of Schur

Is it possible to color the integers from 1 to n using colors from $\{$red, blue, green, orange$\}$ so that whenever $a + b = c$, the integers a, b and c don't all have the same color?

For small values of n it is possible. Consider, for example, $\{1, 2, 3, 4, 5, 6\}$ which doesn't even use orange.

We can list all solutions of $a + b = c$ with $a, b, c \in \{1, 2, 3, 4, 5, 6\}$ and verify that each solution uses at least two colors.

A Combinatorial Problem of Schur

Is it possible to color the integers from 1 to n using colors from $\{$red, blue, green, orange$\}$ so that whenever $a + b = c$, the integers a, b and c don't all have the same color?

For small values of n it is possible. Consider, for example, $\{1, 2, 3, 4, 5, 6\}$ which doesn't even use orange.

We can list all solutions of $a + b = c$ with $a, b, c \in \{1, 2, 3, 4, 5, 6\}$ and verify that each solution uses at least two colors.

$$\begin{array}{lll} 1+1=2 & 1+2=3 & 1+3=4 \\ 1+4=5 & 1+5=6 & 2+2=4 \\ 2+3=5 & 2+4=6 & 3+3=6 \end{array}$$

A Combinatorial Problem of Schur

Is it possible to color the integers from 1 to n using colors from $\{$red, blue, green, orange$\}$ so that whenever $a + b = c$, the integers a, b and c don't all have the same color?

For small values of n it is possible. Consider, for example, $\{1, 2, 3, 4, 5, 6\}$ which doesn't even use orange.

We can list all solutions of $a + b = c$ with $a, b, c \in \{1, 2, 3, 4, 5, 6\}$ and verify that each solution uses at least two colors.

$$\begin{array}{ccc} 1+1=2 & 1+2=3 & 1+3=4 \\ 1+4=5 & 1+5=6 & 2+2=4 \\ 2+3=5 & 2+4=6 & 3+3=6 \end{array}$$

As n gets larger, such colorings will become more difficult to produce; eventually we will need to use orange, and for sufficiently large n such colorings will be impossible to produce. This is a consequence of Schur's Theorem.

Solving a Combinatorial Problem of Schur using SAT

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist.

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist. We introduce variables $x_1, x_2, \ldots, x_{4n}$ where

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist. We introduce variables $x_1, x_2, \ldots, x_{4n}$ where x_{4k} is true iff k,

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist. We introduce variables $x_1, x_2, \ldots, x_{4n}$ where x_{4k} is true iff k, x_{4k-1} is true iff k,

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist. We introduce variables $x_1, x_2, \ldots, x_{4n}$ where x_{4k} is true iff k, x_{4k-1} is true iff k, x_{4k-2} is true iff k,

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist. We introduce variables $x_1, x_2, \ldots, x_{4n}$ where x_{4k} is true iff k, x_{4k-1} is true iff k, x_{4k-2} is true iff k, and x_{4k-3} is true iff k.

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist. We introduce variables $x_1, x_2, \ldots, x_{4n}$ where x_{4k} is true iff k, x_{4k-1} is true iff k, x_{4k-2} is true iff k, and x_{4k-3} is true iff k.

For each $k = 1, 2, \ldots n$ we ensure that k has **at least** one color with the following clause: $x_{4k} \vee x_{4k-1} \vee x_{4k-2} \vee x_{4k-3}$.

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist. We introduce variables $x_1, x_2, \ldots, x_{4n}$ where x_{4k} is true iff k, x_{4k-1} is true iff k, x_{4k-2} is true iff k, and x_{4k-3} is true iff k.

For each $k = 1, 2, \ldots n$ we ensure that k has **at least** one color with the following clause: $x_{4k} \vee x_{4k-1} \vee x_{4k-2} \vee x_{4k-3}$.

For each $k = 1, 2, \ldots n$ we ensure that k has **at most** one color with the following six clauses:

$$\begin{array}{ccc} \neg x_{4k} \vee \neg x_{4k-1} & \neg x_{4k} \vee \neg x_{4k-2} & \neg x_{4k} \vee \neg x_{4k-3} \\ \neg x_{4k-1} \vee \neg x_{4k-2} & \neg x_{4k-1} \vee \neg x_{4k-3} & \neg x_{4k-2} \vee \neg x_{4k-3}. \end{array}$$

Solving a Combinatorial Problem of Schur using SAT

We will use SAT to find the smallest value of n for which such colorings using four colors do not exist. We introduce variables $x_1, x_2, \ldots, x_{4n}$ where x_{4k} is true iff k, x_{4k-1} is true iff k, x_{4k-2} is true iff k, and x_{4k-3} is true iff k.

For each $k = 1, 2, \ldots n$ we ensure that k has **at least** one color with the following clause: $x_{4k} \vee x_{4k-1} \vee x_{4k-2} \vee x_{4k-3}$.

For each $k = 1, 2, \ldots n$ we ensure that k has **at most** one color with the following six clauses:

$$\begin{array}{ccc} \neg x_{4k} \vee \neg x_{4k-1} & \neg x_{4k} \vee \neg x_{4k-2} & \neg x_{4k} \vee \neg x_{4k-3} \\ \neg x_{4k-1} \vee \neg x_{4k-2} & \neg x_{4k-1} \vee \neg x_{4k-3} & \neg x_{4k-2} \vee \neg x_{4k-3}. \end{array}$$

For each solution of $a + b = c$ with $a, b, c \in \{1, \ldots n\}$ we ensure that a, b, and c **don't all have the same color** with the following four clauses:

$$\begin{array}{cc} \neg x_{4a} \vee \neg x_{4b} \vee \neg x_{4c} & \neg x_{4a-1} \vee \neg x_{4b-1} \vee \neg x_{4c-1} \\ \neg x_{4a-2} \vee \neg x_{4b-2} \vee \neg x_{4c-2} & \neg x_{4a-3} \vee \neg x_{4b-3} \vee \neg x_{4c-3}. \end{array}$$

Mathematicians are Interested in Machine-Assisted Proofs

ipam

Machine Assisted Proofs

FEBRUARY 13 - 17, 2023

ORGANIZING COMMITTEE

Erika Abraham (RWTH Aachen University)
Jeremy Avigad (Carnegie Mellon University)
Kevin Buzzard (Imperial College London)
Jordan Ellenberg (University of Wisconsin-Madison)
Tim Gowers (College de France)
Marijn Heule (Carnegie Mellon University)
Terence Tao (University of California, Los Angeles (UCLA))

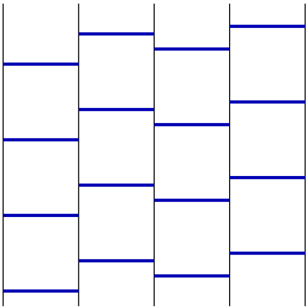
nature

NEWS | 18 June 2021

Mathematicians welcome computer-assisted proof in 'grand unification' theory

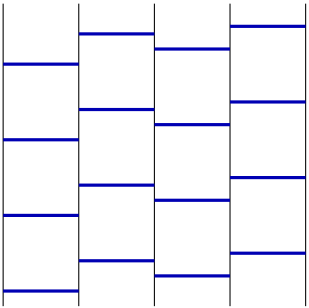
Keller's Conjecture: A Tiling Problem

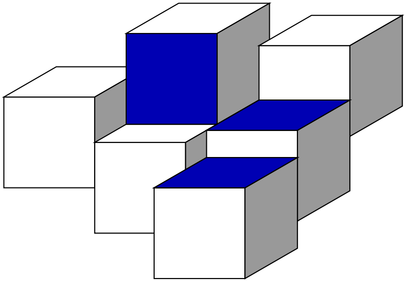
Consider tiling a floor with square tiles, all of the same size. Is it the case that any gap-free tiling results in at least two fully connected tiles, i.e., tiles that have an entire edge in common?



Keller's Conjecture: A Tiling Problem

Consider tiling a floor with square tiles, all of the same size. Is it the case that any gap-free tiling results in at least two fully connected tiles, i.e., tiles that have an entire edge in common?



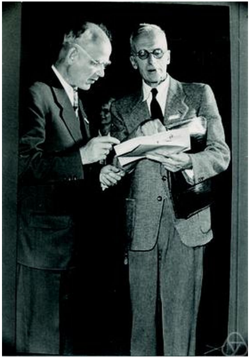


Keller's Conjecture: Resolved

[Brakensiek, Heule, Mackey, & Narvaez 2019]

In 1930, Ott-Heinrich Keller conjectured that this phenomenon holds in every dimension.

Keller's Conjecture.
For all $n \geq 1$, every tiling of the n-dimensional space with unit cubes has two which fully share a face.



[Wikipedia, CC BY-SA]

GEOMETRY

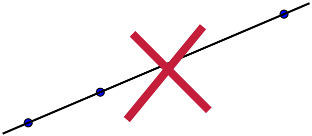
Computer Search Settles 90-Year-Old Math Problem

10

By translating Keller's conjecture into a computer-friendly search for a type of graph, researchers have finally resolved a problem about covering spaces with tiles.

An Empty Hexagon in Every Set of 30 Points

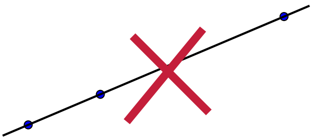
Computational geometry and SAT: Shapes in point sets in general position (no three points on a line)



k-hole: empty k-point convex shape

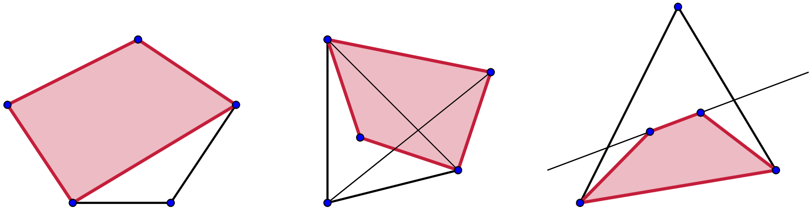
An Empty Hexagon in Every Set of 30 Points

Computational geometry and SAT: Shapes in point sets in general position (no three points on a line)



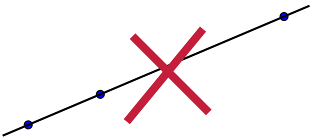
k-hole: empty k-point convex shape

Every set of 5 points contains in a 4-hole [Klein 1932]



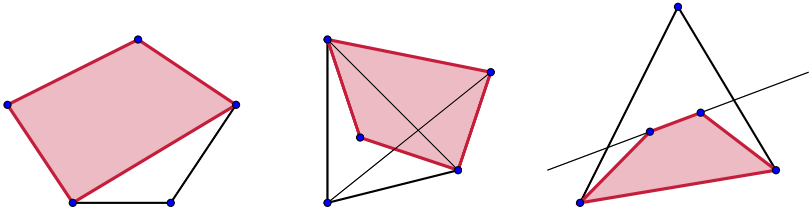
An Empty Hexagon in Every Set of 30 Points

Computational geometry and SAT: Shapes in point sets in general position (no three points on a line)



k-hole: empty k-point convex shape

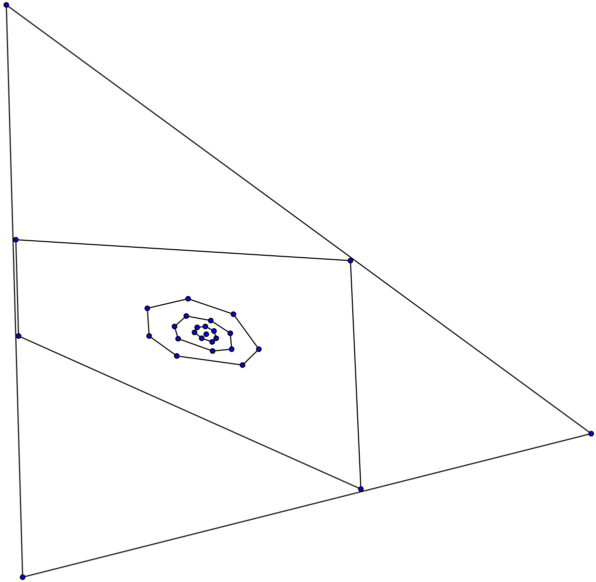
Every set of 5 points contains in a 4-hole [Klein 1932]



Every set of 30 points contains in a 6-hole (using SAT)

[Heule & Scheucher 2023]

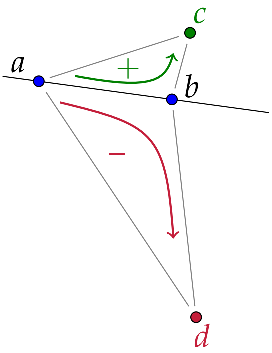
Avoiding an Empty Hexagon in a Set of 29 Points



SAT Encoding: Orientation Variables

No explicit coordinates of points

Instead, for every triple $a < b < c$, one orientation variable $O_{a,b,c}$ to denote whether point c is above the line ab



Not all assignments are realizable

- Axioms eliminate many unrealizable assignments

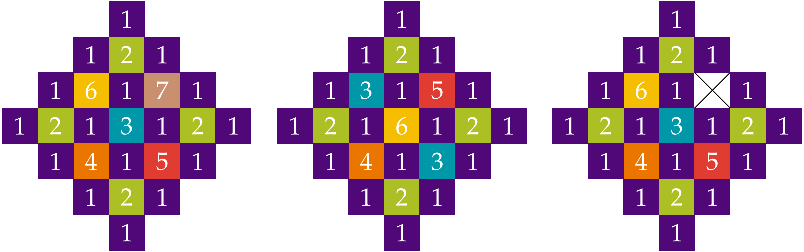
Many possible SAT encodings

- Big impact on performance
- Machine learning can help!

Packing Chromatic Number

Definition

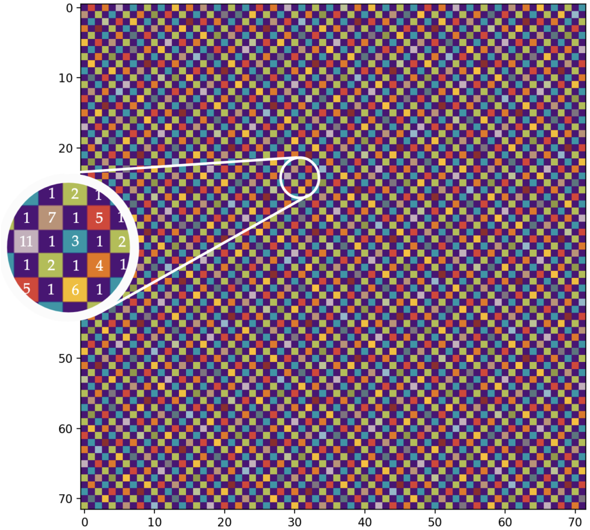
A packing k-coloring of a simple undirected graph $G = (V, E)$ is a function φ from V to $\{1, \ldots, k\}$ such that for any two distinct vertices $u, v \in V$, and any color $c \in \{1, \ldots, k\}$, it holds that $\varphi(u) = \varphi(v) = c$ implies $d(u, v) > c$.



Packing Chromatic Number of the Infinite Grid is 15

The 72×72 15-coloring below can be used to tile the infinite grid

- This is not possible with 14 colors [Subercaseaux & Heule'23]

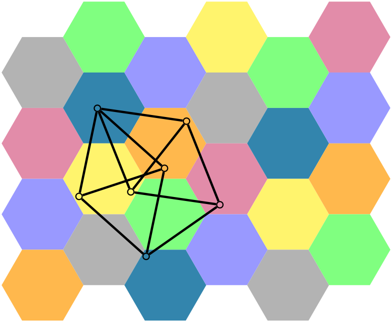


Chromatic Number of the Plane (CNP)

The Hadwiger-Nelson problem (around 1950):
How many colors are required to color the plane such that each pair of points that are exactly 1 apart are colored differently?

Chromatic Number of the Plane (CNP)

The Hadwiger-Nelson problem (around 1950):
How many colors are required to color the plane such that each pair of points that are exactly 1 apart are colored differently?



- The Moser Spindle graph shows the lower bound of 4
- A coloring of the plane showing the upper bound of 7

CNP: First progress in decades

Recently enormous progress:

- Lower bound of 5 [DeGrey '18] based on a 1581-vertex graph
- This breakthrough started a polymath project
- Improved bounds of the fractional chromatic number of the plane

CNP: First progress in decades

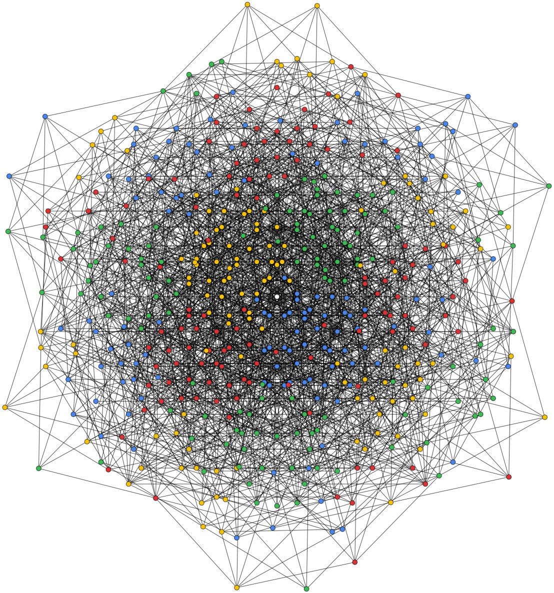
Recently enormous progress:

- Lower bound of 5 [DeGrey '18] based on a 1581-vertex graph
- This breakthrough started a polymath project
- Improved bounds of the fractional chromatic number of the plane

We found smaller graphs with SAT:

- 874 vertices on April 14, 2018
- 803 vertices on April 30, 2018
- 610 vertices on May 14, 2018

Proof Minimization: 510 Vertices [Heule 2021]



Beyond NP: The Collatz Conjecture

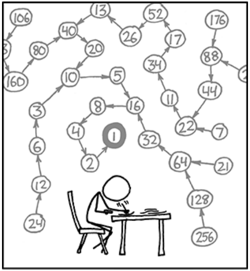
Resolving foundational algorithm questions

$$Col(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ (3n+1)/2 & \text{if } n \text{ is odd} \end{cases}$$

Does `while`$(n > 1)$ $n = Col(n)$; terminate?

Find a non-negative function $fun(n)$ s.t.

$$\forall n > 1 : fun(n) > fun(Col(n))$$



THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

source: xkcd.com/710

Beyond NP: The Collatz Conjecture

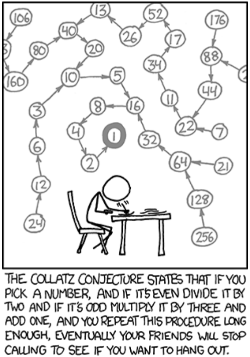
Resolving foundational algorithm questions

$$Col(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ (3n+1)/2 & \text{if } n \text{ is odd} \end{cases}$$

Does `while` $(n > 1)\ n = Col(n);$ terminate?

Find a non-negative function $fun(n)$ s.t.

$$\forall n > 1 : fun(n) > fun(Col(n))$$



source: xkcd.com/710

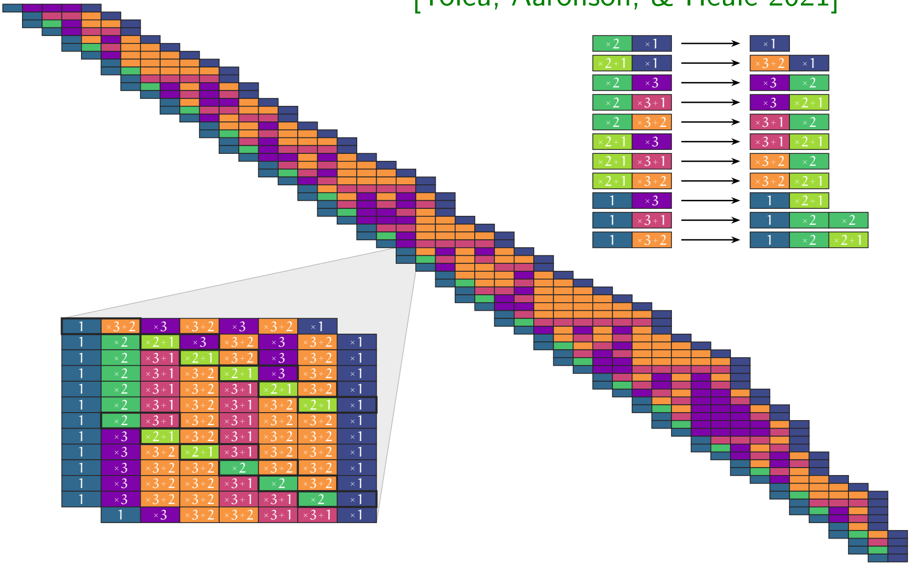
Can we construct a function s.t. $fun(n) > fun(Col(n))$ holds?

$fun(3)$	$fun(5)$	$fun(8)$	$fun(4)$	$fun(2)$	$fun(1)$
5	4	3	2	1	0

Collatz Conjecture: Studying a Rewrite System

[Yolcu, Aaronson, & Heule 2021]

×2 ×1	→	×1
×2+1 ×1	→	×3+2 ×1
×2 ×3	→	×3 ×2
×2 ×3+1	→	×3 ×2+1
×2 ×3+2	→	×3+1 ×2
×2+1 ×3	→	×3+1 ×2+1
×2+1 ×3+1	→	×3+2 ×2
×2+1 ×3+2	→	×3+2 ×2+1
1 ×3	→	1 ×2+1
1 ×3+1	→	1 ×2 ×2
1 ×3+2	→	1 ×2 ×2+1



Collatz Conjecture: Successes and Challenge

Success. Rewrite system with 11 rules: Their termination solves Collatz. Our tool proves termination of any subset of 10 rules.

Collatz Conjecture: Successes and Challenge

Success. Rewrite system with 11 rules: Their termination solves Collatz. Our tool proves termination of any subset of 10 rules.

Success. Our tool proves termination of Farkas' variant:

$$F(n) = \begin{cases} \frac{n-1}{3} & \text{if } n \equiv 1 \pmod 3 \\ \frac{n}{2} & \text{if } n \equiv 0 \text{ or } n \equiv 2 \pmod 6 \\ \frac{3n+1}{2} & \text{if } n \equiv 3 \text{ or } n \equiv 5 \pmod 6 \end{cases}$$

Collatz Conjecture: Successes and Challenge

Success. Rewrite system with 11 rules: Their termination solves Collatz. Our tool proves termination of any subset of 10 rules.

Success. Our tool proves termination of Farkas' variant:

$$F(n) = \begin{cases} \frac{n-1}{3} & \text{if } n \equiv 1 \pmod 3 \\ \frac{n}{2} & \text{if } n \equiv 0 \text{ or } n \equiv 2 \pmod 6 \\ \frac{3n+1}{2} & \text{if } n \equiv 3 \text{ or } n \equiv 5 \pmod 6 \end{cases}$$

Challenge (\$500). An easier generalized Collatz problem is open:

$$H(n) = \begin{cases} \frac{3n}{4} & \text{if } n \equiv 0 \pmod 4 \\ \frac{9n+1}{8} & \text{if } n \equiv 7 \pmod 8 \\ \perp & \text{otherwise} \end{cases}$$

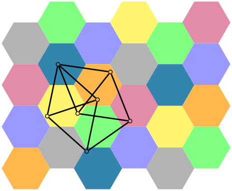
Conclusions

Successes, Advances, and Trust:

- A performance boost of SAT technology allows solving new problems in mathematics
- Problems beyond NP are ready for an automated approach
- Some proofs may be gigantic, but can be validated using formally-verified checkers

Classic problems ready for mechanization?

- Chromatic number of the plane
- Optimal matrix multiplication
- Collatz Conjecture



One More Thing: Costas Arrays