# MATHCHECK2: COMBINING LEARNING-BASED SEARCH (SAT) WITH SYMBOLIC COMPUTATION (CAS)

**Vijay Ganesh, Curtis Bright,**
**Albert Heinle, Ilias Kotsireas, Krzysztof Czarnecki**
**University of Waterloo, Canada**

**Sept 24, 2016**
**SC^2 Workshop, Timisoara, Romania**

# HOW TO SOLVE A SET OF MATHEMATICAL CONSTRAINTS

- The symbolic method
  - Formula manipulation: set of sound and hopefully complete rules
  - Completeness: desirable but not always achievable
  - Efficiency: the method may not be efficient for many interesting fragments of mathematics
  - Examples: Computer algebra systems, decision procedures for arithmetic, word equations,…

# HOW TO SOLVE A SET OF MATHEMATICAL CONSTRAINTS

- The search method
  - Search: systematically enumerate all models until termination condition is met
  - Gained relevance thanks to ultra-fast modern computers
  - Complete only for finite search spaces
  - Surprisingly efficient provided it is combined with learning, e.g., CDCL Boolean SAT solvers

# What is a SAT/SMT Solver?

# Automation of Mathematical Logic


- Rich logics (Modular arithmetic, arrays, strings, non-linear arithmetic, theories with quantifiers, ...)
- From proof procedures to validity to satisfiability
- SAT problem is NP-complete, PSPACE-complete,...
- Practical, scalable, usable, automatic
- Enable novel software reliability approaches

# THE BOOLEAN SATISFIABILITY PROBLEM
# SOME STANDARD DEFINITIONS

- A **literal** $p$ is a Boolean variable $x$ or its negation $\neg x$. A **clause** $C$ is a disjunction of literals: $x_2 \vee \neg x_{41} \vee x_{15}$
- A **CNF** is a conjunction of clauses: $(x_2 \vee \neg x_1 \vee x_5) \wedge (x_6 \vee \neg x_2) \wedge (x_3 \vee \neg x_4 \vee \neg x_6)$
- An **assignment** is a mapping from variables to Boolean values **(True, False).** A **unit clause** $C$ is a clause with a single unbound literal
- The Boolean **SAT problem** is
  - Find an assignment such that each input clause has a true literal (aka input formula is SAT) OR establish that input formula has no solution (aka input formula is UNSAT)
  - SAT solvers are required to output a solution if input is SAT (many solvers also produce a proof if input is UNSAT)
- Boolean formulas are typically represented in **DIMACS Format**

# DPLL SAT SOLVER ARCHITECTURE
# THE BASIC SOLVER

**DPLL($\Theta_{cnf}$, assign) {**

Propagate unit clauses;

**if** *"conflict"*: **return** FALSE;

**if** *"complete assign"*: **return** TRUE;

*"pick decision variable x"*;

**Return**

DPLL($\Theta_{cnf}$ $|_{x=0}$, assign[x=0])

||

DPLL($\Theta_{cnf}$ $|_{x=1}$, assign[x=1]);

**}**

Key Steps in a DPLL SAT Solver

Propagate (Boolean Constant Propagation)

- Propagate inferences due to unit clauses
- Most of solving "effort" goes into this step

Detect Conflict

- Conflict: partial assignment is not satisfying

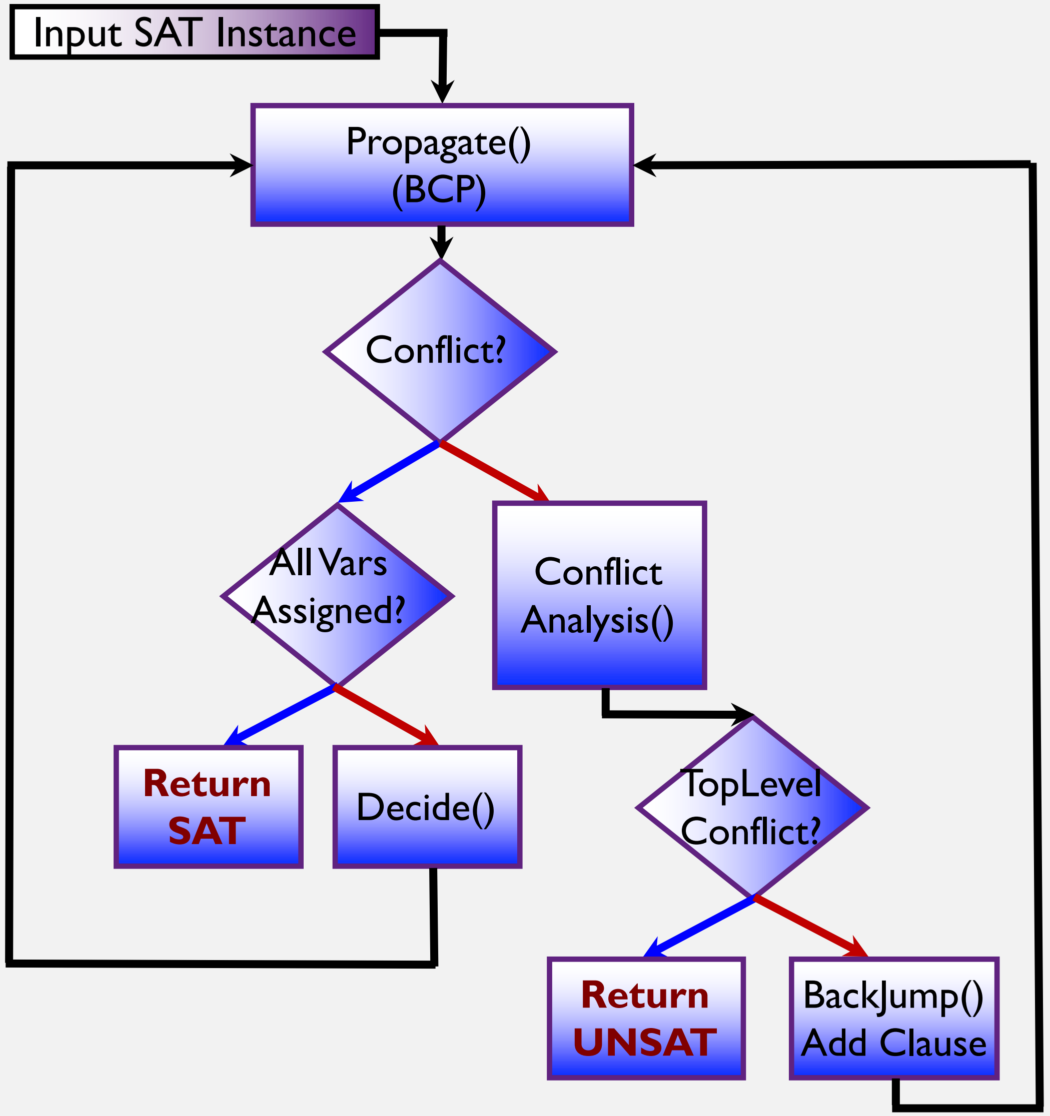
Decide (Branch)

- Choose a variable & assign some value

Backtracking

- Implicitly done via recursive calls in DPLL

# MODERN CDCL SAT SOLVER ARCHITECTURE
# KEY STEPS AND DATA-STRUCTURES




Key steps

- Decide()
- Propagate() (Boolean constant propagation)
- Conflict analysis and learning() (CDCL)
- Backjump()
- Forget()
- Restart()

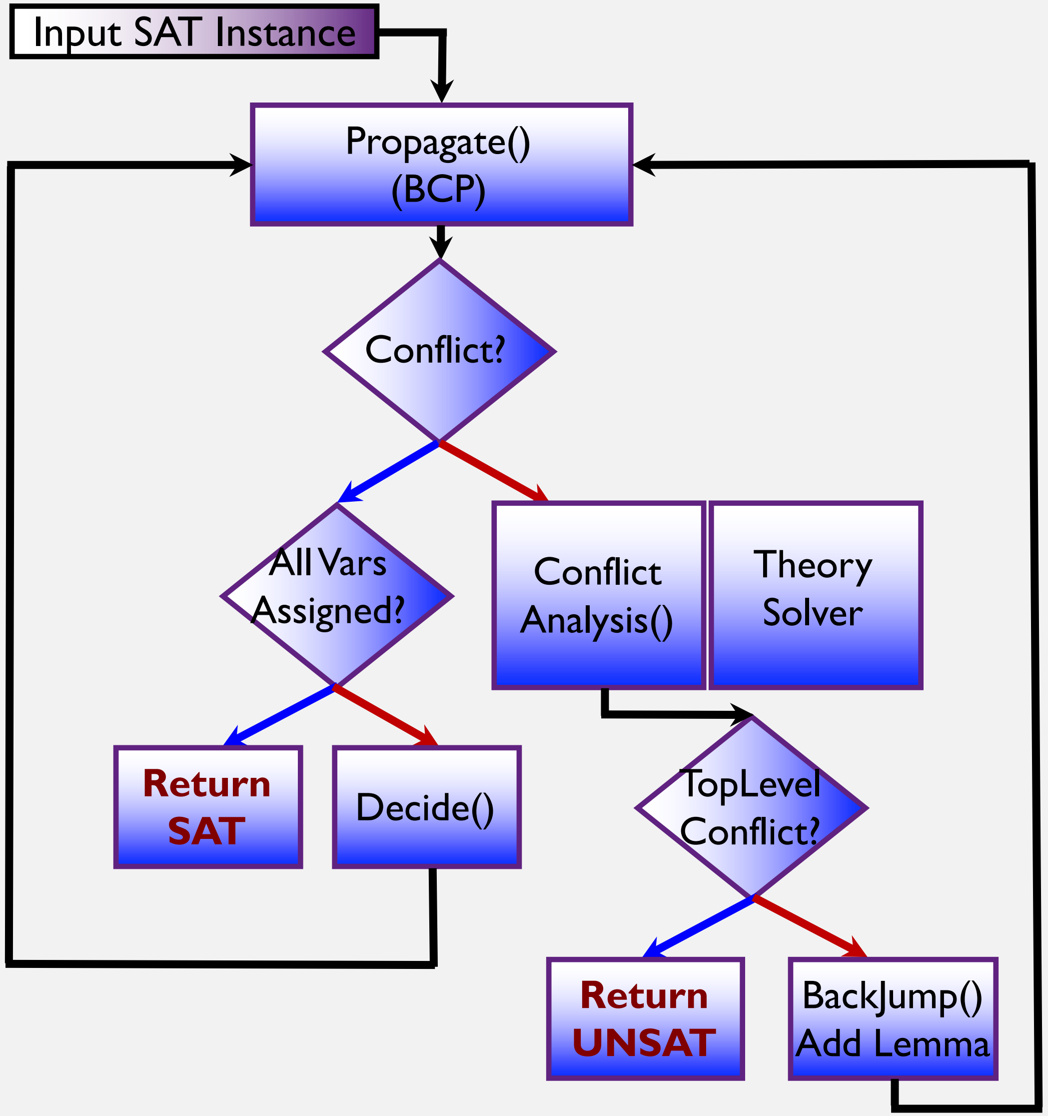
CDCL: Conflict-Driven Clause-Learning

- Conflict analysis is a key step
- Results in learning a learnt clause
- Prunes the search space

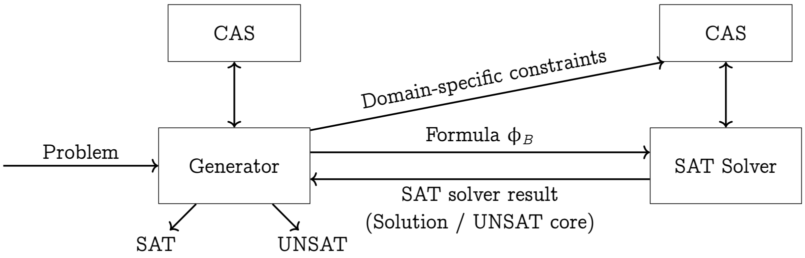
Key data-structures (Solver state)

- Stack or trail of partial assignments (AT)
- Input clause database
- Conflict clause database
- Conflict graph
- Decision level (DL) of a variable

# MODERN CDCL(T)
# PROGRAMMATIC SAT, CDCL(CAS)



# The MathCheck2 System



# Conjectures studied by MathCheck

- **Ruskey–Savage conjecture** (1993): Any matching of a hypercube can be extended to a Hamiltonian cycle.

Our result: Conjecture holds for hypercubes of dimension $d \leqslant 5$.

- **Norine conjecture** (2008): There always exists a monochromatic path between two antipodal vertices in an edge-antipodal coloring of a hypercube.

Our result: Conjecture holds for hypercubes of dimension $d \leqslant 6$.

- **Hadamard conjecture** (1893): Hadamard matrices exist for all orders divisible by 4.

Our result: Williamson-generated Hadamard matrices exist for all orders $4n$ with $n < 35$ but not for $n = 35$.

- **Complex Golay conjecture** (2002): Complex Golay sequences do not exist for order 23.

Our result: Confirmation of the conjecture (computations in progress).

# Hadamard matrices

- square matrix with $\pm 1$ entries
- any two distinct rows are orthogonal

# Hadamard matrices

- square matrix with $\pm 1$ entries
- any two distinct rows are orthogonal

## Example

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & 1 & -1 \end{bmatrix}$$

## Conjecture

An $n \times n$ Hadamard matrix exists for any $n$ a multiple of 4.

# Williamson Matrices

- $n \times n$ matrices $A$, $B$, $C$, $D$
- entries $\pm 1$
- symmetric, circulant
- $A^2 + B^2 + C^2 + D^2 = 4nI_n$

# Symmetric and Circulant Matrices

Such matrices are defined by their first $\lceil \frac{n+1}{2} \rceil$ entries so we may refer to them as if they were sequences.

## Examples ($n = 5$ and $6$)

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_2 & a_1 \\ a_1 & a_0 & a_1 & a_2 & a_2 \\ a_2 & a_1 & a_0 & a_1 & a_2 \\ a_2 & a_2 & a_1 & a_0 & a_1 \\ a_1 & a_2 & a_2 & a_1 & a_0 \end{bmatrix}$$

symmetric conditions

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_1 & a_2 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_1 & a_2 & a_3 \\ a_3 & a_2 & a_1 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_2 & a_1 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_2 & a_1 & a_0 \end{bmatrix}$$

circulant conditions

# Symmetric and Circulant Matrices

Such matrices are defined by their first $\left\lceil \frac{n+1}{2} \right\rceil$ entries so we may refer to them as if they were sequences.

## Examples ($n = 5$ and $6$)

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_2 & a_1 \\ a_1 & a_0 & a_1 & a_2 & a_2 \\ a_2 & a_1 & a_0 & a_1 & a_2 \\ a_2 & a_2 & a_1 & a_0 & a_1 \\ a_1 & a_2 & a_2 & a_1 & a_0 \end{bmatrix} \qquad \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_1 & a_2 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_1 & a_2 & a_3 \\ a_3 & a_2 & a_1 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_2 & a_1 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_2 & a_1 & a_0 \end{bmatrix}$$

symmetric conditions      circulant conditions

# Williamson ~~Matrices~~ Sequences

- sequences $A$, $B$, $C$, $D$ of length $\lceil \frac{n+1}{2} \rceil$
- entries $\pm 1$
- $\mathrm{PAF}_A(s) + \mathrm{PAF}_B(s) + \mathrm{PAF}_C(s) + \mathrm{PAF}_D(s) = 0$ for $s = 1, \ldots, \lceil \frac{n-1}{2} \rceil$.

The PAF[1] here is defined

$$\mathrm{PAF}_A(s) \coloneqq \sum_{k=0}^{n-1} a_k a_{(k+s) \bmod n}.$$

[1] Periodic Autocorrelation Function

# Results

MathCheck2 was able to show that...

- Williamson matrices of order 35 do not exist.
  - First shown by Đoković[2], who requested an independent verification.
- Williamson matrices exist for all orders $n < 35$.
  - Even orders were mostly previously unstudied.
- Found over 160 Hadamard matrices which were not previously in the library of the CAS Magma.
  - Orders up to $168 \times 168$.

[2] Williamson matrices of order $4n$ for $n = 33, 35, 39$. *Discrete Mathematics.*

# Example: Williamson Sequences of Order 3

- Objective: Find $\pm 1$ values for the variables $a_0$, $a_1$, $b_0$, $b_1$, $c_0$, $c_1$, $d_0$, $d_1$ which satisfy the constraint

$$a_0 a_1 + b_0 b_1 + c_0 c_1 + d_0 d_1 + 2 = 0.$$

# Linearize the Problem

- Let $p_0 \coloneqq a_0 a_1$, $p_1 \coloneqq b_0 b_1$, $p_2 \coloneqq c_0 c_1$, and $p_3 \coloneqq d_0 d_1$.
- The constraint now becomes

$$p_0 + p_1 + p_2 + p_3 + 2 = 0.$$

# Rewrite as a Cardinality Constraint

- Since $p_0 + p_1 + p_2 + p_3 + 2 = 0$ and each $p_i \in \{\pm 1\}$, we can determine that

$$\#\{\, i : p_i = 1 \,\} = 1 \qquad \text{and} \qquad \#\{\, i : p_i = -1 \,\} = 3.$$

# Determining a Conflict Clause

- Say the SAT solver finds a partial assignment with $\{p_0 = 1, p_1 = -1, p_2 = 1\}$.
- Since $\#\{\ i : p_i = 1\ \} > 1$, we know that this assignment can never result in an actual solution to the problem.
- We tell the SAT solver to learn the constraint

$$\neg\big(\{p_0 = 1\} \wedge \{p_2 = 1\}\big).$$

# Example: Using Filtering Theorems

- Consider now the larger problem with the 36 variables $a_0$, $a_1$, $a_2$, $a_3$, $a_4$, $a_5$, $b_0$, ..., $d_5$.
- Given an assignment to all of the $a_i$ variables, we can form the symmetric sequence

$$[a_0, a_1, a_2, a_3, a_4, a_5, a_5, a_4, a_3, a_2, a_1]$$

and possibly filter (i.e., discard) the assignment using its *power spectral density*.

# Sample PSD Calculation

- Say we have the assignment with $\{a_0 = a_1 = a_2 = 1, a_3 = a_4 = a_5 = -1\}$.
- The power spectral density of

$$A \coloneqq [1, 1, 1, -1, -1, -1, -1, -1, -1, 1, 1]$$

can be computed to be approximately

$$[1, 49.37, 1.09, 5.79, 1.41, 2.33, 2.33, 1.41, 5.79, 1.09, 49.37].$$

# Đoković–Kotsireas Filtering Theorem

- A theorem of Đoković–Kotsireas says that a sequence cannot be Williamson if it has a PSD value larger than 4 times the length of the sequence.
- One PSD value of $A$ was $49.37 > 44$ and therefore we can tell the SAT solver to learn the filtering constraint

$$\neg\big(\{a_0 = a_1 = a_2 = 1\} \land \{a_3 = a_4 = a_5 = -1\}\big).$$

# Average Timings (in Seconds)

| Order | CAS Preprocessor | CAS Preprocessor + CDCL(CAS) |
|---|---|---|
| 24 | 0.01 | 0.01 |
| 26 | 0.09 | 0.08 |
| 28 | 0.06 | 0.05 |
| 30 | 0.48 | 0.28 |
| 32 | 0.04 | 0.05 |
| 34 | 2.69 | 1.51 |
| 36 | 0.83 | 0.75 |
| 38 | 10.62 | 6.08 |
| 40 | 1.02 | 1.08 |
| 42 | 112.51 | 42.21 |

# Conclusions

- We have demonstrated the power of the SAT+CAS combination by
  - performing a requested verification of a nonexistence result
  - establishing the existence of Williamson matrices of even orders up to 42
  - generating new matrices for MAGMA's Hadamard database.
- We are working on extending the system to search for other types of combinatorial objects.
- Our system is free software and available at

  `sites.google.com/site/uwmathcheck`