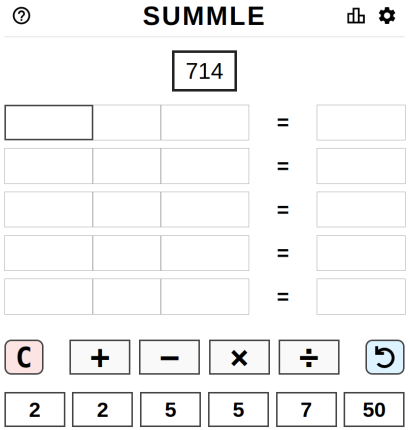
Solving Puzzles (Place to calculate 714!)



Satisfiability Testing

Recent Developments and Open Problems

Norbert Manthey

nmanthey@conp-solutions.com

Introduction

- Who Am I?
  - PhD in parallel SAT [Man14b], advancing solvers full time
  - Success in international competitions
    - SAT: Riss [Man14a], Pcasso [ILM14], MergeSat [Man21b] family
    - MaxSAT: open-wbo
    - HWMCC [hwm15]: ShiftBMC
  - Since 2016: *hacking from couch* [FMSS20b, FMSS20a, Man21b]
    - `github/conp-solutions` [Man21a, Man21b, FMSS20b]
    - `twitter/norbert_manthey`
    - `linkedinin/norbert-manthey-5378b983`
- Audience experience with SAT and solvers?

Introduction

- Who Am I?
  - PhD in parallel SAT [Man14b], advancing solvers full time
  - Success in international competitions
    - SAT: Riss [Man14a], Pcasso [ILM14], MergeSat [Man21b] family
    - MaxSAT: open-wbo
    - HWMCC [hwm15]: ShiftBMC
  - Since 2016: *hacking from couch* [FMSS20b, FMSS20a, Man21b]
    - `github/conp-solutions` [Man21a, Man21b, FMSS20b]
    - `twitter/norbert_manthey`
    - `linkedinin/norbert-manthey-5378b983`
- Audience experience with SAT and solvers?

Introduction

- Who Am I?
  - PhD in parallel SAT [Man14b], advancing solvers full time
  - Success in international competitions
    - SAT: Riss [Man14a], Pcasso [ILM14], MergeSat [Man21b] family
    - MaxSAT: open-wbo
    - HWMCC [hwm15]: ShiftBMC
  - Since 2016: *hacking from couch* [FMSS20b, FMSS20a, Man21b]
    - `github/conp-solutions` [Man21a, Man21b, FMSS20b]
    - `twitter/norbert_manthey`
    - `linkedinin/norbert-manthey-5378b983`
- Audience experience with SAT and solvers?

Solving Puzzles – Using Solver

- Parse Puzzle
  - Translate to High Order Logic (C program)
    - Translate to SAT/SMT/... $\rightarrow$ solve
    - Run with test input
  - Convert solution

Solving Puzzles – Using Solver

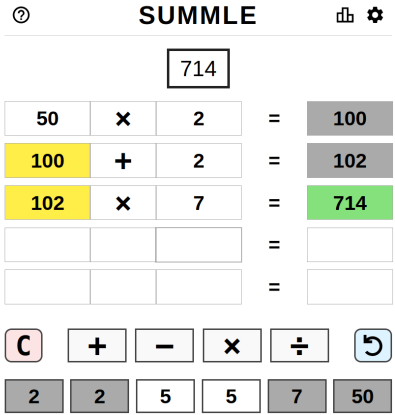
- Parse Puzzle
  - Translate to High Order Logic (C program)
    - Translate to SAT/SMT/$\ldots \rightarrow$ solve
    - Run with test input
  - Convert solution

Solving Puzzles – Example Formulas

```
p cnf 7 7
-1 2 0
-2 3 0
-3 1 0
2 7 6 0
2 -7 -6 0
-6 7 0
-7 6 0
```

```
p cnf 7 10
1 3 0
-3 7 0
-5 -2 0
-1 -7 0
-7 -3 0
1 5 3 0
3 -5 2 0
5 7 -2 0
5 7 2 0
-7 -3 6 0
```

Solving Puzzles – Solution



Outline

Solving Puzzles
SAT – Foundations
SAT – Open Areas
Summary

SAT Introduction

Satisfiability Testing

- Propositional Logic (conjunction of clauses)
  - *Clause*: *disjunction*, non-trivial set of *literals* (integers)
  - *Formula*: *conjunction* (*multiset*) of clauses (list of sets)

- Applying Truth Values
  - Interpretation: complement-free set of literals (collision free hashmap + stack)
  - $C = \{1, 2, 3, 4\}$, $I = \{-1, -3\}$, $C|_I = \{2, 4\}$ ($\{1.2, 3.4\}$)
  - SAT solver does not actually modify clauses
  - Actual simplification is performed separately

Satisfiability Testing

- Propositional Logic (conjunction of clauses)
  - *Clause*: *disjunction*, non-trivial set of *literals* (integers)
  - *Formula*: *conjunction* (*multiset*) of clauses (list of sets)

- Applying Truth Values
  - Interpretation: complement-free set of literals (collision free hashmap + stack)
  - $C = \{1, 2, 3, 4\}$, $I = \{-1, -3\}$, $C|_I = \{2, 4\}$ ($\{1.2, 3.4\}$)
  - SAT solver does not actually modify clauses
  - Actual simplification is performed separately

Satisfiability Testing

- Propositional Logic (conjunction of clauses)
  - *Clause*: *disjunction*, non-trivial set of *literals* (integers)
  - *Formula*: *conjunction* (*multiset*) of clauses (list of sets)

- Applying Truth Values
  - Interpretation: complement-free set of literals (collision free hashmap + stack)
  - $C = \{1, 2, 3, 4\}$, $I = \{-1, -3\}$, $C|_I = \{2, 4\}$ ($\{1, 2, 3, 4\}$)
  - SAT solver does not actually modify clauses
  - Actual simplification is performed separately

Satisfiability Testing

- Propositional Logic (conjunction of clauses)
  - *Clause*: *disjunction*, non-trivial set of *literals* (integers)
  - *Formula*: *conjunction* (*multiset*) of clauses (list of sets)

- Applying Truth Values
  - Interpretation: complement-free set of literals (collision free hashmap + stack)
  - $C = \{1, 2, 3, 4\}$, $I = \{-1, -3\}$, $C|_I = \{2, 4\}$ ($\{1, 2, 3, 4\}$)
  - SAT solver does not actually modify clauses
  - Actual simplification is performed separately

Satisfiability Testing

- **Is a formula satisfiable?**
  - What is the *model*?
  - Why is it not satisfiable?
  - Given a formula, which ask applies?

Satisfiability Testing

- **Is a formula satisfiable?**
  - What is the *model*?
  - Why is it not satisfiable?
  - Given a formula, which ask applies?

Satisfiability Testing

- **Is a formula satisfiable?**
  - What is the *model*?
  - Why is it not satisfiable?
  - Given a formula, which ask applies?

Satisfiability Testing

- **Is a formula satisfiable?**
  - What is the *model*?
  - Why is it not satisfiable?
  - Given a formula, which ask applies?

Satisfiability Testing – SAT

- Assign a truth value to each variable
  - Via stochastic local search (SLS) [BHvMW21]
  - Incremental tree-search (backtracking) [DLL62]
    - Heuristically (all false, most frequent, . . .)
    - Re-Use previous polarities (*phase-saving*)
    - Re-calculate polarities to be used (*rephase*)
    - Restarting
- **Best case: linear**

Satisfiability Testing – SAT

- Assign a truth value to each variable
  - Via stochastic local search (SLS) [BHvMW21]
  - Incremental tree-search (backtracking) [DLL62]
    - Heuristically (all false, most frequent, . . .)
    - Re-Use previous polarities (*phase-saving*)
    - Re-calculate polarities to be used (*rephase*)
    - Restarting
- Best case: linear

Satisfiability Testing – SAT

- Assign a truth value to each variable
  - Via stochastic local search (SLS) [BHvMW21]
  - Incremental tree-search (backtracking) [DLL62]
    - Heuristically (all false, most frequent, . . .)
    - Re-Use previous polarities (*phase-saving*)
    - Re-calculate polarities to be used (*rephase*)
    - Restarting
- **Best case: linear**

Satisfiability Testing – Search Example

p cnf 7 10

C_{01}: 1 3 0

C_{02}: -3 7 0

C_{03}: -5 -2 0

C_{04}: -1 -7 0

C_{05}: -7 -3 0

C_{06}: 1 5 3 0

C_{07}: 3 -5 2 0

C_{08}: 5 7 -2 0

C_{09}: 5 7 2 0

C_{10}: -7 -3 6 0

- Pick a literal, 7, propagate
  1. $C_{05} \rightarrow -3\ 0$
  2. $C_{04} \rightarrow -1\ 0$
  3. $C_{10} \rightarrow \top$ (thanks to -3)
- Propagate literal -3
  1. $C_{01} \rightarrow \bot$ (conflict)
- Learn: $-7\ 0$ ($C_{01} \otimes C_{04} \otimes C_{05}$)
- Add learned clause as **redundant** clause

Satisfiability Testing – Search Example

p cnf 7 10

C_{01}: 1 3 0

C_{02}: -3 7 0

C_{03}: -5 -2 0

C_{04}: -1 -7 0

C_{05}: -7 -3 0

C_{06}: 1 5 3 0

C_{07}: 3 -5 2 0

C_{08}: 5 7 -2 0

C_{09}: 5 7 2 0

C_{10}: -7 -3 6 0

- Pick a literal, 7, propagate
  1. $C_{05} \rightarrow -3\ 0$
  2. $C_{04} \rightarrow -1\ 0$
  3. $C_{10} \rightarrow \top$ (thanks to -3)
- Propagate literal -3
  1. $C_{01} \rightarrow \bot$ (conflict)
- Learn: $-7\ 0$ ($C_{01} \otimes C_{04} \otimes C_{05}$)
- Add learned clause as **redundant** clause

Satisfiability Testing – Search Example

p cnf 7 10

C_{01}: 1 3 0

C_{02}: -3 7 0

C_{03}: -5 -2 0

C_{04}: -1 -7 0

C_{05}: -7 -3 0

C_{06}: 1 5 3 0

C_{07}: 3 -5 2 0

C_{08}: 5 7 -2 0

C_{09}: 5 7 2 0

C_{10}: -7 -3 6 0

- Pick a literal, 7, propagate
  1. $C_{05} \rightarrow -3\ 0$
  2. $C_{04} \rightarrow -1\ 0$
  3. $C_{10} \rightarrow \top$ (thanks to -3)
- Propagate literal -3
  1. $C_{01} \rightarrow \bot$ (conflict)
- Learn: $-7\ 0$ ($C_{01} \otimes C_{04} \otimes C_{05}$)
- Add learned clause as **redundant** clause

Satisfiability Testing – Search Example

p cnf 7 10

C_{01}: 1 3 0

C_{02}: -3 7 0

C_{03}: -5 -2 0

C_{04}: -1 -7 0

C_{05}: -7 -3 0

C_{06}: 1 5 3 0

C_{07}: 3 -5 2 0

C_{08}: 5 7 -2 0

C_{09}: 5 7 2 0

C_{10}: -7 -3 6 0

- Pick a literal, 7, propagate
  1. $C_{05} \rightarrow -3\ 0$
  2. $C_{04} \rightarrow -1\ 0$
  3. $C_{10} \rightarrow \top$ (thanks to -3)
- Propagate literal -3
  1. $C_{01} \rightarrow \bot$ (conflict)
- Learn: $-7\ 0$ ($C_{01} \otimes C_{04} \otimes C_{05}$)
- Add learned clause as **redundant** clause

Satisfiability Testing – Search Example

p cnf 7 10

C_{01}: 1 3 0

C_{02}: -3 7 0

C_{03}: -5 -2 0

C_{04}: -1 -7 0

C_{05}: -7 -3 0

C_{06}: 1 5 3 0

C_{07}: 3 -5 2 0

C_{08}: 5 7 -2 0

C_{09}: 5 7 2 0

C_{10}: -7 -3 6 0

- Pick a literal, 7, propagate
  1. $C_{05} \rightarrow -3\ 0$
  2. $C_{04} \rightarrow -1\ 0$
  3. $C_{10} \rightarrow \top$ (thanks to -3)
- Propagate literal -3
  1. $C_{01} \rightarrow \bot$ (conflict)
- Learn: $-7\ 0$ ($C_{01} \otimes C_{04} \otimes C_{05}$)
- Add learned clause as **redundant** clause

Satisfiability Testing – Search Example

p cnf 7 10

C_{01}: 1 3 0

C_{02}: -3 7 0

C_{03}: -5 -2 0

C_{04}: -1 -7 0

C_{05}: -7 -3 0

C_{06}: 1 5 3 0

C_{07}: 3 -5 2 0

C_{08}: 5 7 -2 0

C_{09}: 5 7 2 0

C_{10}: -7 -3 6 0

- Pick a literal, 7, propagate
  1. $C_{05} \rightarrow -3\ 0$
  2. $C_{04} \rightarrow -1\ 0$
  3. $C_{10} \rightarrow \top$ (thanks to -3)
- Propagate literal -3
  1. $C_{01} \rightarrow \bot$ (conflict)
- Learn: $-7\ 0$ ($C_{01} \otimes C_{04} \otimes C_{05}$)
- Add learned clause as **redundant** clause

Satisfiability Testing – UNSAT

- Resolve clauses
  - Which clauses to resolve next?
  - Variable elimination [DP60] – exponential
- Use a different representation
- Introduce new variables – which ones?
- Separate field: **proof complexity** [BN21]

Satisfiability Testing – UNSAT

- Resolve clauses
  - Which clauses to resolve next?
  - Variable elimination [DP60] – exponential
- Use a different representation
- Introduce new variables – which ones?
- Separate field: **proof complexity** [BN21]

Satisfiability Testing – UNSAT

- Resolve clauses
  - Which clauses to resolve next?
  - Variable elimination [DP60] – exponential
- Use a different representation
- Introduce new variables – which ones?
- Separate field: **proof complexity** [BN21]

Example Proof

p cnf 7 10

C_{01}: 1 3 0
C_{02}: -3 7 0
C_{03}: -5 -2 0
C_{04}: -1 -7 0
C_{05}: -7 -3 0
C_{06}: 1 5 3 0
C_{07}: 3 -5 2 0
C_{08}: 5 7 -2 0
C_{09}: 5 7 2 0
C_{10}: -7 -3 6 0

RUP proof (with resolution steps)

C^1_{11}: -3 0 $(C_{02} \otimes C_{05})$
C^1_{12}: -3 6 0 $(C_{02} \otimes C_{10})$
C^1_{13}: -5 3 0 $(C_{03} \otimes C_{07})$
C^1_{14}: -2 7 0 $(C_{03} \otimes C_{08})$
C^1_{15}: 5 7 0 $(C_{08} \otimes C_{09})$
C^2_{16}: 1 0 $(C_{01} \otimes C_{11})$
C^2_{17}: -5 0 $(C_{11} \otimes C_{13})$
C^3_{18}: -7 0 $(C_{04} \otimes C_{16})$ or $(C_{01} \otimes C_{05} \otimes C_{04})$
C^4_{19}: 0 $(C_{15} \otimes C_{17} \otimes C_{18})$

Notation for a clause: C^{depth}_{step}

This proof: length: 9, width: 3, depth: 4

Satisfiability Testing – Algorithm

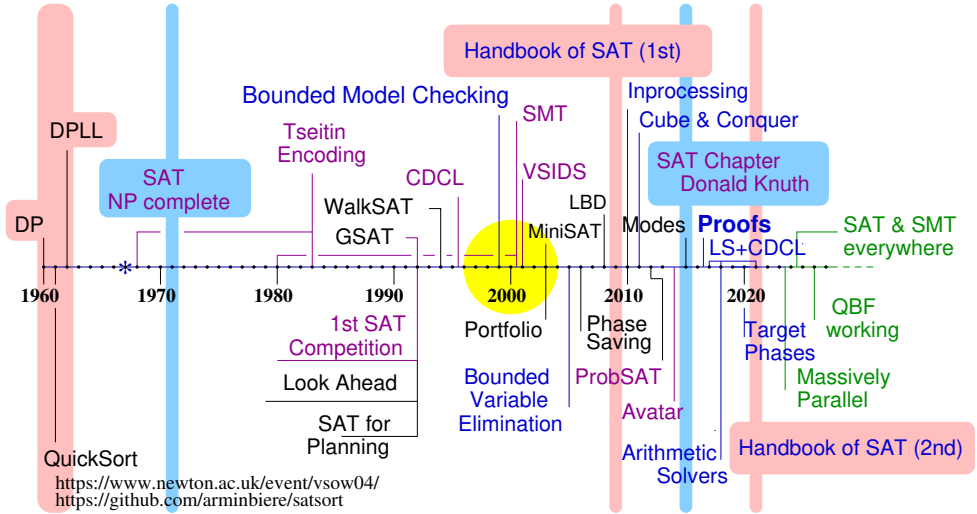
- **while** no solution
  1. Propagate assignment, imply unit clauses
     - Conflict? learn redundant resolvent for more propagation, jump-back
  2. Simplify formula?
  3. Reduce redundant clauses?
  4. Rephase search polarities?
  5. Restart search?
  6. Decide search literal heuristically

Satisfiability Testing – Algorithm

- **while** no solution
  1. Propagate assignment, imply unit clauses
     - Conflict? learn redundant resolvent for more propagation, jump-back
  2. Simplify formula?
  3. Reduce redundant clauses?
  4. Rephase search polarities?
  5. Restart search?
  6. Decide search literal heuristically

Satisfiability Testing – Algorithm

- **while** no solution
  1. Propagate assignment, imply unit clauses
     - Conflict? learn redundant resolvent for more propagation, jump-back
  2. Simplify formula?
  3. Reduce redundant clauses?
  4. Rephase search polarities?
  5. Restart search?
  6. Decide search literal heuristically



https://www.newton.ac.uk/event/vsow04/
https://github.com/arminbiere/satsort

(with permission from Armin Biere)

Why Is SAT Used?

- Simple representation
  - effective data structures [MMZ+01]
  - fast algorithms
- Translation into SAT uses additional variables
  - improves UNSAT results (see proof complexity)
- Development driven by annual competition [BFH+21]

SAT – More Details

- Past KRR Lectures, thesis and projects
- Parallel SAT Solving [Man14b]
- Videos from Simon's Institute lectures (`talks/sat-solving`)
- Handbook of SAT v2 [BHvMW21, BHvMW09]
- TAOCP by Don Knuth, SAT Chapter [Knu06]
- SAT competition solver descriptions [BFH+21]
- SAT solver source code

SAT – More Details

- Past KRR Lectures, thesis and projects
- Parallel SAT Solving [Man14b]
- Videos from Simon's Institute lectures (`talks/sat-solving`)
- Handbook of SAT v2 [BHvMW21, BHvMW09]
- TAOCP by Don Knuth, SAT Chapter [Knu06]
- SAT competition solver descriptions [BFH+21]
- SAT solver source code

SAT – More Details

- Past KRR Lectures, thesis and projects
- Parallel SAT Solving [Man14b]
- Videos from Simon's Institute lectures (`talks/sat-solving`)
- Handbook of SAT v2 [BHvMW21, BHvMW09]
- TAOCP by Don Knuth, SAT Chapter [Knu06]
- SAT competition solver descriptions [BFH+21]
- SAT solver source code

SAT – Open Areas

SAT – Open Areas

- Areas
  - Proof Complexity Related
  - Parallel Solving
  - Application
  - Implementation and Hardware

Related to Proof Complexity

- Which clauses to resolve?
  - Which redundant clauses to keep (or share)?
  - How to find the shortest proof?
  - Learn from proof properties to steer search?
- How to represent the input? (have framework to auto-select)
  - Unary, binary or order encoding [TTKB09]?
  - Propagation complete encodings
- Introduce variables while solving? Which?
  - Representing conjunctions [AKS10]
  - Detect semantically, from recent proof?

Related to Proof Complexity

- Which clauses to resolve?
  - Which redundant clauses to keep (or share)?
  - How to find the shortest proof?
  - Learn from proof properties to steer search?
- How to represent the input? (have framework to auto-select)
  - Unary, binary or order encoding [TTKB09]?
  - Propagation complete encodings
- Introduce variables while solving? Which?
  - Representing conjunctions [AKS10]
  - Detect semantically, from recent proof?

Related to Proof Complexity

- Which clauses to resolve?
  - Which redundant clauses to keep (or share)?
  - How to find the shortest proof?
  - Learn from proof properties to steer search?
- How to represent the input? (have framework to auto-select)
  - Unary, binary or order encoding [TTKB09]?
  - Propagation complete encodings
- Introduce variables while solving? Which?
  - Representing conjunctions [AKS10]
  - Detect semantically, from recent proof?

Parallel Solving

- Show that conflict analysis is inherently sequential
- Parallelize core algorithm, e.g. unit propagation (for 4 cores) [Man11]
- Portfolio solving vs partitioning [ILM13, Man14b]
  - Portfolio is limited wrt configurations and proofs
  - Static vs dynamic, recursive partitioning; **how**?
  - What to share?
  - How to search in parallel? [GHJS10, ILM13, Man21b]
  - Purpose build vs code reuse

Parallel Solving

- Show that conflict analysis is inherently sequential
- Parallelize core algorithm, e.g. unit propagation (for 4 cores) [Man11]
- Portfolio solving vs partitioning [ILM13, Man14b]
  - Portfolio is limited wrt configurations and proofs
  - Static vs dynamic, recursive partitioning; **how**?
  - What to share?
  - How to search in parallel? [GHJS10, ILM13, Man21b]
  - Purpose build vs code reuse

Parallel Solving

- Show that conflict analysis is inherently sequential
- Parallelize core algorithm, e.g. unit propagation (for 4 cores) [Man11]
- Portfolio solving vs partitioning [ILM13, Man14b]
  - Portfolio is limited wrt configurations and proofs
  - Static vs dynamic, recursive partitioning; **how**?
  - What to share?
  - How to search in parallel? [GHJS10, ILM13, Man21b]
  - Purpose build vs code reuse

SAT Applications

- Process larger input, handle more complex functions
- Effective incremental interface
- Effective encoding vs formula size
  - Effective representation
  - Propagation complete
- Experimental, sharing negative results
- Feedback from application owners, high and low level benchmarks

SAT Applications

- Process larger input, handle more complex functions
- Effective incremental interface
- Effective encoding vs formula size
  - Effective representation
  - Propagation complete

- Experimental, sharing negative results
- Feedback from application owners, high and low level benchmarks

SAT Applications

- Process larger input, handle more complex functions
- Effective incremental interface
- Effective encoding vs formula size
  - Effective representation
  - Propagation complete
- Experimental, sharing negative results
- Feedback from application owners, high and low level benchmarks

Compute Hardware and SAT

- How to make solvers work better on hardware?
  - How to build data structures? [MMZ+01, HMS10]
  - How to exploit more hardware properties/features? [HMS10, FMSS20b]
  - Same rules for sequential and parallel?
- On which hardware does SAT work well? SpecSAT

Compute Hardware and SAT

- How to make solvers work better on hardware?
  - How to build data structures? [MMZ+01, HMS10]
  - How to exploit more hardware properties/features? [HMS10, FMSS20b]
  - Same rules for sequential and parallel?
- On which hardware does SAT work well? `SpecSAT`

My Recent Activities

- MergeSat, combining enhancements of other solvers
  - Stable interface (MiniSat, IPASIR, DRUP proof)
  - Automated (Unit tests, CI, ...)
  - Deterministic parallel solving
  - Parallel proof, clause sharing
  - WIP: Recursive search space partitioning and (lazy) prallel incremental solving
- WIP: SpecSAT – Benchmark hardware for SAT
  - Comparable score for sequential and parallel solving per platform
  - Have competitive, parallel, cross-platform deterministic, solver

My Recent Activities

- MergeSat, combining enhancements of other solvers
  - Stable interface (MiniSat, IPASIR, DRUP proof)
  - Automated (Unit tests, CI, ...)
  - Deterministic parallel solving
  - Parallel proof, clause sharing
  - WIP: Recursive search space partitioning and (lazy) prallel incremental solving
- WIP: SpecSAT – Benchmark hardware for SAT
  - Comparable score for sequential and parallel solving per platform
  - Have competitive, parallel, cross-platform deterministic, solver

My Recent Activities

- MergeSat, combining enhancements of other solvers
  - Stable interface (MiniSat, IPASIR, DRUP proof)
  - Automated (Unit tests, CI, ...)
  - Deterministic parallel solving
  - Parallel proof, clause sharing
  - WIP: Recursive search space partitioning and (lazy) prallel incremental solving
- WIP: SpecSAT – Benchmark hardware for SAT
  - Comparable score for sequential and parallel solving per platform
  - Have competitive, parallel, cross-platform deterministic, solver

Summary

Propositional Logic is Relevant

- Works well for many applications
  - No publishing of bad results
  - Hard to find the current weaknesses
- Many open areas
  - Heuristics and algorithms for applications
  - Implementation improvements
  - Proof complexity driven change
  - Better parallel solving

Propositional Logic is Relevant

- Works well for many applications
  - No publishing of bad results
  - Hard to find the current weaknesses
- Many open areas
  - Heuristics and algorithms for applications
  - Implementation improvements
  - Proof complexity driven change
  - Better parallel solving

Open SAT Areas

- Solver development needs a good mix of
  - Systems knowledge
  - Algorithm knowledge
  - Automation
  - Compute resources
- Run SpecSAT and share results!

[AKS10] Gilles Audemard, George Katsirelos, and Laurent Simon.
A restriction of extended resolution for clause learning SAT solvers.
In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[BFH+21] Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors.
volume B-2021-1 of *Department of Computer Science Report Series B*.
University of Helsinki, Helsinki, Finland, 2021.

[BHvMW09] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors.
Handbook of Satisfiability.
IOS Press, Amsterdam, 2009.

[BHvMW21] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors.
Handbook of Satisfiability - Second Edition, volume 336 of *Frontiers in Artificial Intelligence and Applications*.
IOS Press, 2021.

[BN21] Sam Buss and Jakob Nordström.
Proof complexity and SAT solving.
In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 233–350. IOS Press, 2021.

[DLL62] Martin Davis, George Logemann, and Donald Loveland.
A machine program for theorem-proving.
Communications of the ACM, 5(7):394–397, 1962.

[DP60] M. Davis and H. Putnam.
A computing procedure for quantification theory.
JACM, 7(3):201–215, 1960.

[FMSS20a] Johannes K. Fichte, Norbert Manthey, Julian Stecklina, and André Schidler.
Towards faster reasoners by using transparent huge pages.
In Helmut Simonis, editor, *Principles and Practice of Constraint Programming*, pages 304–322, Cham, 2020. Springer International Publishing.

[FMSS20b] Johannes K. Fichte, Norbert Manthey, Julian Stecklina, and André Schidler.
Towards faster reasoners by using transparent huge pages, 2020.

[GHJS10] Long Guo, Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais.
Diversification and intensification in parallel SAT solving.
In *CP 2010*, volume 6308 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2010.

[HMS10] S. Hölldobler, N. Manthey, and A. Saptawijaya.
Improving resource-unaware SAT solvers.
volume 6397 of *LNCS*, pages 519–534, Heidelberg, 2010. Springer.

[hwm15] Hardware model checking competition.
`fmv.jku.at/hwmcc15`, October 2015.

[ILM13] Ahmed Irfan, Davide Lanti, and Norbert Manthey.
PCASSO – a parallel cooperative SAT solver, 2013.

[ILM14] Ahmed Irfan, Davide Lanti, and Norbert Manthey.
PCASSO – a parallel cooperative SAT solver, 2014.

[Knu06] Donald E. Knuth.
The Art of Computer Programming, Volume 4, Fascicle 4: Generating All Trees–History of Combinatorial Generation (Art of Computer Programming).
Addison-Wesley Professional, 2006.

[Man11] Norbert Manthey.
A More Efficient Parallel Unit Propagation.
Technical report, TU Dresden, Knowledge Representation and Reasoning, 2011.

[Man14a] Norbert Manthey.
Riss 4.27.
In *SAT Competition*, 2014.

[Man14b] Norbert Manthey.
Towards Next Generation Sequential and Parallel SAT Solvers.
PhD thesis, TU Dresden, 2014.

[Man21a] Norbert Manthey.
Cadical modification – watch sat.
In Balyo et al. [BFH+21], pages 28–29.

[Man21b] Norbert Manthey.
The mergesat solver.
In Chu-Min Li and Felip Manyà, editors, *Theory and Applications of Satisfiability Testing - SAT 2021 - 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings*, volume 12831 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 2021.

[MMZ+01] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik.
Chaff: Engineering an efficient SAT solver.
In *DAC 2001*, pages 530–535, New York, 2001. ACM.

[TTKB09] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara.
Compiling finite linear CSP into SAT.
Constraints, 14(2):254–272, 2009.