Boolean Satisfiability: From Theoretical Hardness to Practical Success

Sharad Malik

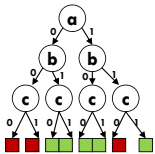
Princeton University

SAT in a Nutshell

- Given a Boolean formula, find a variable assignment such that the formula evaluates to 1, or prove that no such assignment exists.

$$\mathbf{F = (a + b)(a' + b' + c)}$$

- For *n* variables, there are 2^n possible truth assignments to be checked.

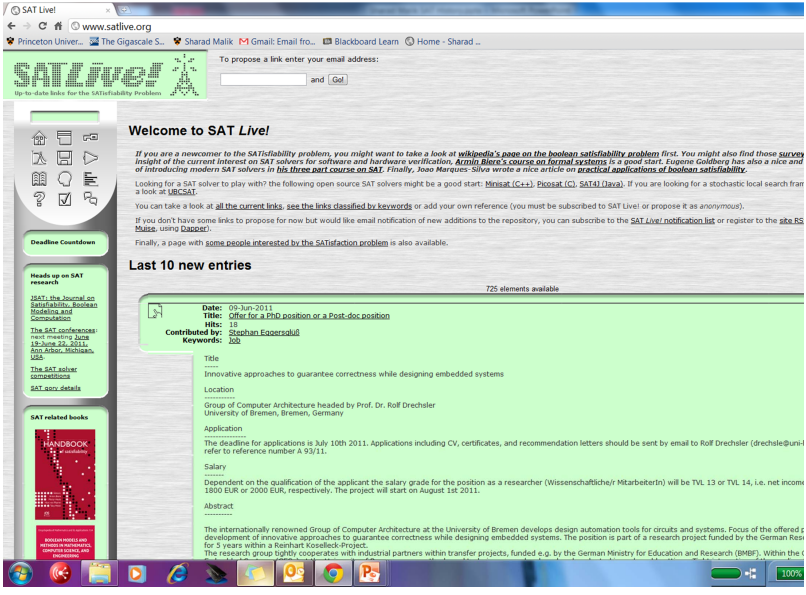


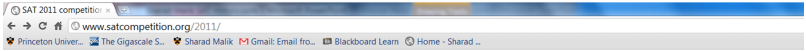
- First established NP-Complete problem.

  S. A. Cook, The complexity of theorem proving procedures, *Proceedings, Third Annual ACM Symp. on the Theory of Computing*,1971, 151-158

Where are we today?

- Intractability of the problem no longer daunting
  - Can regularly handle practical instances with millions of variables and constraints
- SAT has matured from theoretical interest to practical impact
  - Electronic Design Automation (EDA)
    - Widely used in many aspects of chip design
  - Increasing use in software verification
    - Commercial use at Microsoft, NEC,…


SAT Live!
www.satlive.org
Princeton Univer...
The Gigascale S...
Sharad Malik
Gmail: Email fro...
Blackboard Learn
Home - Sharad ...
SAT Live!
Up-to-date links for the SATisfiability Problem
To propose a link enter your email address:
and
Go!
Welcome to SAT Live!
If you are a newcomer to the SATisfiability problem, you might want to take a look at wikipedia's page on the boolean satisfiability problem first. You might also find those survey
Insight of the current interest on SAT solvers for software and hardware verification, Armin Biere's course on formal systems is a good start. Eugene Goldberg has also a nice and
of introducing modern SAT solvers in his three part course on SAT. Finally, Joao Marques-Silva wrote a nice article on practical applications of boolean satisfiability.
Looking for a SAT solver to play with? the following open source SAT solvers might be a good start: Minisat (C++), Picosat (C), SAT4J (Java). If you are looking for a stochastic local search fram
a look at UBCSAT.
You can take a look at all the current links, see the links classified by keywords or add your own reference (you must be subscribed to SAT Live! or propose it as anonymous).
If you don't have some links to propose for now but would like email notification of new additions to the repository, you can subscribe to the SAT Live! notification list or register to the site RS
Mutrie, using Dapper!.
Finally, a page with some people interested by the SATisfaction problem is also available.
Deadline Countdown
Heads up on SAT research
JSAT: the Journal on Satisfiability, Boolean Modeling and Computation
The SAT conferences: next meeting June 19-June 22, 2011, Ann Arbor, Michigan, USA
The SAT solver competitions
SAT gory details
SAT related books
HANDBOOK of satisfiability
BOOLEAN MODELING METHODS IN MATHEMATICS, COMPUTER SCIENCE, AND ENGINEERING
Last 10 new entries
725 elements available
Date: 09-Jun-2011
Title: Offer for a PhD position or a Post-doc position
Hits: 18
Contributed by: Stephan Eggersglüß
Keywords: job
Title
Innovative approaches to guarantee correctness while designing embedded systems
Location
Group of Computer Architecture headed by Prof. Dr. Rolf Drechsler
University of Bremen, Bremen, Germany
Application
The deadline for applications is July 10th 2011. Applications including CV, certificates, and recommendation letters should be sent by email to Rolf Drechsler (drechsle@uni-
refer to reference number A 93/11.
Salary
Dependent on the qualification of the applicant the salary grade for the position as a researcher (Wissenschaftliche/r Mitarbeiter/in) will be TVL 13 or TVL 14, i.e. net income
1800 EUR or 2000 EUR, respectively. The project will start on August 1st 2011.
Abstract
The internationally renowned Group of Computer Architecture at the University of Bremen develops design automation tools for circuits and systems. Focus of the offered p
development of innovative approaches to guarantee correctness while designing embedded systems. The position is part of a research project funded by the German Rese
for 5 years within a Reinhart Koselleck-Project.
The research group tightly cooperates with industrial partners within transfer projects, funded e.g. by the German Ministry for Education and Research (BMBF). Within the
100%



SAT Competition 2011

A competitive event of the SAT 2011 Conference

June 19th - June 22nd 2011, Ann Arbor, MI, USA

Last modification: $LastChangedDate: 2011-04-25 21:14:21 +0200 (Mon, 25 Apr 2011) $.

Quick links

Sponsors

The SAT competition is organized thanks to our generous sponsors:

Registration

Register and submit your solver or benchmark

What's new this year?

There are several new features in the SAT competition this year:

New Hardware

The competition will run on a new cluster at CRIL, composed of nodes with two Bi-Xeon Quad core processors and 32 GB of RAM. The operating system is CentOS 5.4, x86 competition is run in two stages. During the first stage, solvers will be allocated 7GB of memory. Each instance of a sequential solver will be allocated 2 cores, each instance of a p cores. This means that at one time, a node will be running either 4 runs of a sequential solver (2 per processor), or 2 runs of a parallel solver (1 per processor). Two different solv node. For the second stage - based on the results of which the best solvers will be awarded - only one sequential solver will be launched on each processor (2 solvers per node) launched on a node (1 solver per node), with a memory limit of 15GB of RAM.

Sequential/Parallel Neutrality

This year, there is no special track dedicated to sequential or parallel solvers. Sequential and parallel solvers are grouped into one single competition, but with two different rankin wall clock time and will promote solvers which use all available resources to give an answer as quickly as possible. The second ranking is based on CPU time and will promote s efficiently as possible. This latter ranking is the one that was used in the previous competitions. In the wall clock based ranking, timeout will be imposed on the wall clock time. In will be imposed on CPU time. It is expected that parallel solvers will perform better in the first ranking while sequential solvers will perform better in the second ranking.

New Award Categories

The competition will award both the fastest SAT solvers in terms of wall-clock time and in terms of CPU time. The most innovative ("non CDCL") SAT solver will be awarded a

Choose Your Category

Unlike the previous competitions, in which all solvers where run on all benchmarks, in order to save computational resources this time submitters are asked to select in which cate random) their solver will compete. The most efficient solvers (selected by the jury) will still compete in every category during the second stage.

Minimally Unsatisfiable Subset (MUS) Special Track

Due to the success of MUS techniques on various applications (especially as core engines in MAXSAT solvers), a special track for MUS systems will be organized for the first ti

Data Analysis Track

Since there are many different ways to analyze the results of the competition, the Data Analysis Track will offer to anyone the possibility to run its own analysis of the competition *the competition web site and as a poster during the SAT conference.* This track is an opportunity to experiment different ranking schemes, as well as analyze the strengths and benchmarks. Contributors will have to submit a program that will be run by the organizers on anonymized results.

Competition tracks

Here is a quick view of the competition. See detailed rules for complete details.

Main track

Where are we today? (contd.)

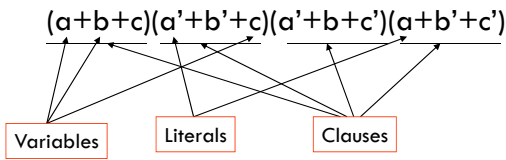
- Significant SAT community
  - SatLive Portal and SAT competitions
  - SAT Conference
- Emboldened researchers to take on even harder problems
  - Satisfiability Modulo Theories (SMT)
  - Max-SAT
  - Quantified Boolean Formulas (QBF)

SAT Solvers: A Condensed History

- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by “resolution”
- Backtrack Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Exhaustive search for satisfying assignment
- Conflict Driven Clause Learning [CDCL]
  - GRASP: Integrate a constraint learning procedure, 1996
- Locality Based Search
  - Emphasis on exhausting local sub-spaces, e.g. Chaff, Berkmin, miniSAT and others, 2001 onwards
  - Added focus on efficient implementation
- “Pre-processing”
  - Peephole optimization, e.g. miniSAT, 2005

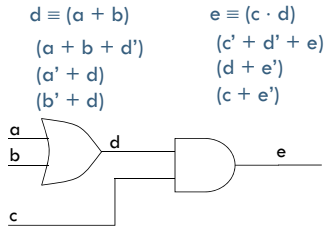
Problem Representation

- Conjunctive Normal Form
  - Representation of choice for modern SAT solvers



Circuit to CNF Conversion

- Tseitin Transformation



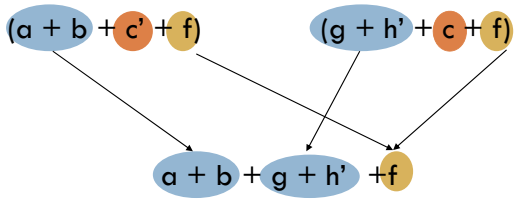
Consistency conditions for circuit variables

- Can 'e' ever become true?

Is (e)(a + b + d')(a'+d)(b'+d)(c'+d+e)(d+e')(c+e') satisfiable?

Resolution

- Resolution of a pair of distance-one clauses

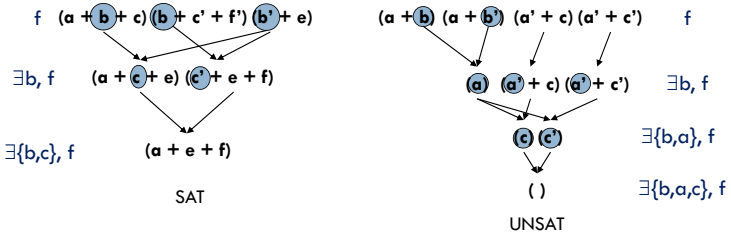


Resolvent implied by the original clauses

Davis Putnam Algorithm

M .Davis, H. Putnam, "A computing procedure for quantification theory", *J. of ACM*, Vol. 7, pp. 201-214, 1960

- Iterative existential quantification of variables



Potential memory explosion problem!

SAT Solvers: A Condensed History

- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by "resolution"
- Backtrack Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Exhaustive search for satisfying assignment
- Conflict Driven Clause Learning [CDCL]
  - GRASP: Integrate a constraint learning procedure, 1996
- Locality Based Search
  - Emphasis on exhausting local sub-spaces, e.g. Chaff, Berkmin, miniSAT and others, 2001 onwards
  - Added focus on efficient implementation
- "Pre-processing"
  - Peephole optimization, e.g. miniSAT, 2005

Basic DLL Search

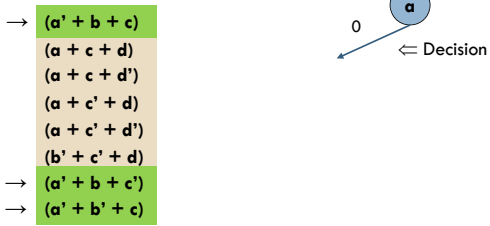
(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM, 5:394–397, 1962*

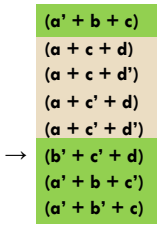
Basic DLL Search

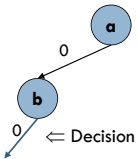
(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

Basic DLL Search

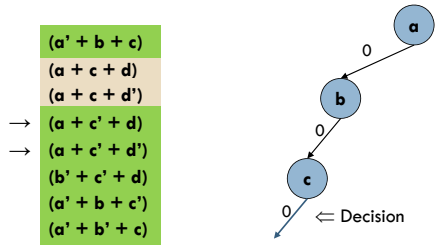


Basic DLL Search

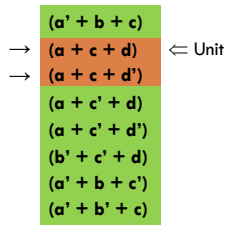


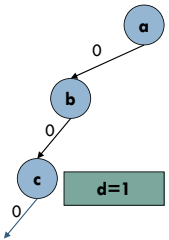


Basic DLL Search



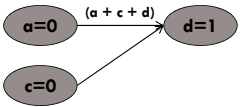
Basic DLL Search



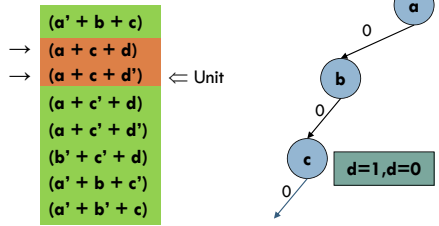


Unit Clause Rule

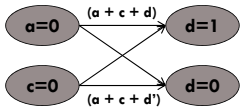
Implication Graph



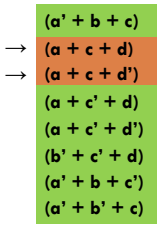
Basic DLL Search

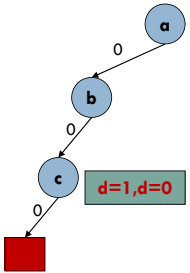


Implication Graph

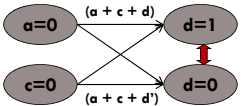


Basic DLL Search



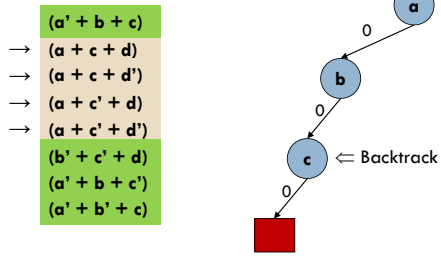


Implication Graph

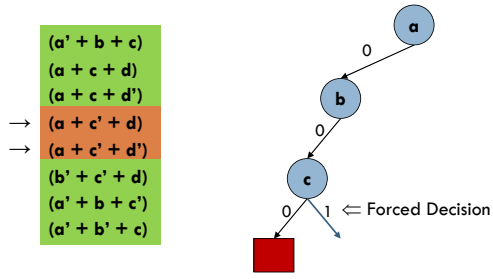


Conflict!

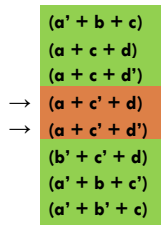
Basic DLL Search

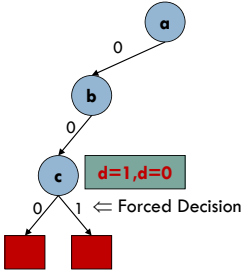


Basic DLL Search

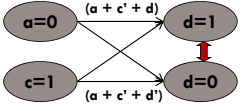


Basic DLL Search



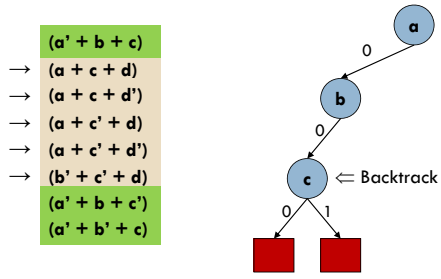


Implication Graph



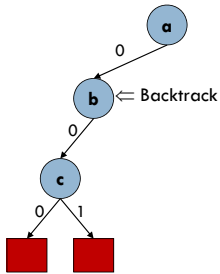
Conflict!

Basic DLL Search



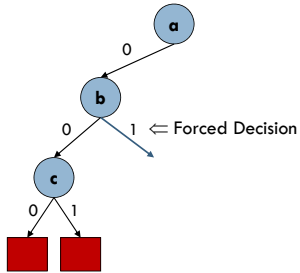
Basic DLL Search

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

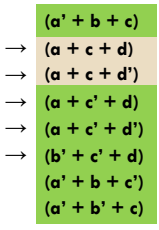


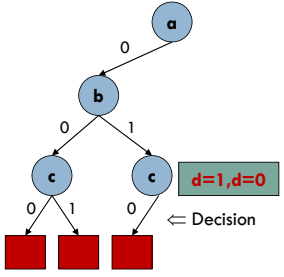
Basic DLL Search

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

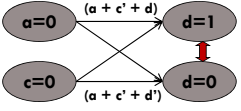


Basic DLL Search



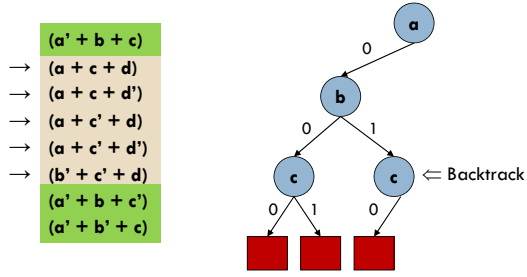


Implication Graph

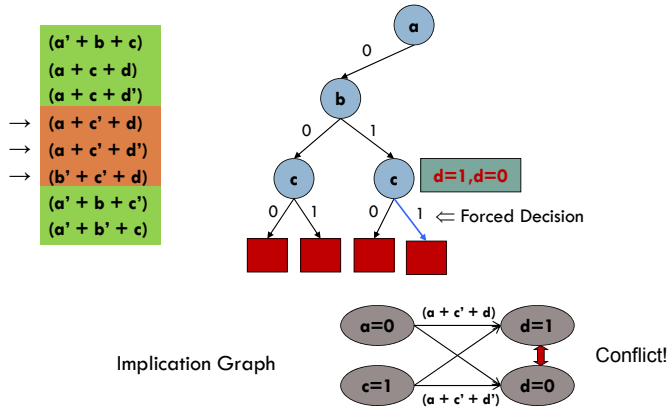


Conflict!

Basic DLL Search

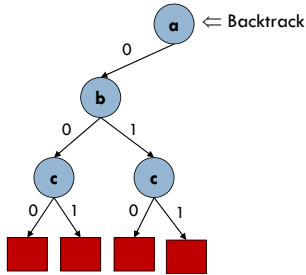


Basic DLL Search

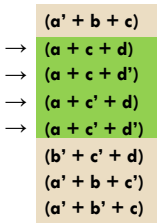


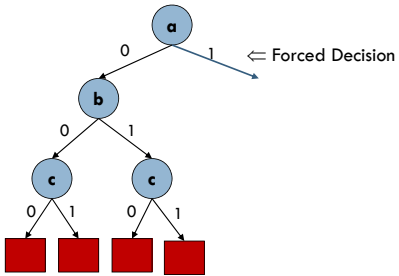
Basic DLL Search

→ (a' + b + c)
→ (a + c + d)
→ (a + c + d')
→ (a + c' + d)
→ (a + c' + d')
→ (b' + c' + d)
→ (a' + b + c')
→ (a' + b' + c)

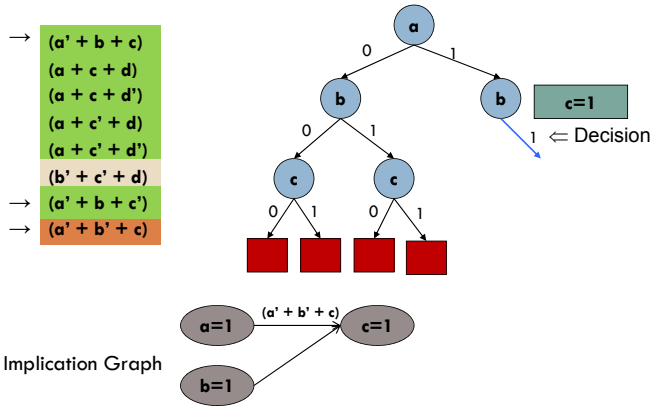


Basic DLL Search

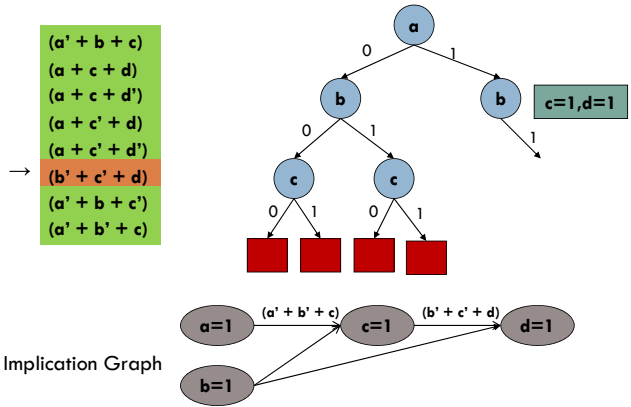




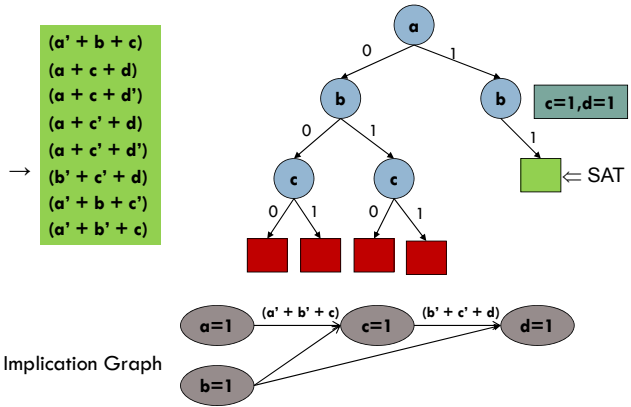
Basic DLL Search



Basic DLL Search



Basic DLL Search



SAT Solvers: A Condensed History

- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by “resolution”
- Backtrack Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Exhaustive search for satisfying assignment
- Conflict Driven Clause Learning [CDCL]
  - GRASP: Integrate a constraint learning procedure, 1996
- Locality Based Search
  - Emphasis on exhausting local sub-spaces, e.g. Chaff, Berkmin, miniSAT and others, 2001 onwards
  - Added focus on efficient implementation
- “Pre-processing”
  - Peephole optimization, e.g. miniSAT, 2005

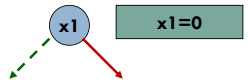
Conflict Driven Learning and Non-chronological Backtracking

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

J. P. Marques-Silva and Karem A. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability", *IEEE Trans. Computers*, C-48, 5:506-521, 1999.

Conflict Driven Learning and Non-chronological Backtracking

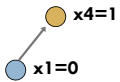
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

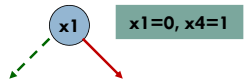


x1=0

Conflict Driven Learning and Non-chronological Backtracking

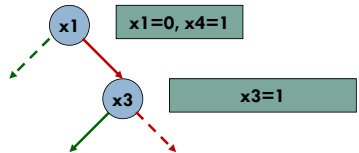
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'





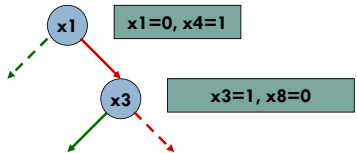
Conflict Driven Learning and Non-chronological Backtracking

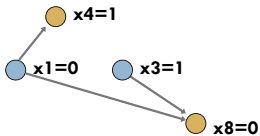
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'



Conflict Driven Learning and Non-chronological Backtracking

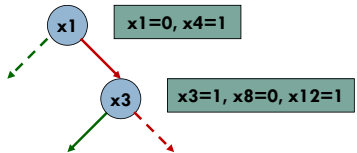
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

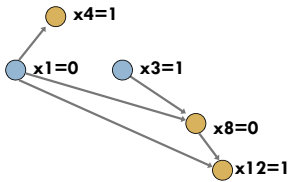




Conflict Driven Learning and Non-chronological Backtracking

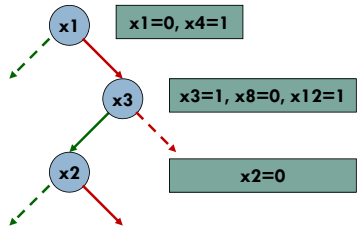
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

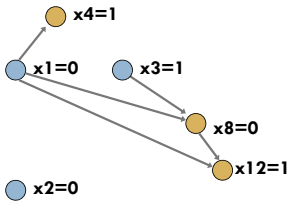




Conflict Driven Learning and Non-chronological Backtracking

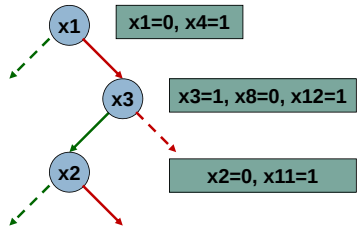
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

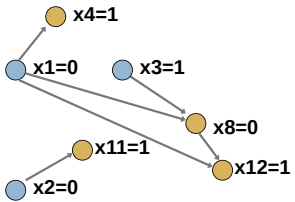




Conflict Driven Learning and Non-chronological Backtracking

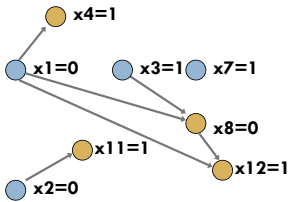
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

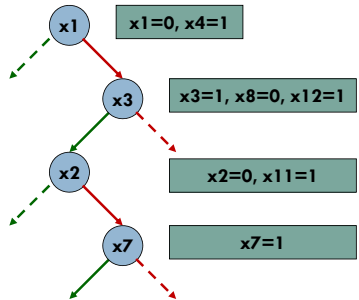




Conflict Driven Learning and Non-chronological Backtracking

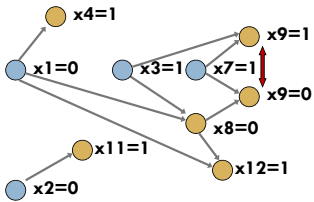
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

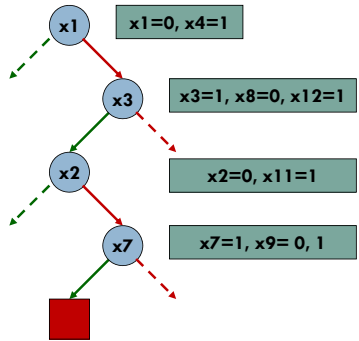




Conflict Driven Learning and Non-chronological Backtracking

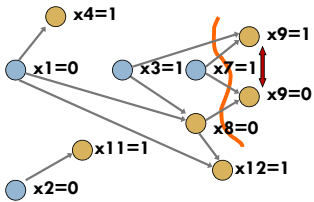
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

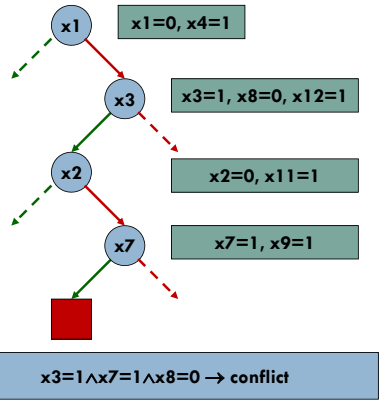




Conflict Driven Learning and Non-chronological Backtracking

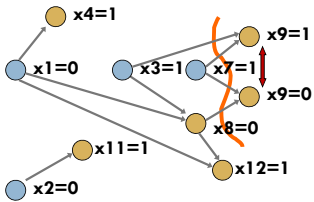
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

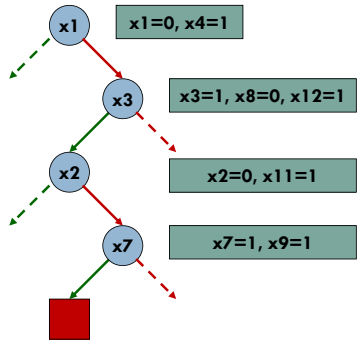




Conflict Driven Learning and Non-chronological Backtracking

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

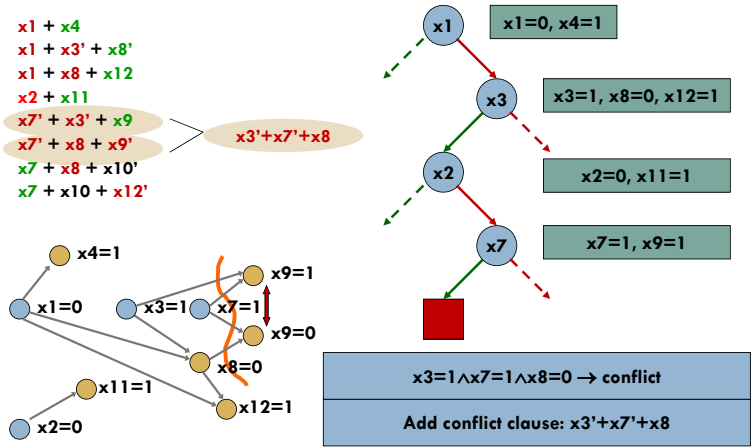




x3=1∧x7=1∧x8=0 → conflict

Add conflict clause: x3'+x7'+x8

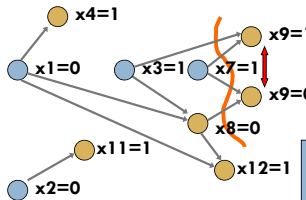
Conflict Driven Learning and Non-chronological Backtracking

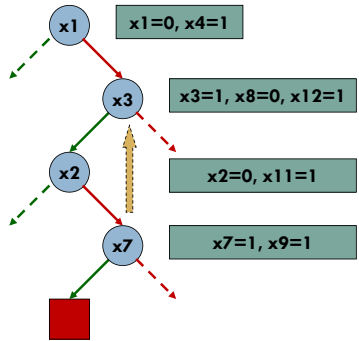


Conflict Driven Learning and Non-chronological Backtracking

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

x3'+x7'+x8

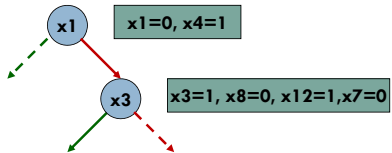


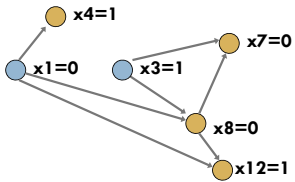


Backtrack to the decision level of x3=1

Conflict Driven Learning and Non-chronological Backtracking

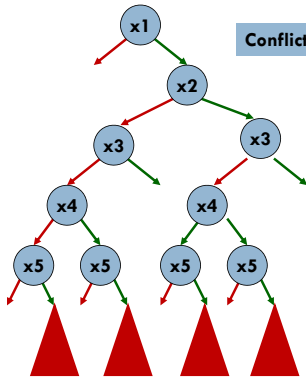
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'
x3' + x7' + x8 ←new clause





Backtrack to the decision level of x3=1
Assign x7 = 0

What's the big deal?



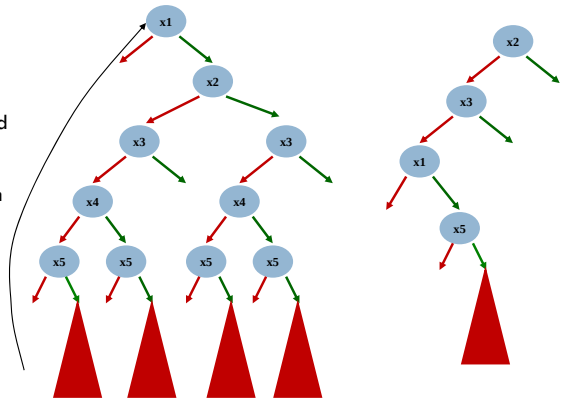
Conflict clause: x1'+x3+x5'

Significantly prune the search space – learned clause is useful forever!

Useful in generating future conflict clauses.

Restart

- Abandon the current search tree and reconstruct a new one
- The clauses learned prior to the restart are *still there* after the restart and can help pruning the search space
- Adds to robustness in the solver



SAT Solvers: A Condensed History

- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by "resolution"
- Backtrack Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Exhaustive search for satisfying assignment
- Conflict Driven Clause Learning [CDCL]
  - GRASP: Integrate a constraint learning procedure, 1996
- Locality Based Search
  - Emphasis on exhausting local sub-spaces, e.g. Chaff, Berkmin, miniSAT and others, 2001 onwards
  - Added focus on efficient implementation
- "Pre-processing"
  - Peephole optimization, e.g. miniSAT, 2005

Success with Chaff

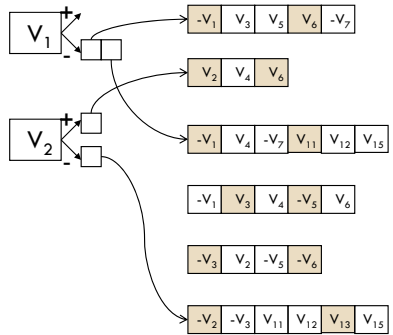
- First major instance: Tough (Industrial Processor Verification)
  - Bounded Model Checking, 14 cycle behavior
- Statistics
  - 1 million variables
  - 10 million literals initially
    - 200 million literals including added clauses
    - 30 million literals finally
  - 4 million clauses (initially)
    - 200K clauses added
  - 1.5 million decisions
  - 3 hour run time

M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc., 38th Design Automation Conference (DAC2001), June* 2001.

Chaff Contribution 1: Lazy Data Structures
2 Literal Watching for Unit-Propagation

- Avoid expensive book-keeping for unit-propagation
- N-literal clause can be unit or conflicting only after N-1 of the literals have been assigned to F
  - (v1 + v2 + v3): implied cases: (0 + 0 + v3) or (0 + v2 + 0) or (v1 + 0 + 0)
- Can completely ignore the first N-2 assignments to this clause
- Pick two literals in each clause to "watch" and thus can ignore any assignments to the other literals in the clause.
  - Example: (v1 + v2 + v3 + v4 + v5)
  - (**v1=X + v2=X** + v3=? {i.e. X or 0 or 1} + v4=? + v5=?)
- *Maintain the invariant*: If a clause can become newly implied via any sequence of assignments, then this sequence will include an assignment of one of the watched literals to F

2 Literal Watching



For every clause, two literals are watched

- When a variable is assigned true, only need to visit clauses where its watched literal is false (only one polarity)
  - Pointers from each literal to all clauses it is watched in
- In a *n* clause formula with *v* variables and *m* literals
  - Total number of pointers is *2n*
  - On average, visit *n/v* clauses per assignment
- *No updates to watched literals on backtrack*

Decision Heuristics – Conventional Wisdom

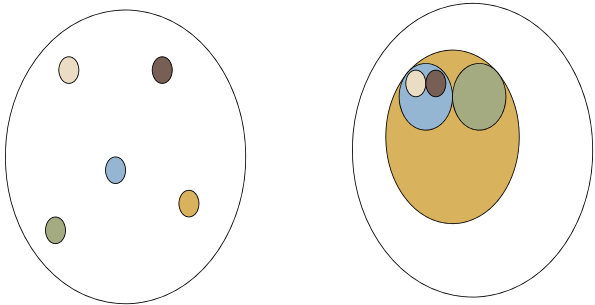
- "Assign most tightly constrained variable" : e.g. DLIS (Dynamic Largest Individual Sum)
  - Simple and intuitive: At each decision simply choose the assignment that satisfies the most unsatisfied clauses.
  - Expensive book-keeping operations required
    - Must touch *every* clause that contains a literal that has been set to true. Often restricted to initial (not learned) clauses.
    - Need to reverse the process for un-assignment.
- Look ahead algorithms even more compute intensive

  C. Li, Anbulagan, "Look-ahead versus look-back for satisfiability problems" *Proc. of CP*, 1997.
- Take a more "global" view of the problem

Chaff Contribution 2: Activity Based Decision Heuristics

- VSIDS: **V**ariable **S**tate **I**ndependent **D**ecaying **S**um
  - Rank variables by literal count in the initial clause database
  - Only increment counts as new (learnt) clauses are added
  - Periodically, divide all counts by a constant
- Quasi-static:
  - Static because it doesn't depend on variable state
  - Not static because it gradually changes as new clauses are added
    - Decay causes bias toward *recent* conflicts.
    - Has a beneficial interaction with 2-literal watching

Activity Based Heuristics and Locality Based Search



- By focusing on a sub-space, the covered spaces tend to coalesce
  - More opportunities for resolution since most of the variables are common.
  - Variable activity based heuristics lead to locality based search

SAT Solvers: A Condensed History

- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by “resolution”
- Backtrack Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Exhaustive search for satisfying assignment
- Conflict Driven Clause Learning [CDCL]
  - GRASP: Integrate a constraint learning procedure, 1996
- Locality Based Search
  - Emphasis on exhausting local sub-spaces, e.g. Chaff, Berkmin, miniSAT and others, 2001 onwards
  - Added focus on efficient implementation
- “Pre-processing”
  - Peephole optimization, e.g. miniSAT, 2005

Pre-Processing of CNF Formulas

N. Eén and A. Biere. Effective Preprocessing in SAT through Variable and Clause Elimination, In *Proceedings of SAT 2005*

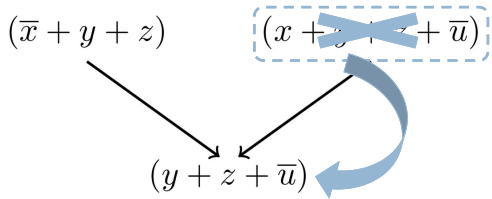
- Use structural information to simplify
  - Subsumption
  - Self-subsumption
  - Substitution

Pre-Processing: Subsumption

- Clause C_1 *subsumes* clause C_2 if C_1 *implies* C_2
- Subsumed clauses can be discarded

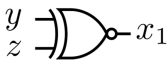
Pre-Processing: Self-Subsumption

- Subsumption *after* resolution step



Pre-Processing: Substitution

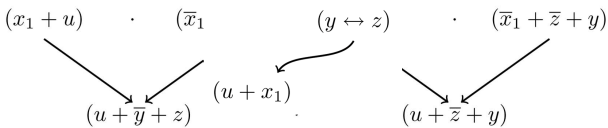
- Tseitin transformation introduces *definition* of variable



$$\underbrace{(x_1 \leftrightarrow (y \leftrightarrow z))}$$

$$(\overline{x}_1+\overline{y}+z)\cdot(\overline{x}_1+\overline{z}+y)\cdot(\overline{y}+\overline{z}+x_1)\cdot(y+z+x_1)$$

- Occurrence of x_1 can be eliminated by substitution
  - Corresponds to resolution with defining clauses



Concluding Remarks

- SAT: Significant shift from theoretical interest to practical impact.
- Quantum leaps between generations of SAT solvers
- Successful application of diverse CS techniques
  - Logic (Deduction and Solving), Search, Caching, Randomization, Data structures, efficient algorithms
  - Engineering developments through experimental computer science
- Presence of drivers results in maximum progress.
  - Electronic design automation – primary driver and main beneficiary
  - Software verification- the next frontier
- Opens attack on even harder problems
  - SMT, Max-SAT, QBF…

Sharad Malik and Lintao Zhang. 2009. Boolean satisfiability from theoretical hardness to practical success. *Commun. ACM* 52, 8 (August 2009), 76-82.

References

- [GJ79] Michael R. Garey and David S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman and Company, San Francisco, 1979
- [T68] G. Tseitin, On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic*, Part 2 (1968)
- [DP 60] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM, 7:201–215, 1960*
- [DLL62] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM, 5:394–397, 1962*
- [SS99] J. P. Marques-Silva and Karem A. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability", *IEEE Trans. Computers*, C-48, 5:506-521, 1999.
- [BS97] R. J. Bayardo Jr. and R. C. Schrag "Using CSP look-back techniques to solve real world SAT instances." *Proc. AAAI*, pp. 203-208, 1997
- [BS00] Luís Baptista and João Marques-Silva, "Using Randomization and Learning to Solve Hard Real-World Instances of Satisfiability," In *Principles and Practice of Constraint Programming – CP 2000,* 2000.

References

- [H07] J. Huang, “The effect of restarts on the efficiency of clause learning,” Proceedings of the Twentieth International Joint Conference on Automated Reasoning, 2007
- [MMZ+01] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik. Chaff: Engineering and efficient sat solver. In *Proc., 38th Design Automation Conference (DAC2001), June* 2001.
- [ZS96] H. Zhang, M. Stickel, “An efficient algorithm for unit-propagation” In Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics,1996
- [ES03] N. Een and N. Sorensson. An extensible SAT solver. In SAT-2003
- [B02] F. Bacchus “Exploring the Computational Tradeoff of more Reasoning and Less Searching”, *Proc. 5th Int. Symp. Theory and Applications of Satisfiability Testing*, pp. 7-16, 2002.
- [GN02] E.Goldberg and Y.Novikov. BerkMin: a fast and robust SAT-solver. In *Proc., DATE-2002, pages 142–149, 2002.*

References

- [R04] L. Ryan, Efficient algorithms for clause-learning SAT solvers, M. Sc. Thesis, Simon Fraser University, 2002.
- [EB05] N. Eén and A. Biere. Effective Preprocessing in SAT through Variable and Clause Elimination, In *Proceedings of SAT 2005*
- [ZM03] L. Zhang and S. Malik, Validating SAT solvers using an independent resolution-based checker: practical implementations and other applications, In Proceedings of Design Automation and Test in Europe, 2003.
- [LSB07] M. Lewis, T. Schubert, B. Becker, Multithreaded SAT Solving, In Proceedings of the 2007 Conference on Asia South Pacific Design Automation
- [HJS08] Youssef Hamadi, Said Jabbour, and Lakhdar Sais, ManySat: solver description, Microsoft Research-TR-2008-83
- [B86] R. E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Transactions on Computers* , vol.C-35, no.8, pp.677-691, Aug. 1986
- [ZM09] Sharad Malik and Lintao Zhang. 2009. Boolean satisfiability from theoretical hardness to practical success. *Commun. ACM* 52, 8 (August 2009), 76-82.