Sharad Malik

From: Rajeev Alur [alur@seas.upenn.edu]
Sent: Friday, October 31, 2008 4:07 PM
To: Clark Barrett; Edmund Clarke; Sharad Malik; pvh@cs.brown.edu; selman@cs.cornell.edu; Jose Meseguer; Amir Pnueli; Leonardo de Moura; Alex Aiken; Thomas Reps; Natarajan Shankar
Subject: NSF Workshop on Symbolic Computation for Constraint Satisfaction Problems

Hi all,
Hope you have already made your travel arrangements.
We need to finalize the program and send it to NSF in a few days, so please respond to this message with the title of your talk.

I am proposing a tentative schedule and topics.

Friday, Nov 14,
Location: FDIC Building, Arlington, VA (a few blocks from NSF bldg)

Tentative schedule:

8.30 -- 9.00 Coffee / Breakfast
9.00 -- 9.15 Welcome and agenda

9.15 -- 11.15 Computational Tools for Constraint Satisfaction Problems

[Talks: 25 min + 5 min for questions]

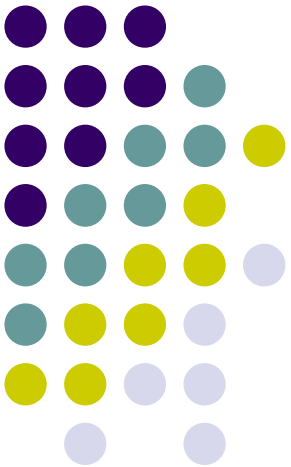
Sharad Malik: SAT/QBF Solvers
Clark Barrett: SMT Solvers
Jose Meseguer: Rewriting tools
Bart Selman: Solvers in Planning

11.15 -- 12.30 Lunch

SAT and QBF: Trick or Treat

Sharad Malik

Princeton University



NSF Workshop

November 14, 2008

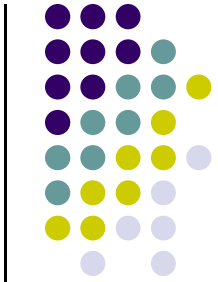
The Trick…

The daunting NP-completeness…

"I can't find an efficient algorithm, but neither can all these famous people."

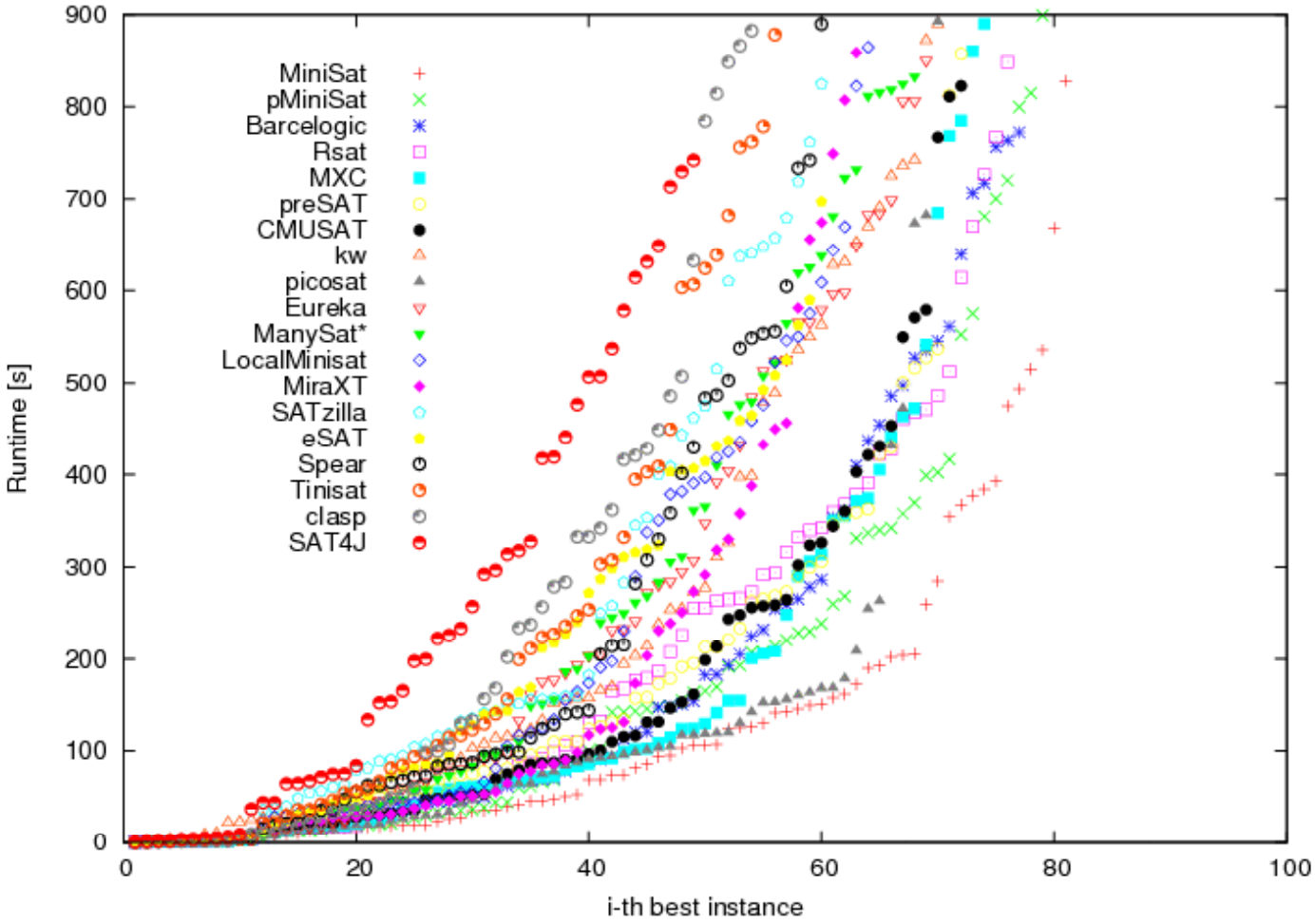
[GJ79]

The Treat…



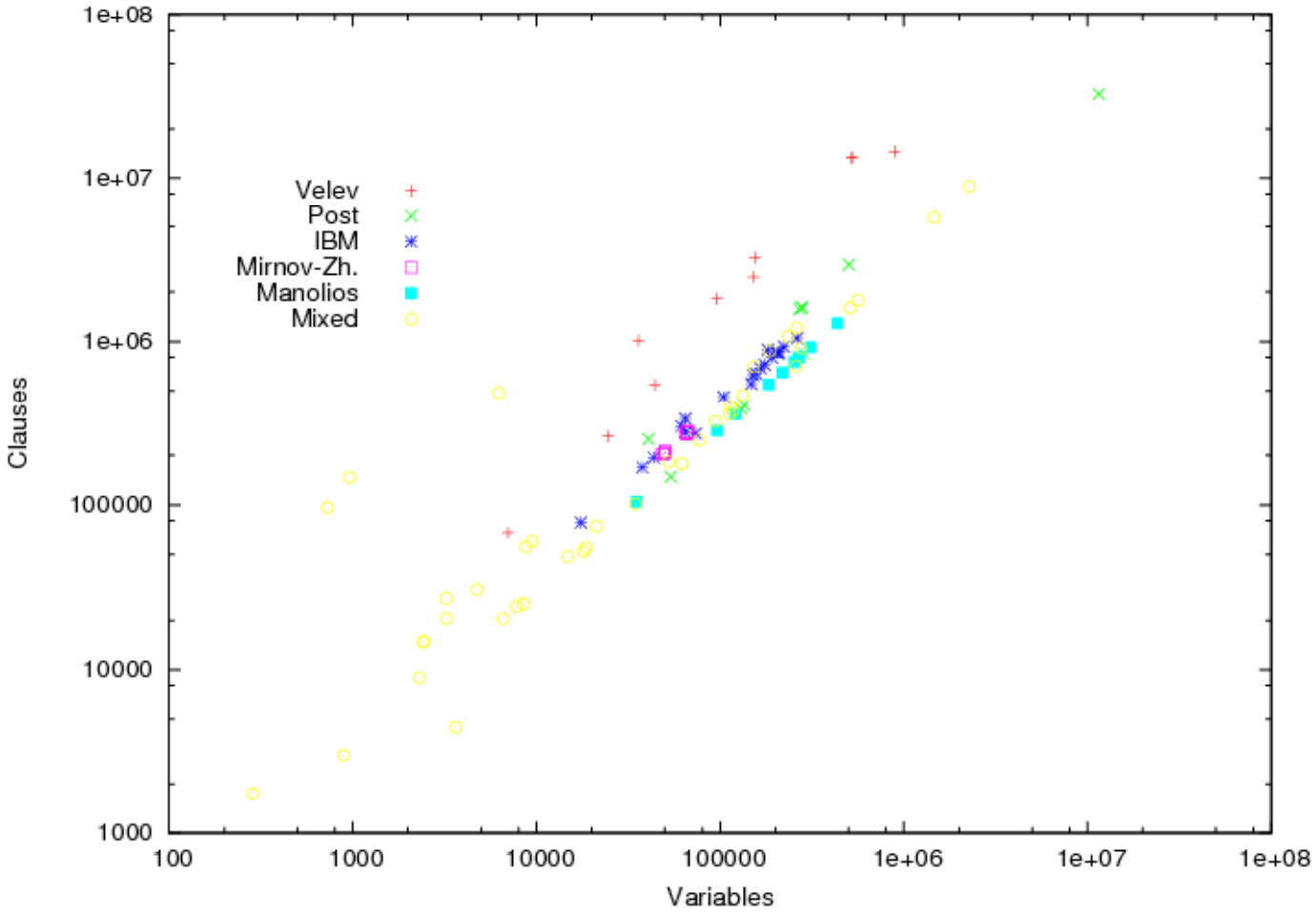
- Large and vibrant SAT community
  - SAT Portal www.satlive.org
  - Satlib Research Infrastructure
    - 60K benchmarks
    - SAT solver competitions
    - Public domain solvers
- Wide practical application of SAT
  - More from de Moura, McMillan and others later…
- Emboldened researchers to attack harder problems
  - QBF and SMT

SAT Solver Competition



http://www-sr.informatik.uni-tuebingen.de/sat-race-2008/analysis.html

SAT Solver Competition



http://www-sr.informatik.uni-tuebingen.de/sat-race-2008/analysis.html

This Talk

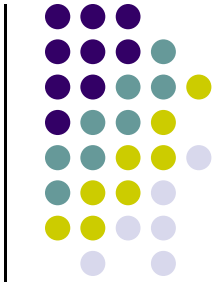
- Successful application of diverse CS techniques
  - Logic (Deduction and Solving)
  - Search
  - Caching
  - Randomization
  - Data structures
  - Cache efficient algorithms
- Open challenges…
  - Limited understanding of why the algorithms work
  - Dynamic application of strategies
  - QBF

Acknowledgements

For discussions:

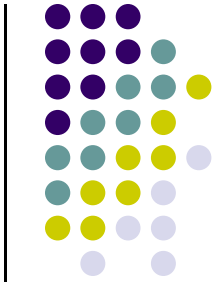
- Armin Bierre
- Daniel Le Berre
- Laurent Simon
- Ofer Strichman
- Lintao Zhang

SAT Solvers: A Condensed History

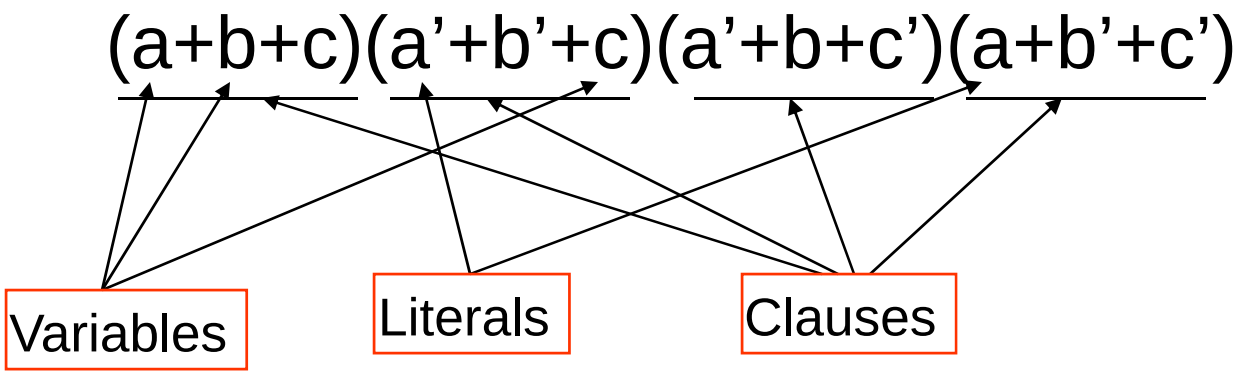


- Deductive/Formula Solving
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by resolution
- Backtrack Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Search with unit propagation
- Conflict Driven Clause Learning [CDCL]
  - GRASP, RelSat: Integrate a constraint learning procedure, 1996
- Locality Based Search
  - Emphasis on exhausting local sub-spaces, e.g. Chaff, Berkmin, miniSAT and others, 2000 onwards
  - Added focus on efficient implementation
    - Boolean Constraint Propagation, Decision Heuristics, …

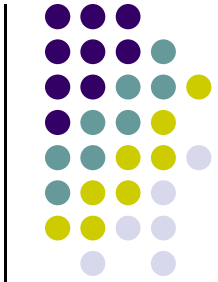
Problem Representation



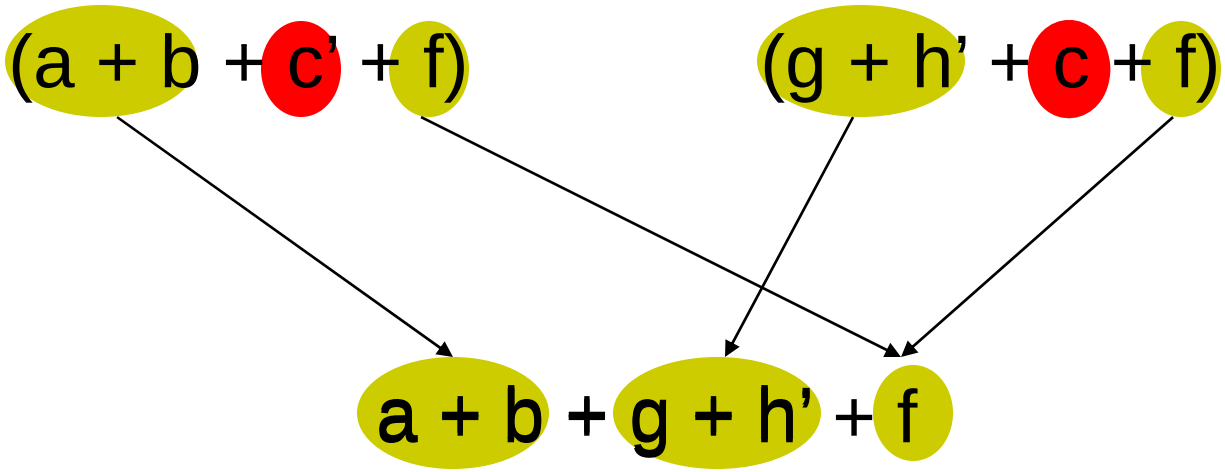
- Conjunctive Normal Form
  - Representation of choice for modern SAT solvers
  - Easy conversion of other representations to CNF
    - E.g. Circuit to CNF using the Tseitin Transformation



Deduction Workhorse I: Resolution

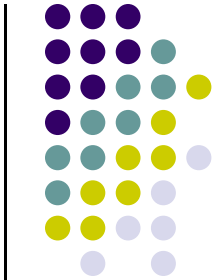


- Resolution of a pair of distance-one clauses

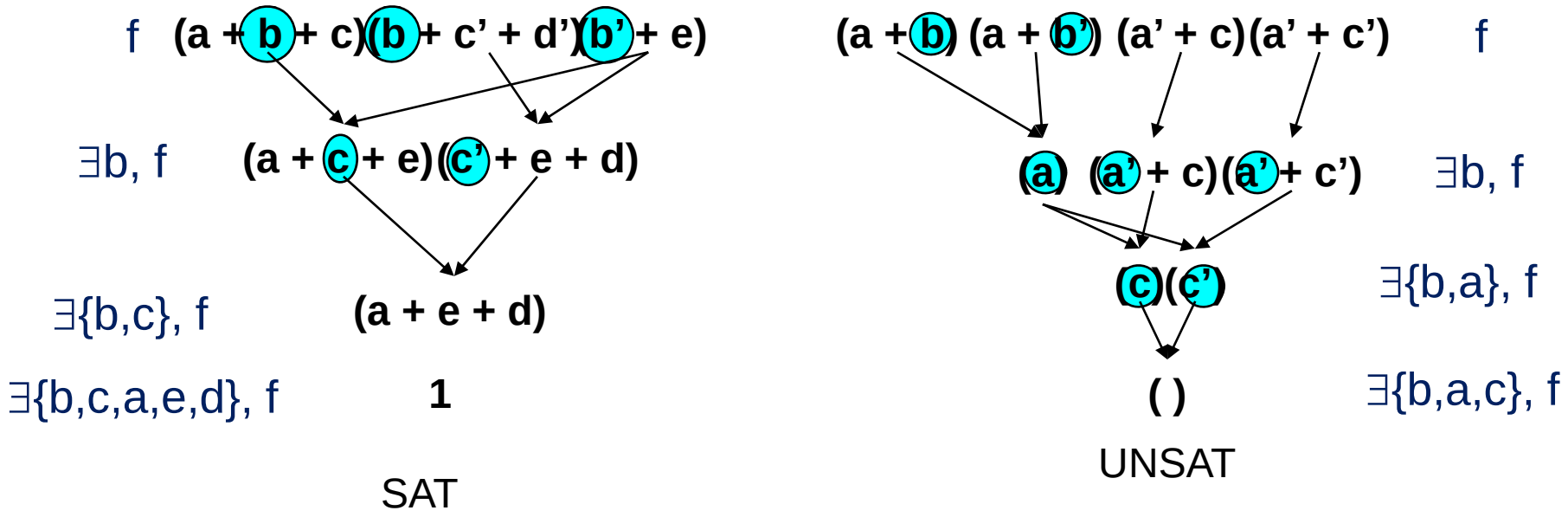


Resolvent implied by the original clauses, thus can be added back to the CNF without changing the formula

Davis Putnam Algorithm

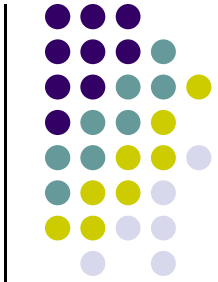


- Iterative existential quantification of variables [DP 60]
  - Using resolution



Potential memory explosion problem!

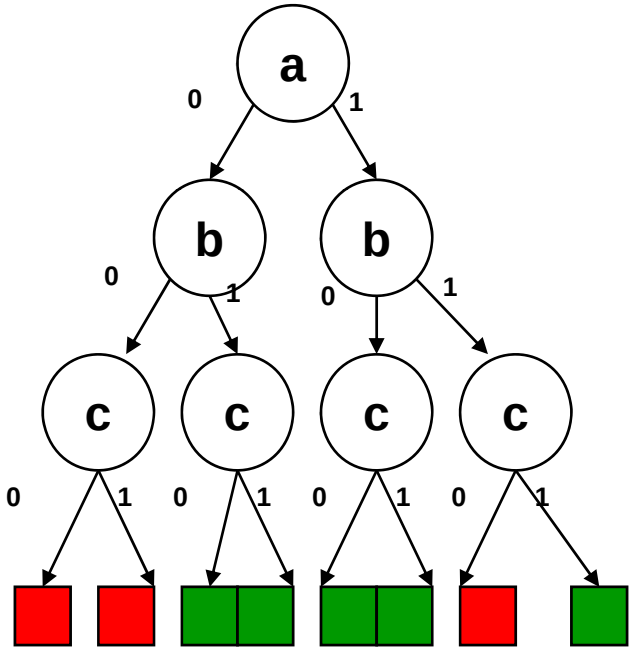
SAT Solvers: A Condensed History



- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by resolution
- Backtrack Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Search with unit propagation
- Conflict Driven Clause Learning [CDCL]
  - GRASP, RelSat: 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
    - Boolean Constraint Propagation, Decision Heuristics, …

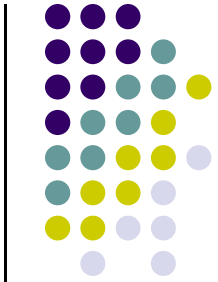
DLL Search

- Search the decision tree for a satisfying assignment
- Unit propagation to prune search
- Deduction Workhorse II: Unit literal rule
  - All but one literal in a clause is assigned false
    - (v1=0 + v2=0 + v3=?)
    - v3 must be 1 for the formula to be satisfiable
- Unit propagation is the iterative application of this rule



[DLL62]

SAT Solvers: A Condensed History

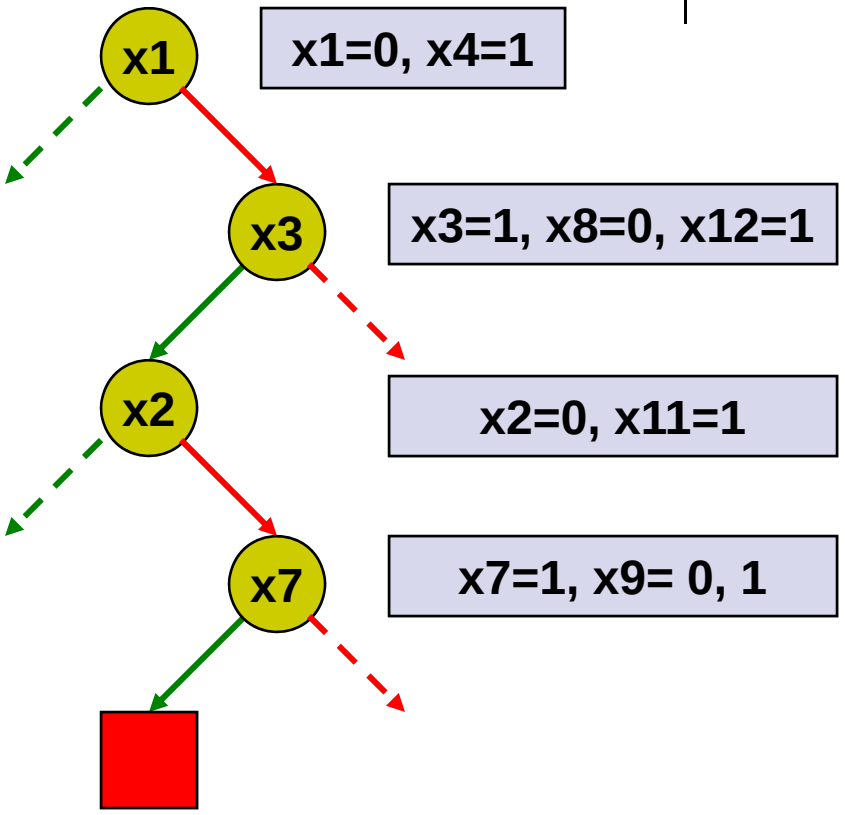


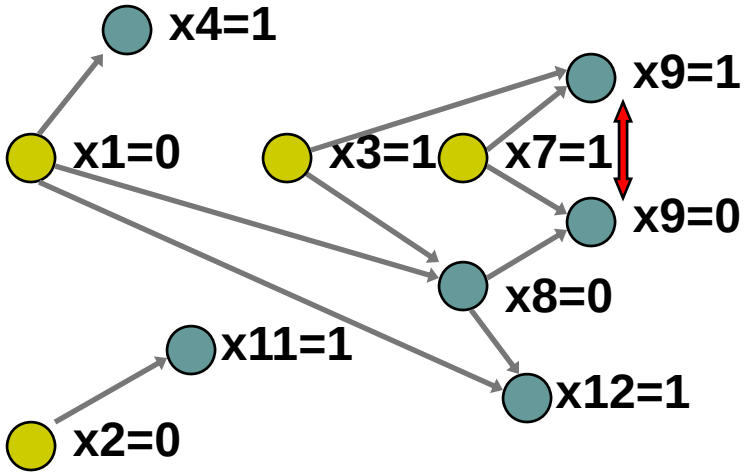
- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by resolution
- Backtracking Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Search with unit propagation
- Conflict Driven Clause Learning [CDCL]
  - GRASP, RelSat: Integrate a constraint learning procedure, 1996
- Locality Based Search
  - Emphasis on exhausting local sub-spaces, e.g. Chaff, Berkmin, miniSAT and others, 2001 onwards
  - Added focus on efficient implementation
    - Boolean Constraint Propagation, Decision Heuristics, …

Conflict Driven Learning

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

[SS99,BS96]

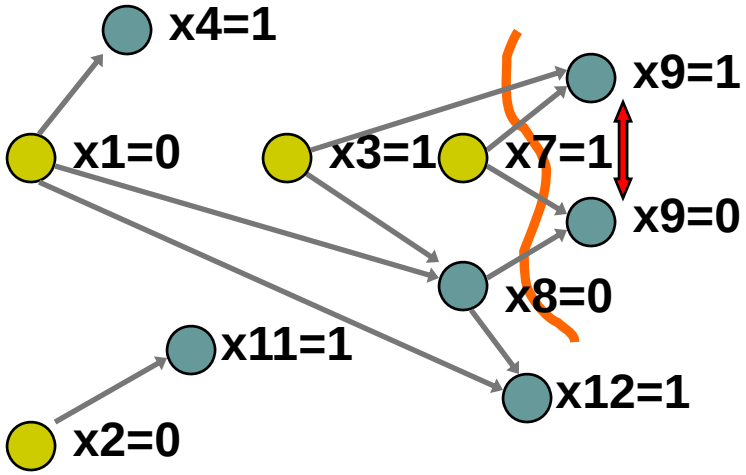


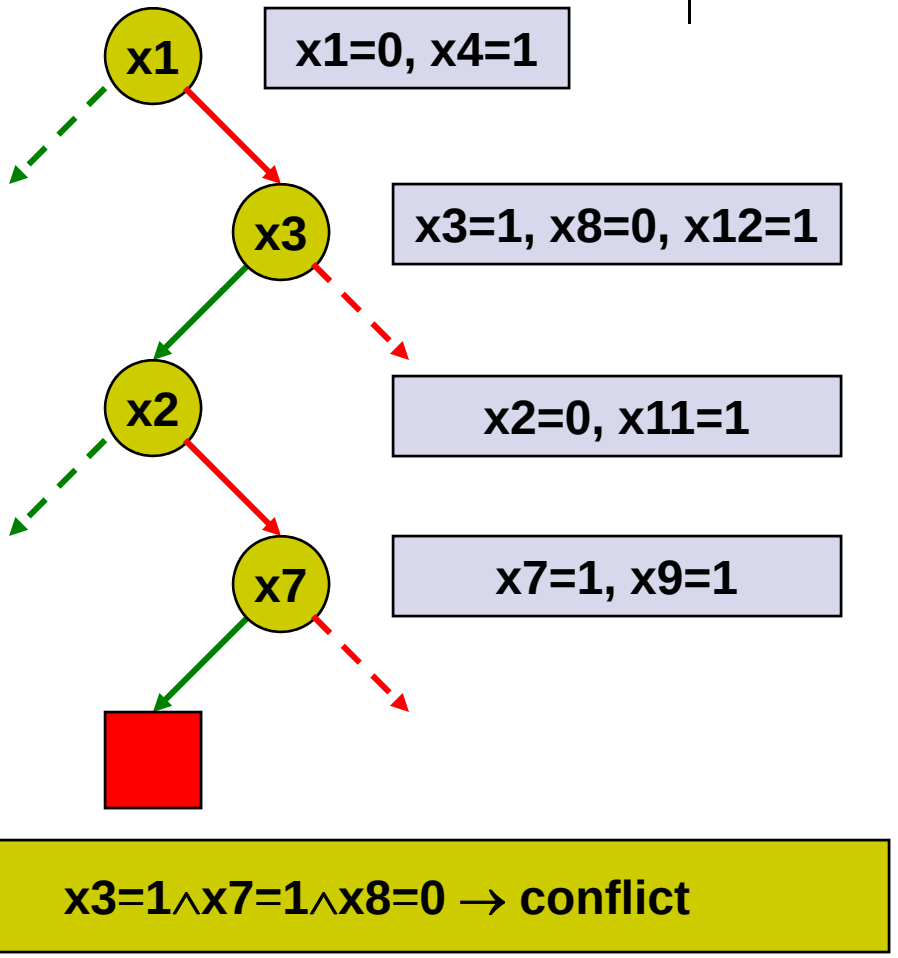


Implication graph:
record history of unit implications

Conflict Driven Learning

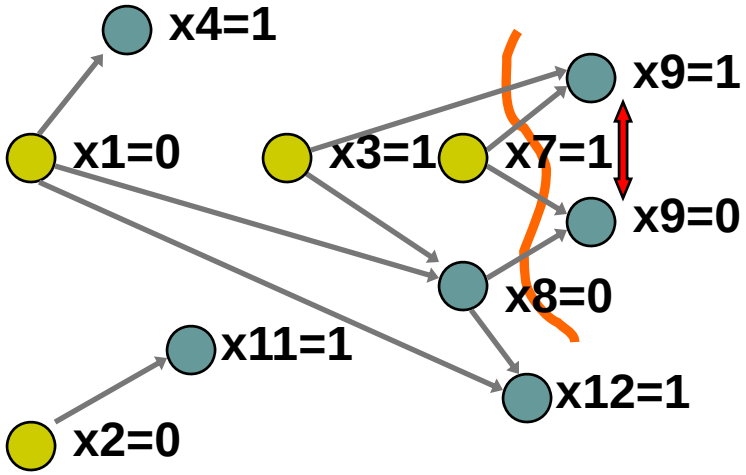
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

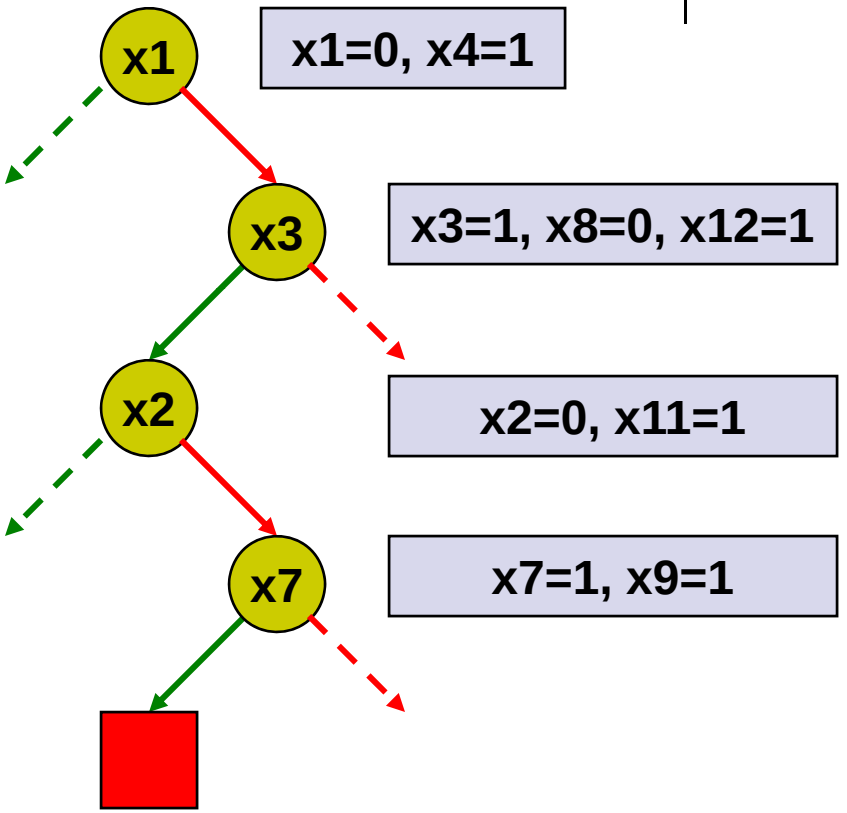




Conflict Driven Learning

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





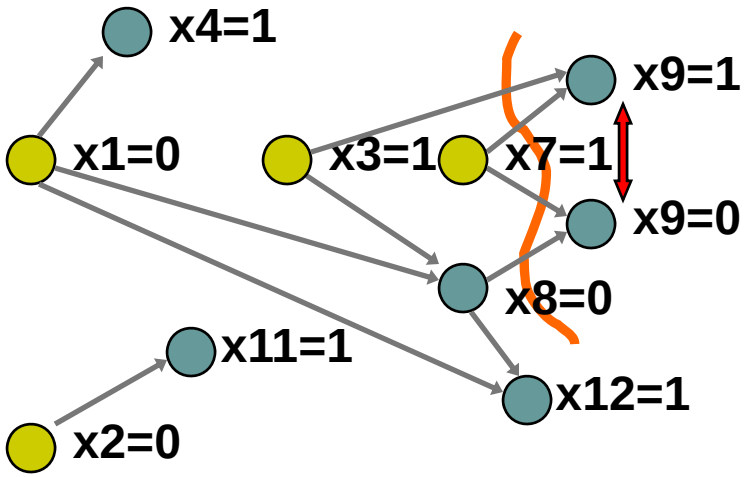
x3=1∧x7=1∧x8=0 → conflict

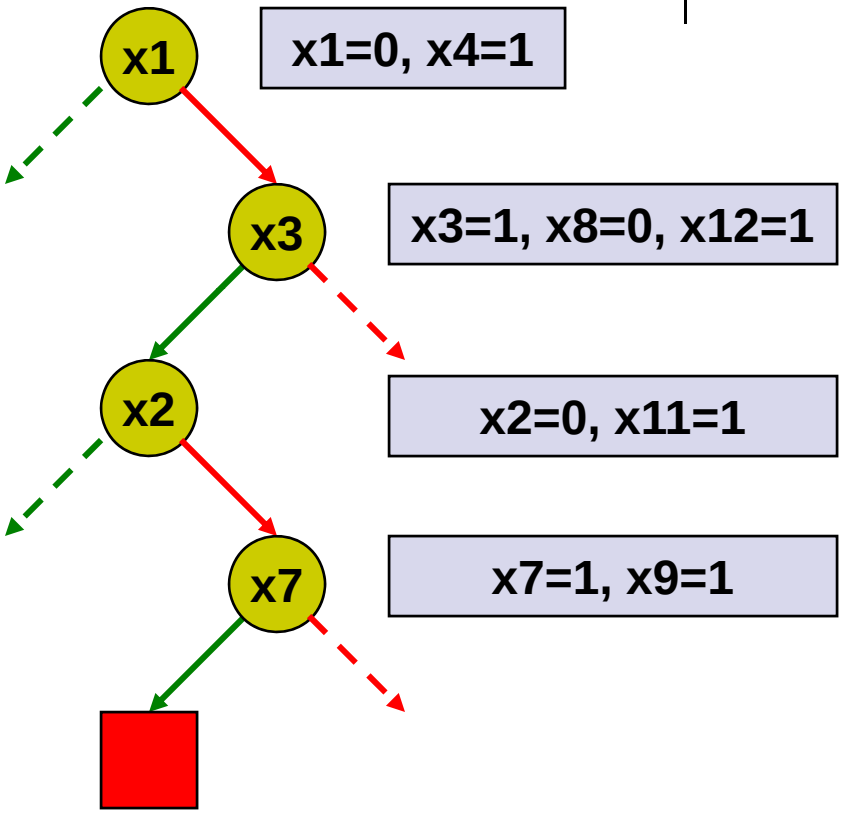
Add conflict clause: x3'+x7'+x8

Conflict Driven Learning

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





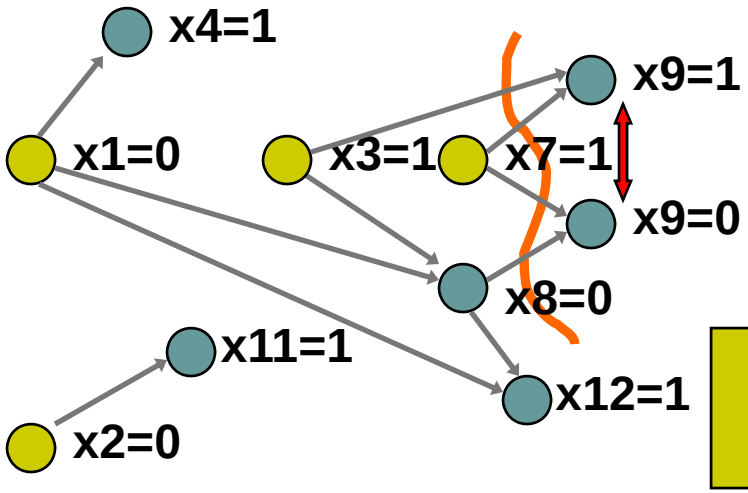
x3=1∧x7=1∧x8=0 → conflict

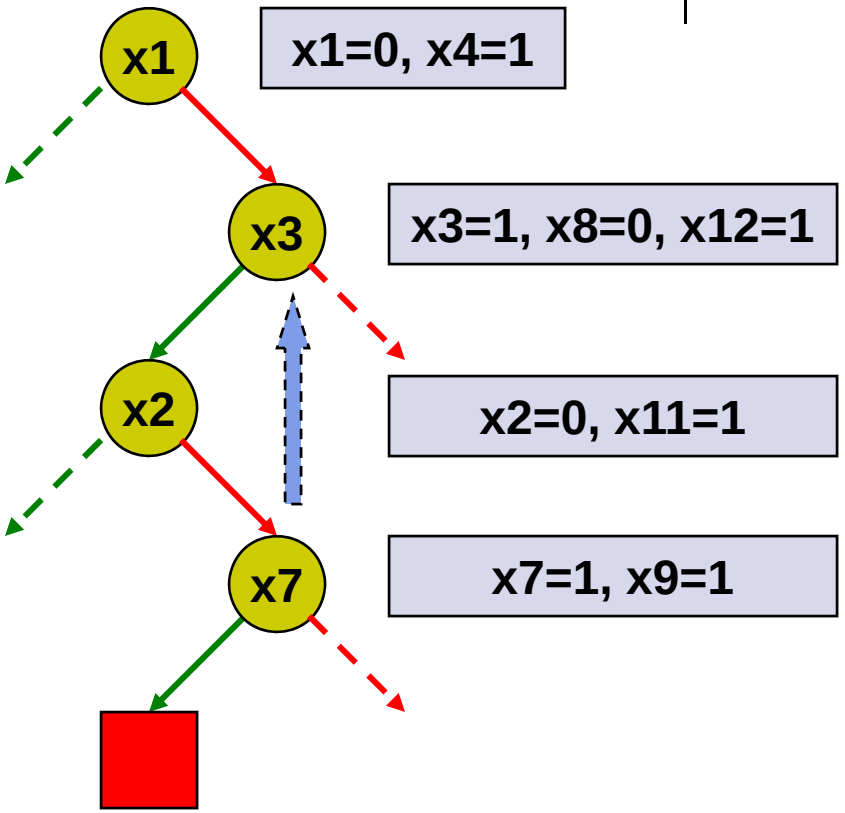
Add conflict clause: x3'+x7'+x8

Conflict Driven Learning

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

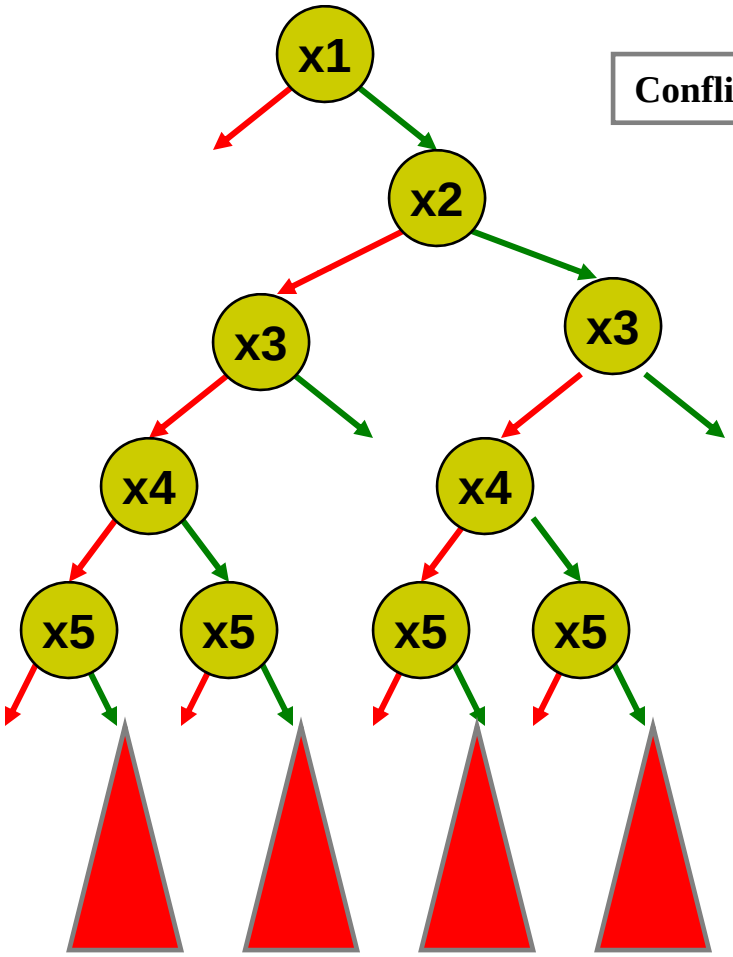
x3'+x7'+x8





Backtrack to the decision level of x3=1

What's the big deal?



Conflict clause: x1'+x3+x5'

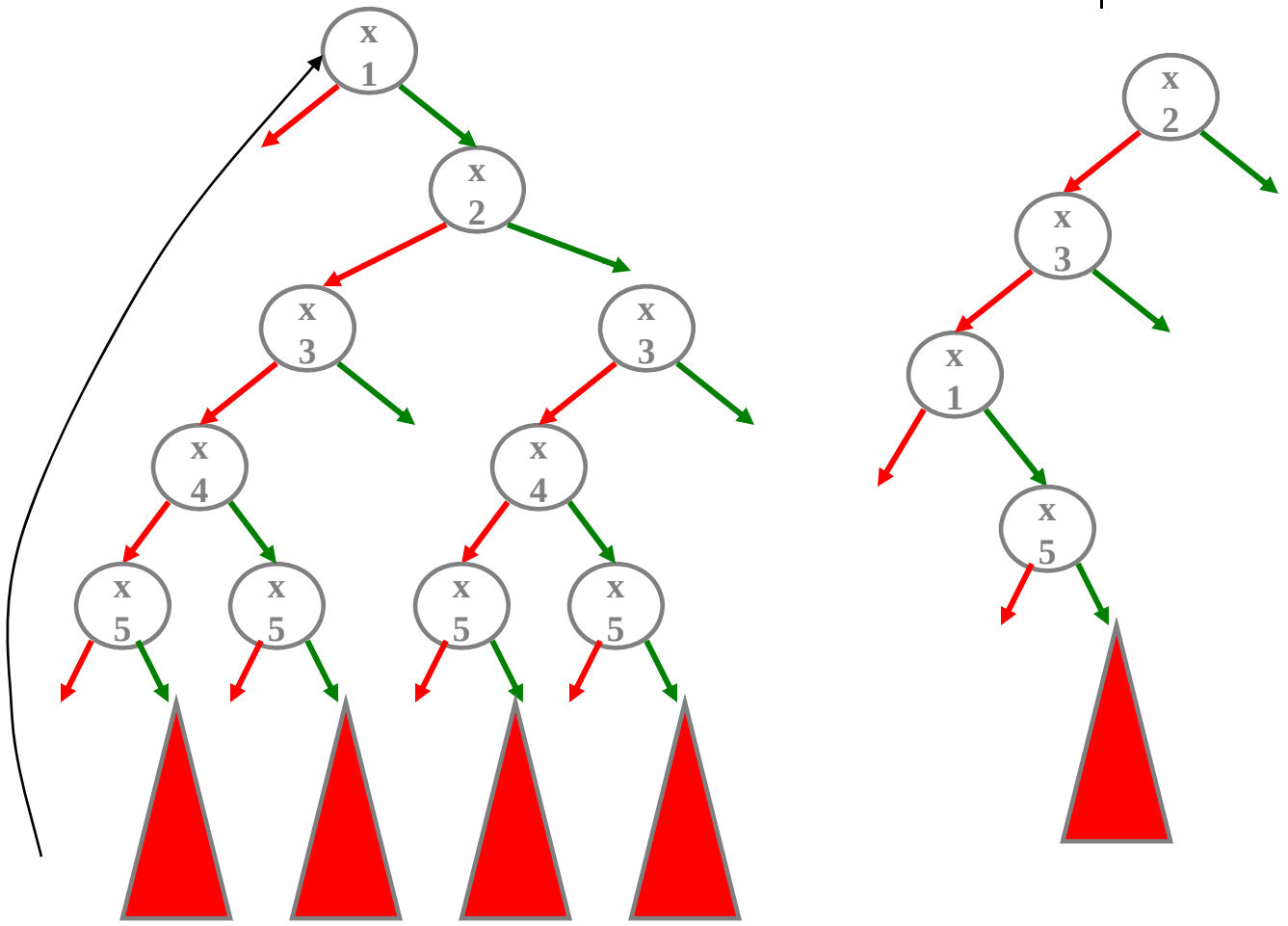
Significantly prune the search space – learned clause is useful forever!

Useful in generating future conflict clauses.

Very effective deduction/caching for search space pruning.

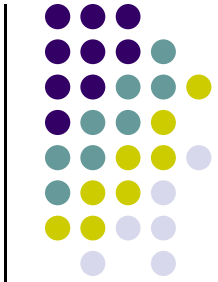
Restarts

- 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
- Effective randomization



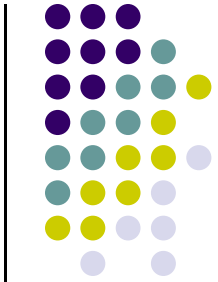
[BS00,H07]

SAT Solvers: A Condensed History



- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by resolution
- Backtracking Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Search with unit propagation
- Conflict Driven Clause Learning [CDCL]
  - GRASP, RelSat: 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
    - Boolean Constraint Propagation, Decision Heuristics, …

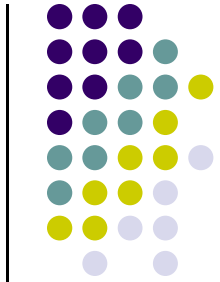
Success with Chaff (2000)



- 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

[MMZ+01]

Constants Matter

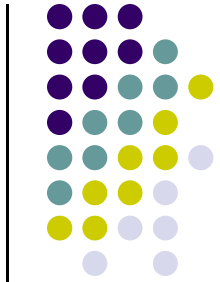


Motivating Metrics: Decisions, Instructions, Cache Performance and Run Time

	1dlx_c_mc_ex_bp_f
Num Variables	**776**
Num Clauses	**3725**
Num Literals	**10045**

	zChaff	SATO	GRASP
# Decisions	3166	3771	1795
# Instructions	86.6M	630.4M	1415.9M
# L1/L2 accesses	24M / 1.7M	188M / 79M	416M / 153M
% L1/L2 misses	4.8% / 4.6%	36.8% / 9.7%	32.9% / 50.3%
# Seconds	0.22	4.41	11.78

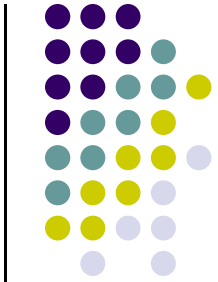
Unit Propagation Dominates



>80% of execution time!

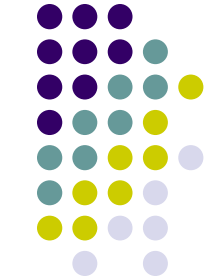
```
while(1) {
   is_conflicting = propagate_unit();

   if(!is_conflicting) {
       if (no_free_vars) return SATISFIABLE;
       make_decision(); // Decision Heuristic
   }
   if(is_conflicting) {
       if (no_unforced_decisions) return UNSAT;
       new_constraint = analyze_conflict(); // Learning
       literal = last_assigned(new_constraint);
       backtrack_to(asserting_level(new_constraint));
       assign(invert(literal)); // Conflict Driven Assertion
   }
}
```

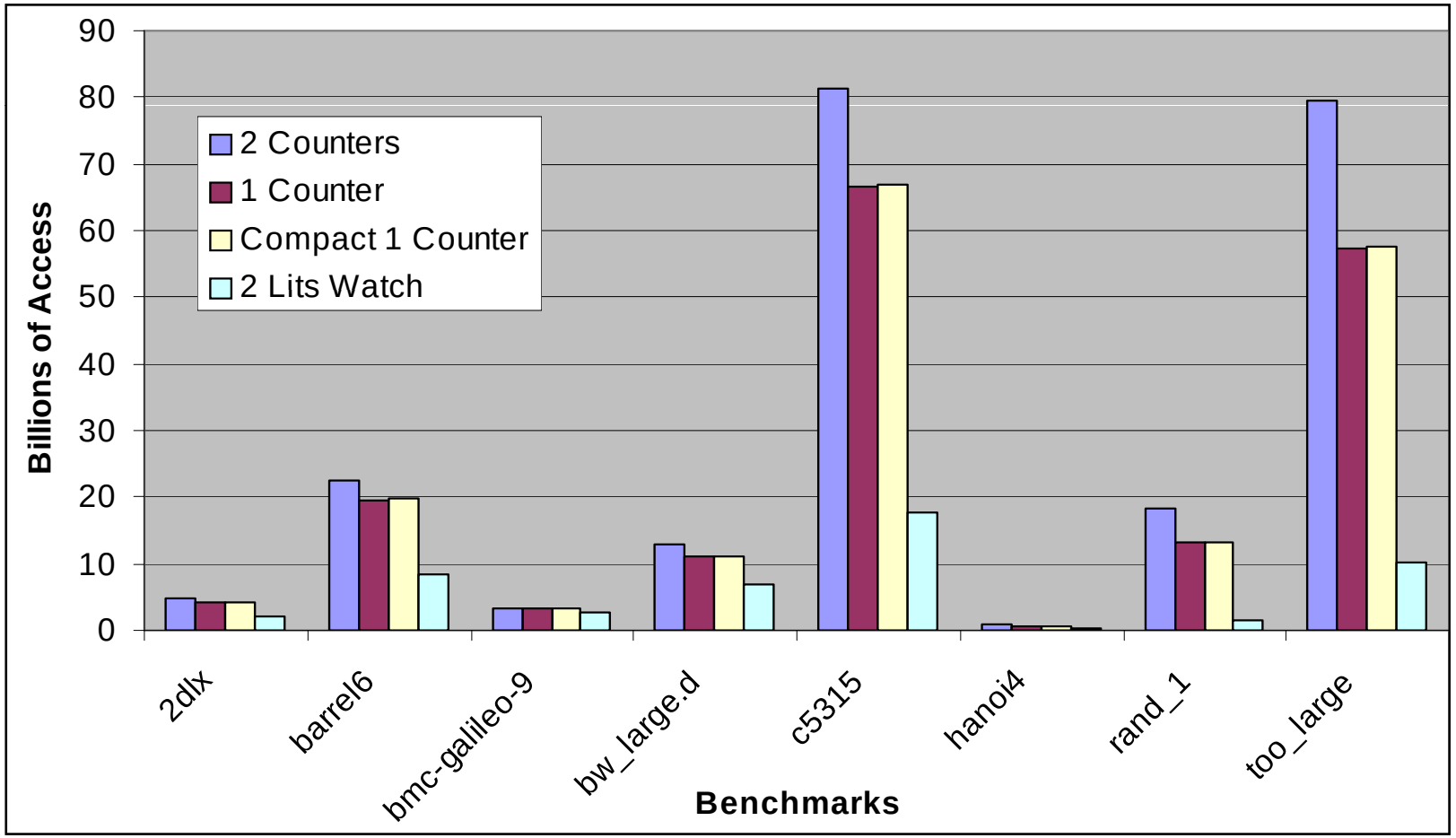
Tracking fewer literals per clause?



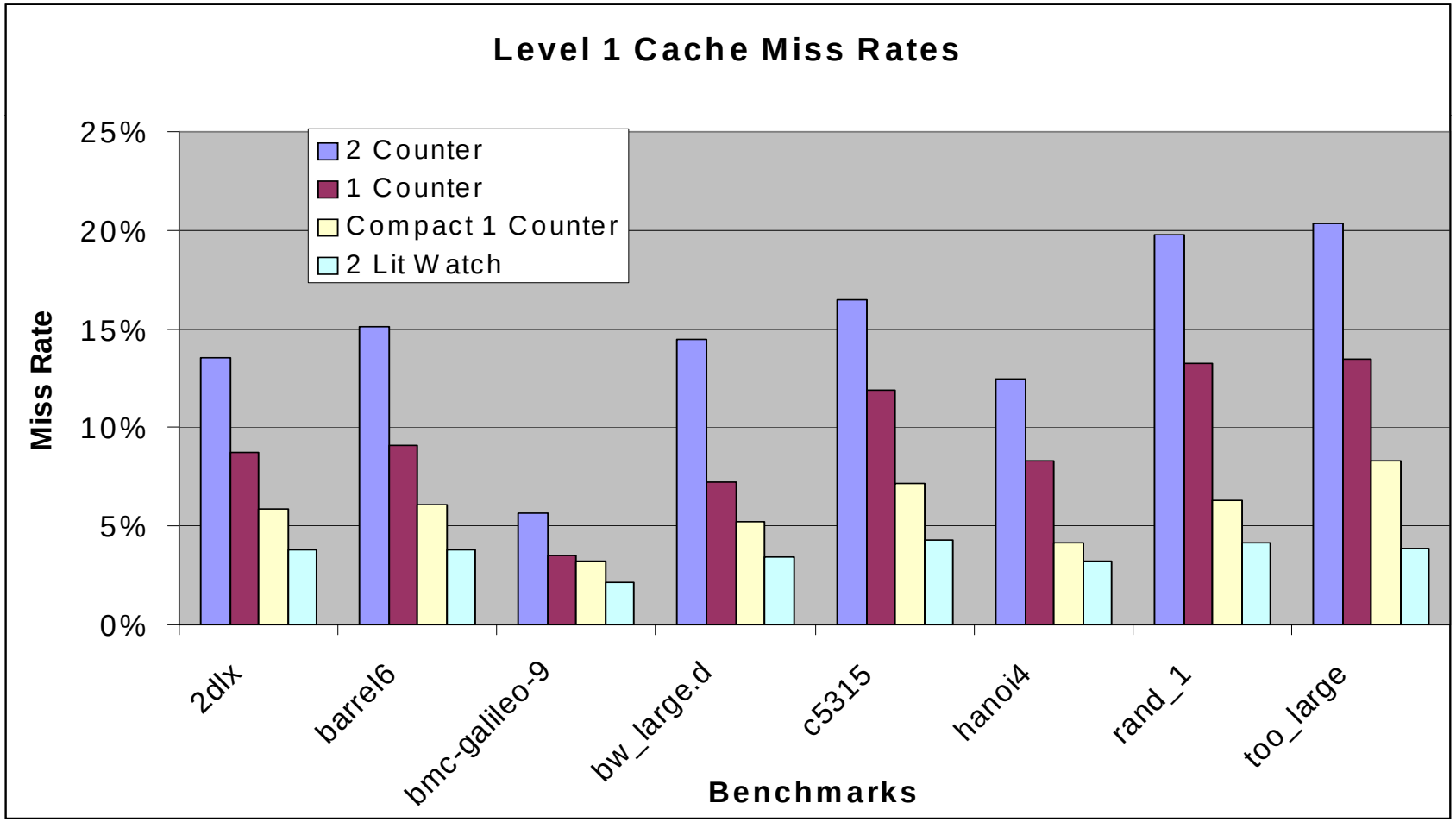
- A clause with 2 non-false literals can neither be unit nor conflicting
  - SATO's Head/Tail lists are based on this idea [HS96]
  - Chaff's 2 literal watching develops it further [MMZ+01]
    - Has significant implications on the algorithms
    - No updates needed on backtracking
  - Efficient Data Structures/Algorithms matter

Memory Accesses: Different BCP Mechanisms

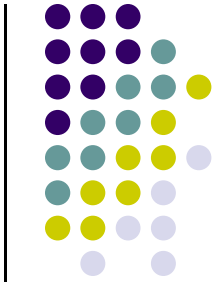




Level 1 Data Cache Miss Rates: Different BCP Methods

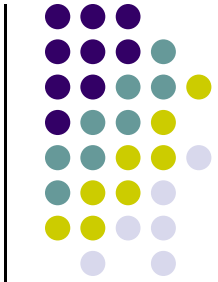


SAT Solvers: A Condensed History



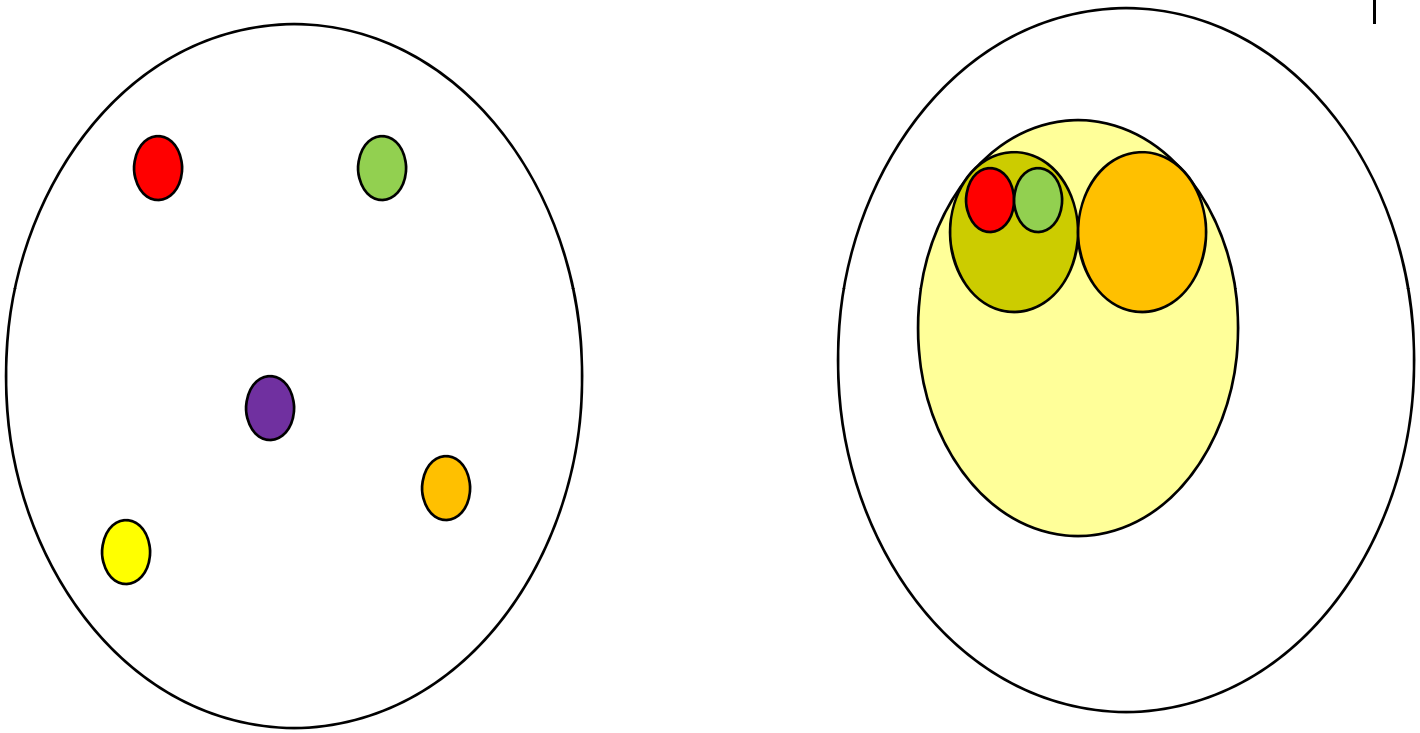
- Deductive
  - Davis-Putnam 1960 [DP]
  - Iterative existential quantification by resolution
- Backtracking Search
  - Davis, Logemann and Loveland 1962 [DLL]
  - Search with unit propagation
- Conflict Driven Clause Learning [CDCL]
  - GRASP,RelSat: Integrate a constraint learning procedure, 1996
- Locality Based Search
  - Emphasis on exhausting local sub-spaces, e.g. Chaff, Berkmin, miniSAT and others, 2001 onwards
  - Added focus on efficient implementation
    - Boolean Constraint Propagation, Decision Heuristics, …

Decision Heuristics



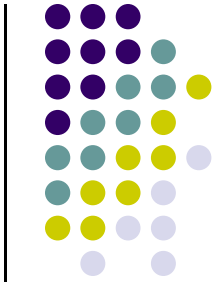
```
while(1) {
   is_conflicting = propagate_unit();

   if(!is_conflicting) {
        if (no_free_vars) return SATISFIABLE;
        make_decision(); // Decision Heuristic
   }
   if(is_conflicting) {
        if (no_unforced_decisions) return UNSAT;
        new_constraint = analyze_conflict(); // Learning
        decision = last_unforced_decision(new_constraint);
        backtrack_to(decision);
        make_forced_decision(invert(decision));
   }
}
```

Locality Based Search



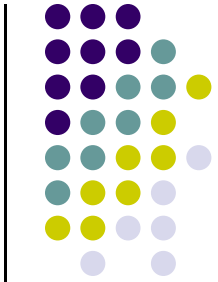
- By focusing on a sub-space, the covered spaces tend to coalesce
  - More opportunities for resolution since most of the variables are common.

Locality Based Search: Decision Heuristics



- Intuitions:
  - Take a more local view of the problem
  - Dynamically identify important constraints and variables
  - Explore space close to recent conflicts
- Not very compute-intensive
- Use relatively simple data structures
- Widely Used
  - VSIDS (Variable State Independent Decaying Sum) in Chaff
  - MiniSAT [ES03]
  - Berkmin [GN02]
  - VMTF in Seige [R04]
  - …

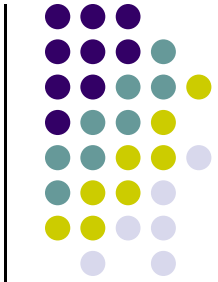
More Deduction



Focused Resolution

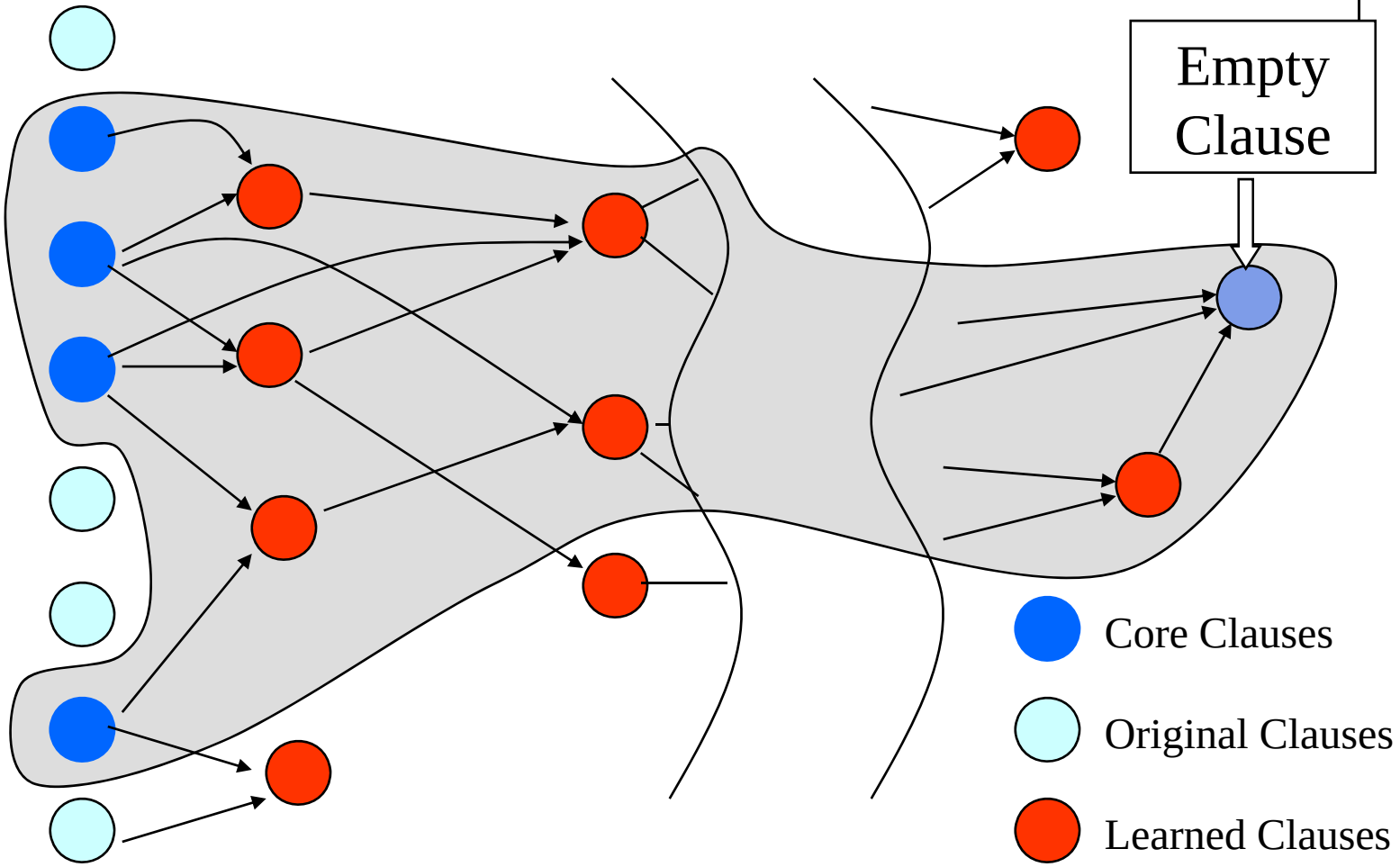
- Significant deduction to simplify the initial CNF instance
- Minisat (SatElite) has efficient implementation [EB05]
  - Variable elimination by resolution
  - Krom subsumption checks
  - Backward subsumption checks
  - Efficient hash based subsumption algorithms
- Hyper-resolution [B02]
- Interestingly, these techniques can also be used during the solving process itself
  - VER – used in Resolve-Expand QBF solver
  - Krom Subsumption – conflict clause minimization in Minisat

Proof Certification and Unsat Cores



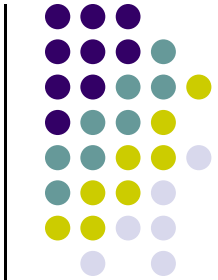
- An UNSAT instance reduces to an empty clause at the end of the deduction/search process
- Can log the resolution trace and independently validate this proof of unsatisfiability
- Additional value in diagnosing the cause of unsatisfiability
  - Unsat Core [ZM03]

The Core as a Checker By-Product



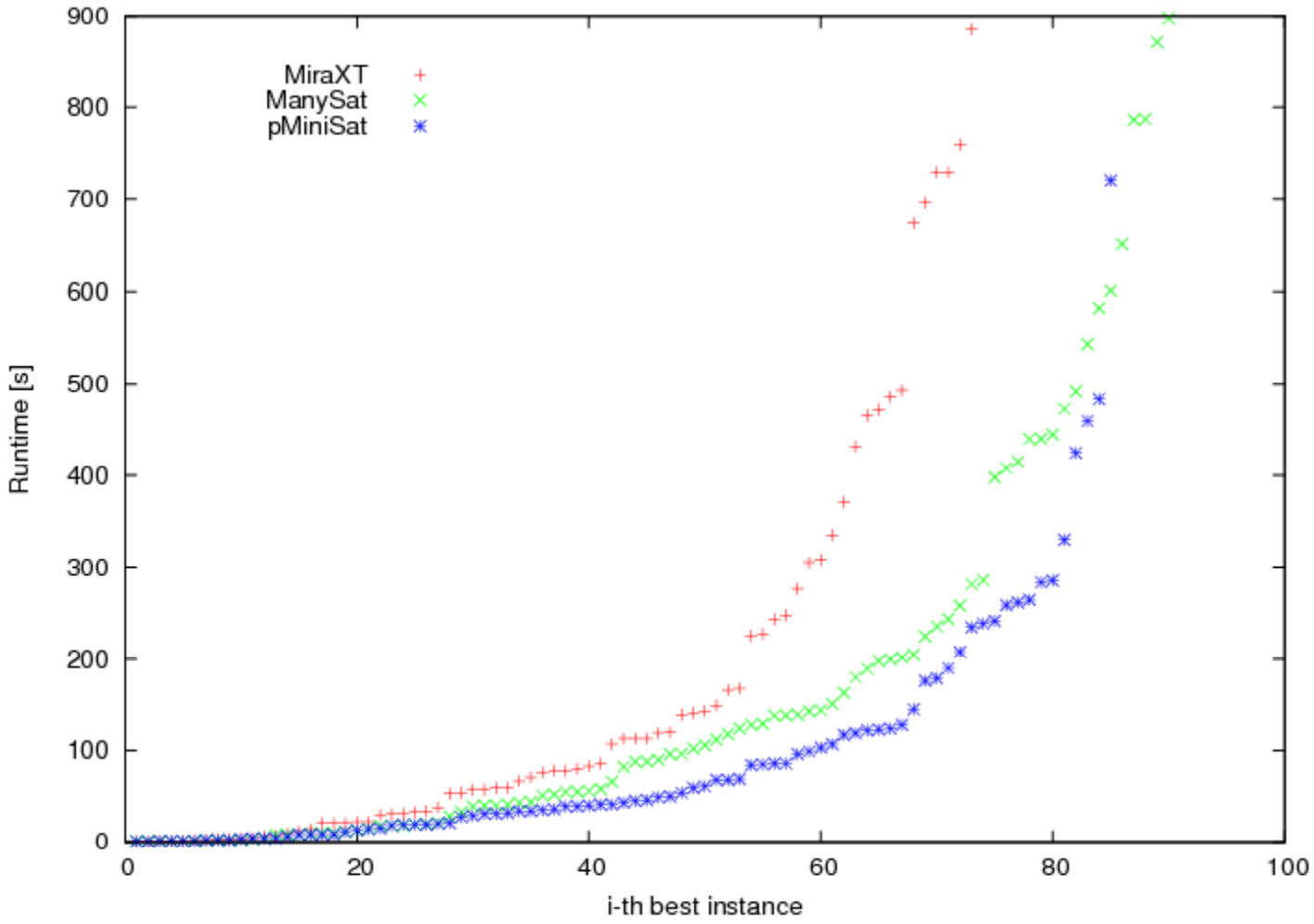
- Unsat Core may be a small fraction of the original clauses
- Multiple applications

Multi-threaded SAT



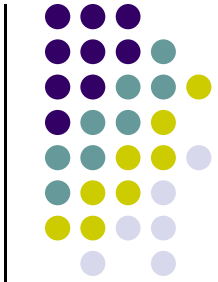
- Increasingly relevant with multi-core processing
- Basic idea:
  - Divide search space among threads
  - Share learned clauses across threads
- Chip-multiprocessors make this real
  - pminiSAT, miraXT [LSB07], manySAT [HJS08]
  - Learned clauses retain relevance across threads
  - Scalability?

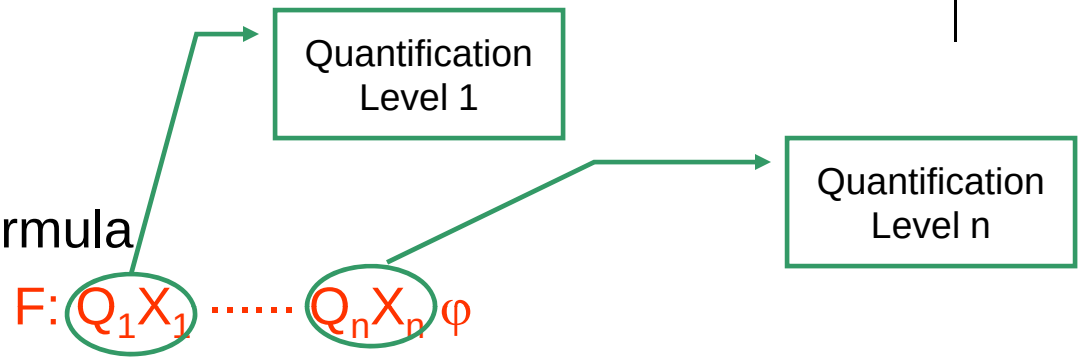
Multi-threaded SAT



http://www-sr.informatik.uni-tuebingen.de/sat-race-2008/analysis.html

Quantified Boolean Formulas





- Quantified Boolean Formula

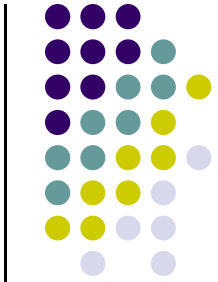
  where Q_i (i=1,···, n) is either ∃ or ∀, φ is a propositional formula
- Example:

  $$\forall u \exists e (u+e')(u'+e)$$

  $$\exists e_4 e_5 \forall u_1 u_2 u_3 \exists e_1 e_2 e_3 \quad f(e_1,e_2,e_3,e_4,e_5,u_1,u_2,u_3)$$
- QBF Problem:

  Is F satisfiable?
- P-Space Complete, theoretically harder than NP-Complete problems such as SAT [GJ79]

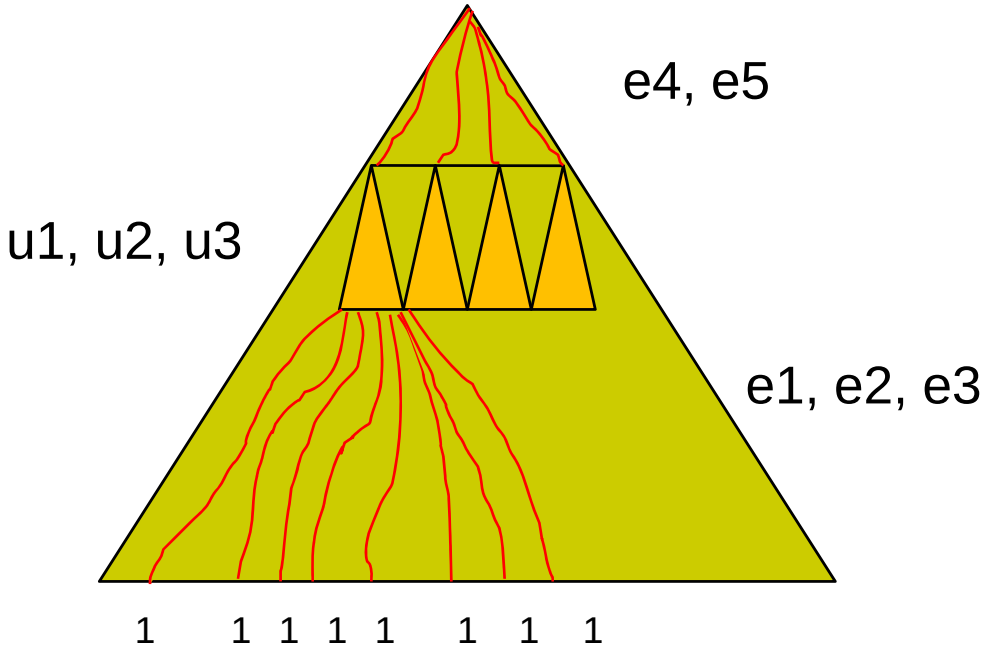
QBF Solvers

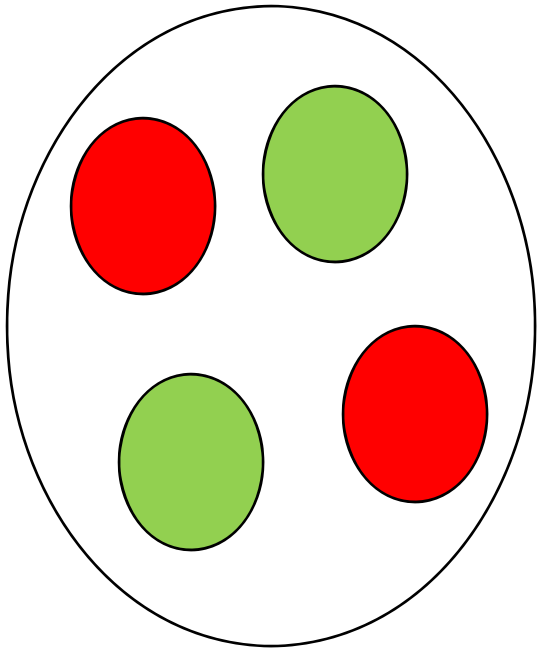


- Like SAT
  - Search and Deduce
- Not like SAT
  - Problem representation
  - Search
  - Deduce

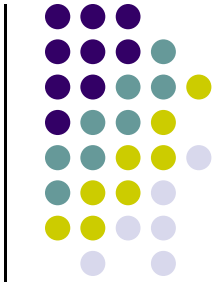
The QBF Search Tree

- $\exists e_4 e_5 \forall u_1 u_2 u_3 \exists e_1 e_2 e_3$ $f(e_1,e_2,e_3,e_4,e_5,u_1,u_2,u_3)$
- Need multiple satisfying assignments
- Need to track conflicting as well as satisfying sub-spaces

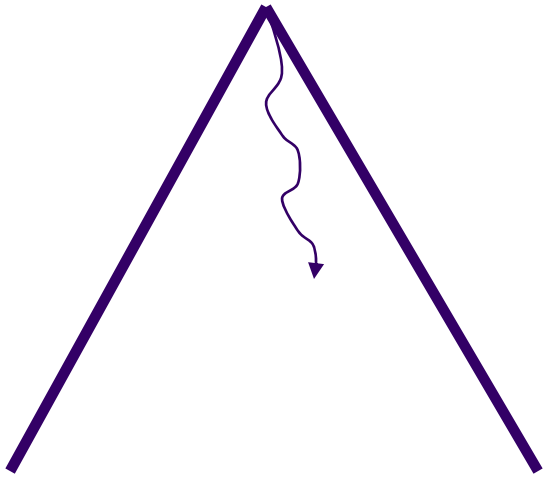




Search Based QBF Algorithms

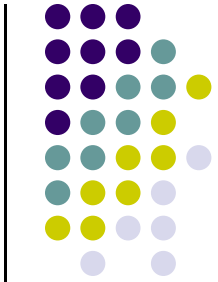


- Work by gradually assigning variables
- A partial assignment $\Rightarrow$

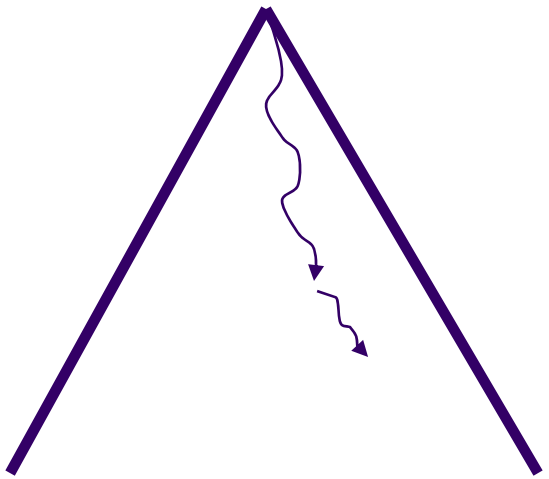


[KGS98]

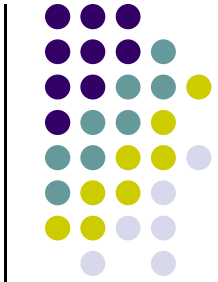
Search Based QBF Algorithms



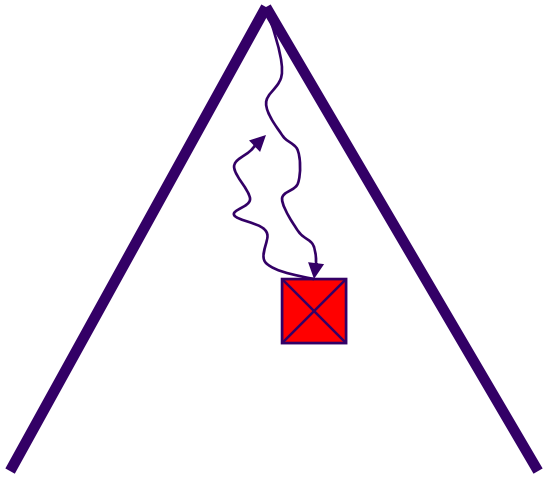
- Work by gradually assigning variables
- A partial assignment $\Rightarrow$
  - Undetermined
    - Continue search



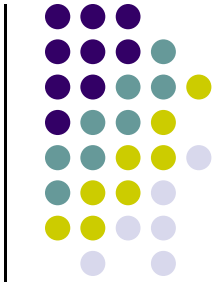
Search Based QBF Algorithms



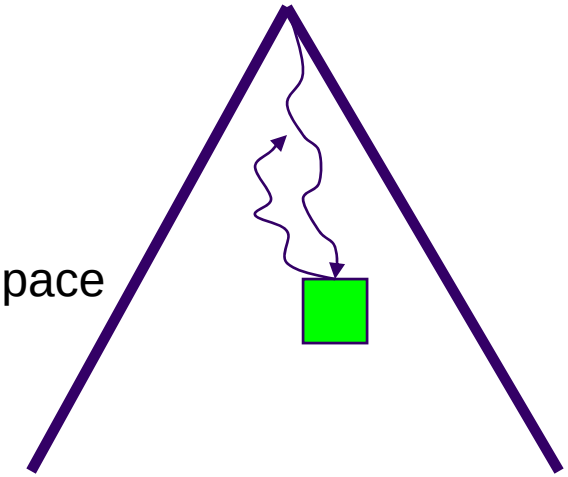
- Work by gradually assigning variables
- A partial assignment $\Rightarrow$
  - Undetermined
  - Conflict
    - Backtrack
    - Record the reason



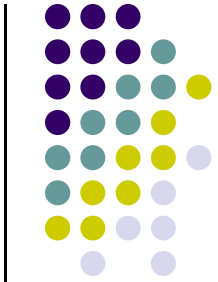
Search Based QBF Algorithms



- Work by gradually assigning variables
- A partial assignment $\Rightarrow$
  - Undetermined
  - Conflict
  - Satisfied
    - Backtrack
    - Determine the covered satisfying space



Impact on Problem Representation

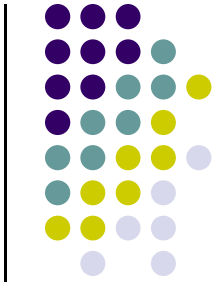


Let $\varphi = C_1\, C_2 \ldots C_m = S_1 + S_2 + \ldots + S_n$

Then:

- $\varphi = (C_1\, C_2 \ldots C_m + S_1 + S_2 + \ldots + S_n)$
  Combined Conjunctive Disjunctive Normal Form (CCDNF)
- $= C_1\, C_2 \ldots C_m\, (S_1 + S_2 + \ldots + S_n)$
- $= (C_1\, C_2 \ldots C_m + \Sigma\text{AnySubset}\{ S_1, S_2, \ldots, S_n\})$
  Augmented CNF (ACNF)
- $= (\Pi\text{AnySubset}\{ C_1, C_2, \ldots, C_m\})(S_1 + S_2 + \ldots + S_n)$
  Augmented DNF

Impact on Problem Representation

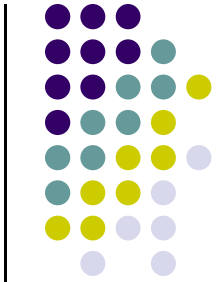


- The disjunctive component can capture satisfied parts of the solution subspace (ACNF) [ZM02]
  - Recorded as cubes during the solution process
    - $f = (a'+b'+c')(a'+b+c)(a+b'+c)(a+b+c') + a'b'c' + ab'c$

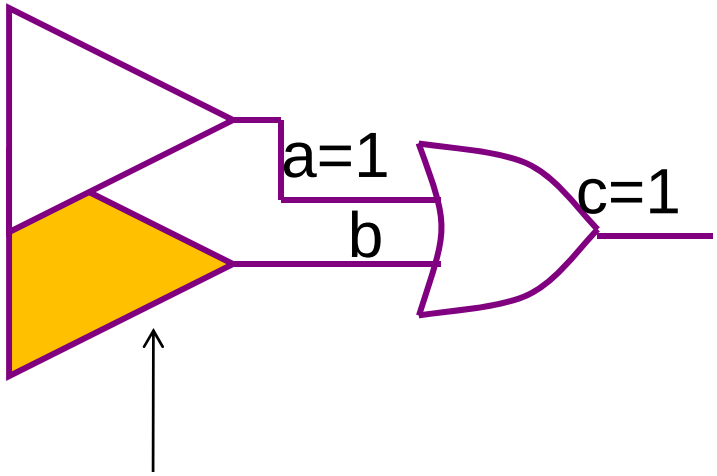
Satisfaction cube
analogous to conflict clause

  - The solver terminates when an empty clause (UNSAT) or a tautology cube (SAT) is obtained.

Impact on Problem Representation

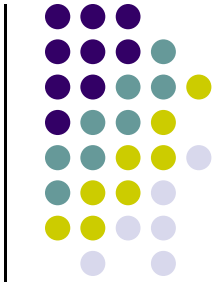


- Can avoid searching irrelevant parts of the space (CCDNF) [Z06]



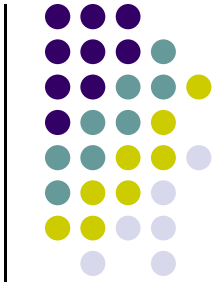
When a=1, values in this circuit are irrelevant
CNF would continue to search for consistent assignments

Impact on Problem Representation



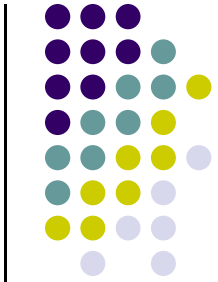
- QBF from Games [SAG+06]
  - Alternating universal and existential quantification
  - Formula structure
    - $Tr_U \rightarrow (I \text{ and } Tr_E \text{ and } G_E)$
    - Tr_U : Transition rules for U vars
    - I: Initial Axioms
    - G_E: Goal axioms for E
  - Dual DNF CNF representation
    - DNF for Tr_U, CNF for the Rest

Solving by Quantifier Elimination



- Resolution based solvers [BKF95]:
  - Eliminate quantifiers from inside out
    - Existential: Resolution with CNF
    - Universal: Trivial with CNF
- Resolve and Expand [B05a]
  - Include non-internal universal quantifier elimination using expansion (duplication)
- Symbolic skolemization to eliminate existential quantifiers
  - Skizzo [B05b]
- Very sensitive to order of operations
- Greater success compared to search based solvers

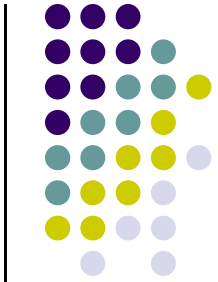
QBF Competition: Success



Largest Solved Instances, QBF Eval 2008

Instance	Class	Solver (time)	totalVars	existsVars	forallVars	totalClauses	totalLits	existsAltern	forallAltern
c1_BMC_p2_k1024	Fixed	quantor3.0(24.72)	1110430	1110420	4	2812460	6641870	2	1
c1_BMC_p1_k1024	Fixed	quantor3.0(47.51)	1110430	1110420	4	2812460	6641870	2	1
vonNeumann-ripple-carry-12-c	Fixed	quantor3.0(1.8)	567460	567148	312	832132	1	2	1
c1_BMC_p2_k512	Fixed	quantor3.0(12.96)	566106	566102	4	1451240	3596820	2	1
c1_BMC_p1_k512	Fixed	quantor3.0(24.54)	566106	566102	4	1451240	3596820	2	1

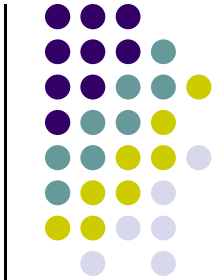
QBF Competition: Challenges



Smallest Unsolved Instances, QBF Eval 2008

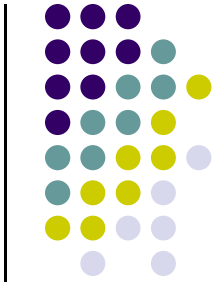
Instance	Class	totalVars	existsVars	forallVars	totalClauses	totalLits	existsAltern	forallAltern
test4_quant_squaring2	Fixed	326	302	24	868	2208	2	1
test4_quant2	Fixed	326	302	24	868	2208	2	1
test3_quant4	Fixed	344	324	20	923	2419	2	1
test4_quant4	Fixed	446	422	24	1204	3120	2	1
C499.blif_0.10_0.20_0_1_out_exact	Fixed	838	826	12	2393	5702	2	1

This Talk



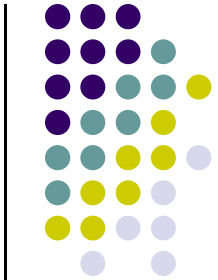
- Successful application of diverse CS techniques
  - Logic (Deduction and Solving)
  - Search
  - Caching
  - Randomization
  - Data structures
  - Cache efficient algorithms
- Open challenges…
  - Limited understanding of why the algorithms work
  - Dynamic application of strategies
  - QBF

Summary



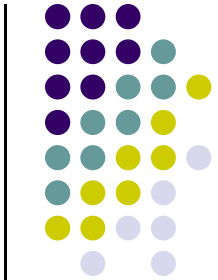
- SAT: Significant shift from theoretical interest to practical impact.
- Quantum leaps between generations of SAT solvers
- Presence of drivers results in maximum progress.
  - Electronic design automation – primary driver and main beneficiary
  - Software verification- the next frontier?
- Opens attack on even harder problems
  - SMT, Max-SAT, QBF…

References



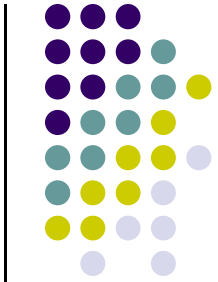
- [GJ79] Michael R. Garey and David S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman and Company, San Francisco, 1979
- [DP 60] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM, 7:201–215, 1960*
- [DLL62] M. 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



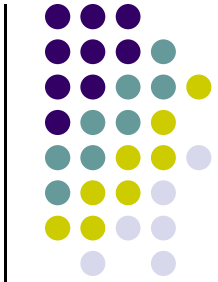
- [H07] J. Huang, “The effect of restarts on the efficiency of clause learning,” Proceedings of the Twentieth International Joint Conference on Automated Reasoning, 2007
- [MMZ+01] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik. Chaff: Engineering and efficient sat solver. In *Proc., 38th Design 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
- [ZM02] L. Zhang and S. Malik, Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation, In Principles and Practice of Constraint Programming - CP 2002

References



- [KGS98] M. Cadoli, A. Giovanardi, M. Schaerf. An Algorithm to Evaluate Quantified Boolean Formulae. In Proc. of 16th National Conference on Artificial Intelligence (AAAI-98)
- [Z06] L. Zhang, Solving QBF with Combined Conjunctive and Disjunctive Normal Form, In Proc. of National Conference on Artificial Intelligence, 2006
- [SAG+06] Ashish Sabharwal , Carlos Ansotegui, Carla P. Gomes, Justin W. Hart and Bart Selman, QBF Modeling: Exploiting Player Symmetry for Simplicity and Efficiency, In Theory and Applications of Satisfiability Testing - SAT 2006
- [BKF95] Hans Kleine Buning, Marek Karpinski, and Andreas Flogel. Resolution for Quantified Boolean Formulas. Information and Computation 117(1): 12-18 (1995).
- [B05a] A. Biere, Resolve and Expand, In Theory and Applications of Satisfiability Testing, 2005
- [B05b] Marco Benedetti, sKizzo: A Suite to Evaluate and Certify QBFs, In Proceedings of Automated Deduction – CADE-20, 2005