SAT

Pete Manolios
Northeastern

Formal Methods, Lecture 9

October 2008

Review of SAT, NP Completeness

- kSAT
  - Literals: variables or their negations
  - Clause: disjunction of literals
  - CNF formula (Conjunctive Normal Form): conjunction of clauses
  - kCNF: CNF formula w/ at most k literals per clause
  - kSAT: The set of satisfiable kCNF formulas
- Recall: SAT (= set of satisfiable CNF formulas) is NP-complete
  - NP: languages whose membership can be verified in P-time
  - NPC:
    - Hardest problems in NP
    - P-time algorithms for an NPC problem means P-time algorithm for every problem in NP
- 3SAT is NP-complete: Can reduce SAT to 3SAT ($SAT \leq_p 3SAT$)
  - Can define a P-time function f s.t. $x \in SAT$ iff $f(x) \in 3SAT$

SAT Remarks

- Can use SAT to check validity
- How?
  - ϕ is valid iff $\neg\phi$ is not SAT
  - ϕ is SAT iff $\neg\phi$ is not valid
- So, does that prove that validity is NPC?
- Random SAT:
  - Phase transition phenomena, e.g., ~4.26 for 3SAT
  - Local search methods
  - Algorithms: WalkSAT, Survey propagation, ...
- Special cases: 2SAT, Horn SAT, Dual-horn SAT, MAX SAT

Algorithms for SAT

- Modern SAT solvers accept input in CNF
  - Dimacs format:
  - 1 -3 4 5 0
  - 2 -4 7 0
  - …
- Davis & Putnam Procedure (DP)
  - Dates back to the 50's
  - Based on resolution (modern algorithms are not)
  - Helps to explain learning

Resolution

Resolution rule:

$$\frac{C, v \qquad D, \neg v}{C, D} \qquad \neg v, v \notin C \cup D$$

- Soundness of rule: above line implies below line
- Also below line is SAT, so is above line (w/ side conditions)
- DP SAT algorithm
  - Base case: empty clause: UNSAT
  - Base case: no clauses: SAT
  - Remove clauses containing pure literals
  - Choose var, perform all possible resolutions, remove trivial clauses and clauses containing x
  - Problem: space blow-up

Boolean Constraint Propagation

Unit resolution rule:

$$\frac{C, \neg \ell \qquad \ell}{C}$$

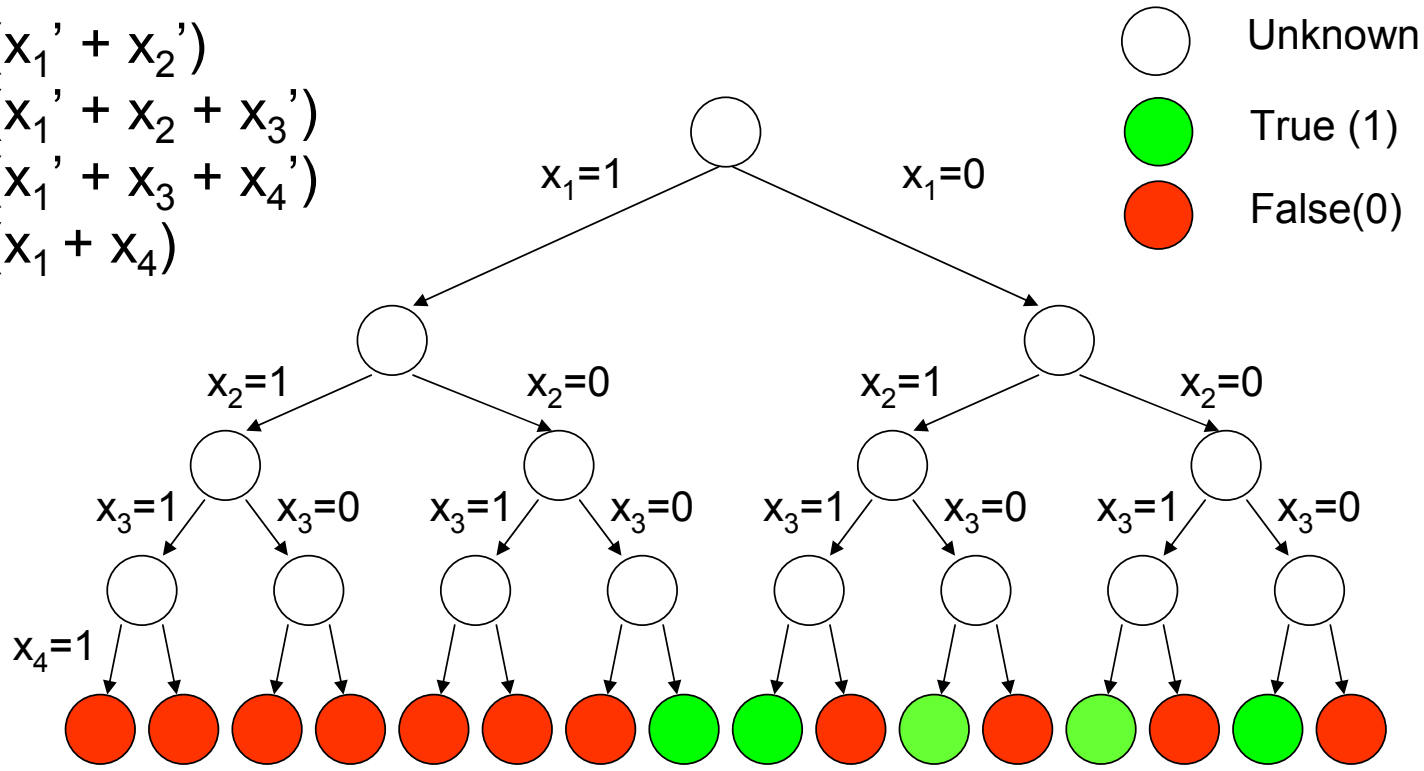
- BCP: given a set of clauses including $\{\ell\}$
  - remove all other clauses containing ℓ (subsumption)
  - remove all occurrences of $\neg\ell$ in clauses (unit resolution)
  - repeat until a fixpoint is reached
- Shannon expansion: $f(x) \equiv [x \wedge f(1)] \vee [x \wedge f(0)]$

DPLL SAT Algorithm

- BCP
- Base case: empty clause: UNSAT
- Remove clauses containing pure literals
- Base case: no clauses: SAT
- Choose some var, say x (has to appear in both phases)
  - Add {x} and recursively call DPLL
  - Add {¬x} and recursively call DPLL
  - If one of the calls returns SAT, return SAT
  - Else return UNSAT
- Correctness follows from Shannon expansion
- In contrast to DP, space is not a problem

DPLL is DFS Search of SAT-Tree

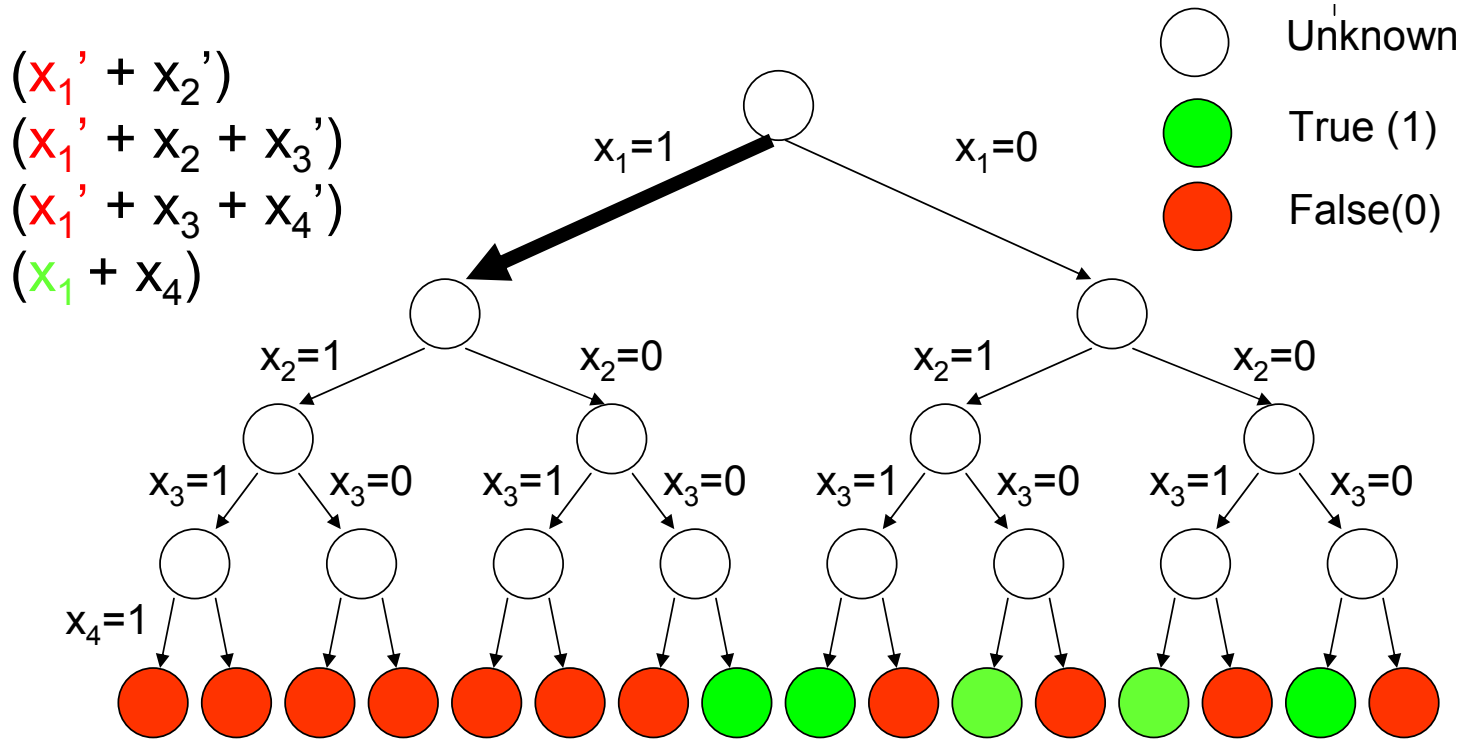
$(x_1' + x_2')$
$(x_1' + x_2 + x_3')$
$(x_1' + x_3 + x_4')$
$(x_1 + x_4)$



Lintao Zhang

Microsoft Research

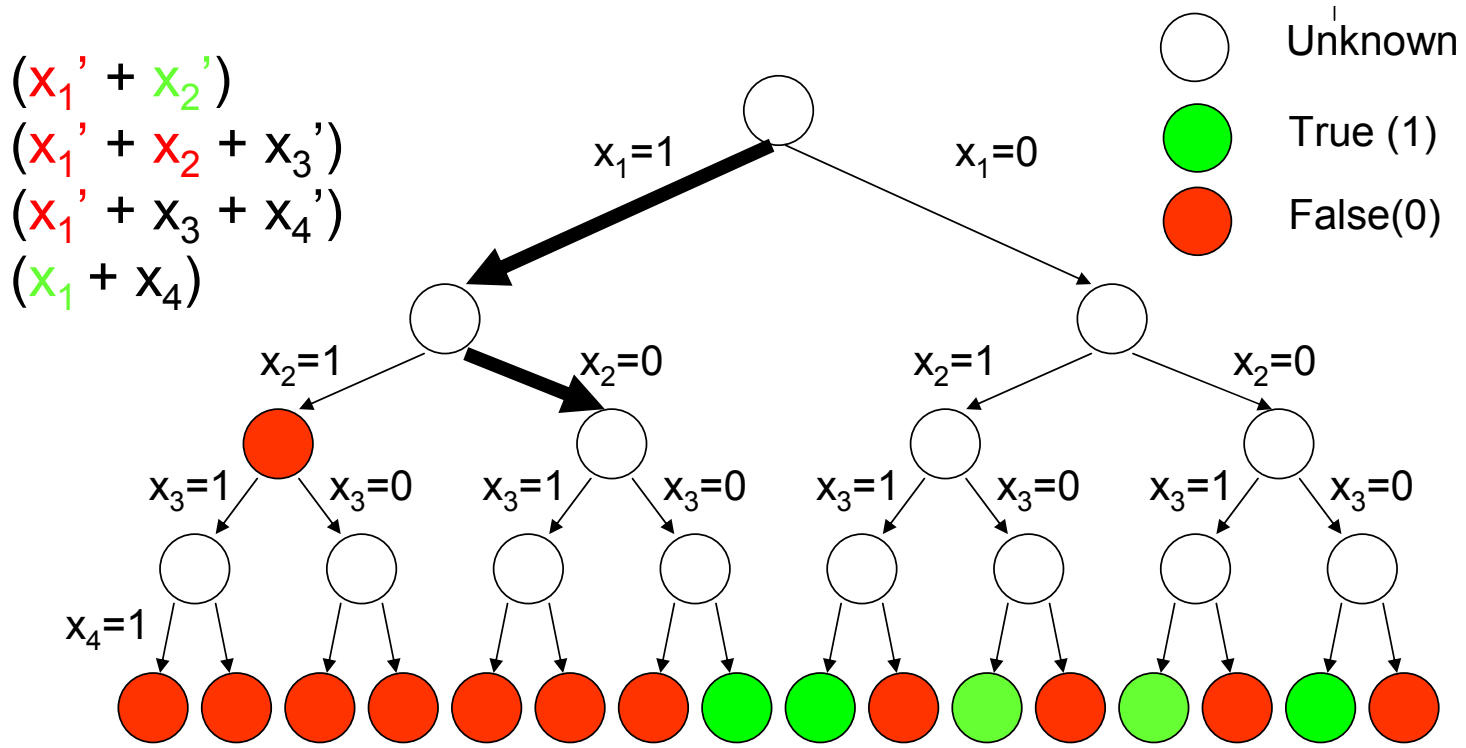
DPLL is DFS Search of SAT-Tree



Lintao Zhang

Microsoft Research

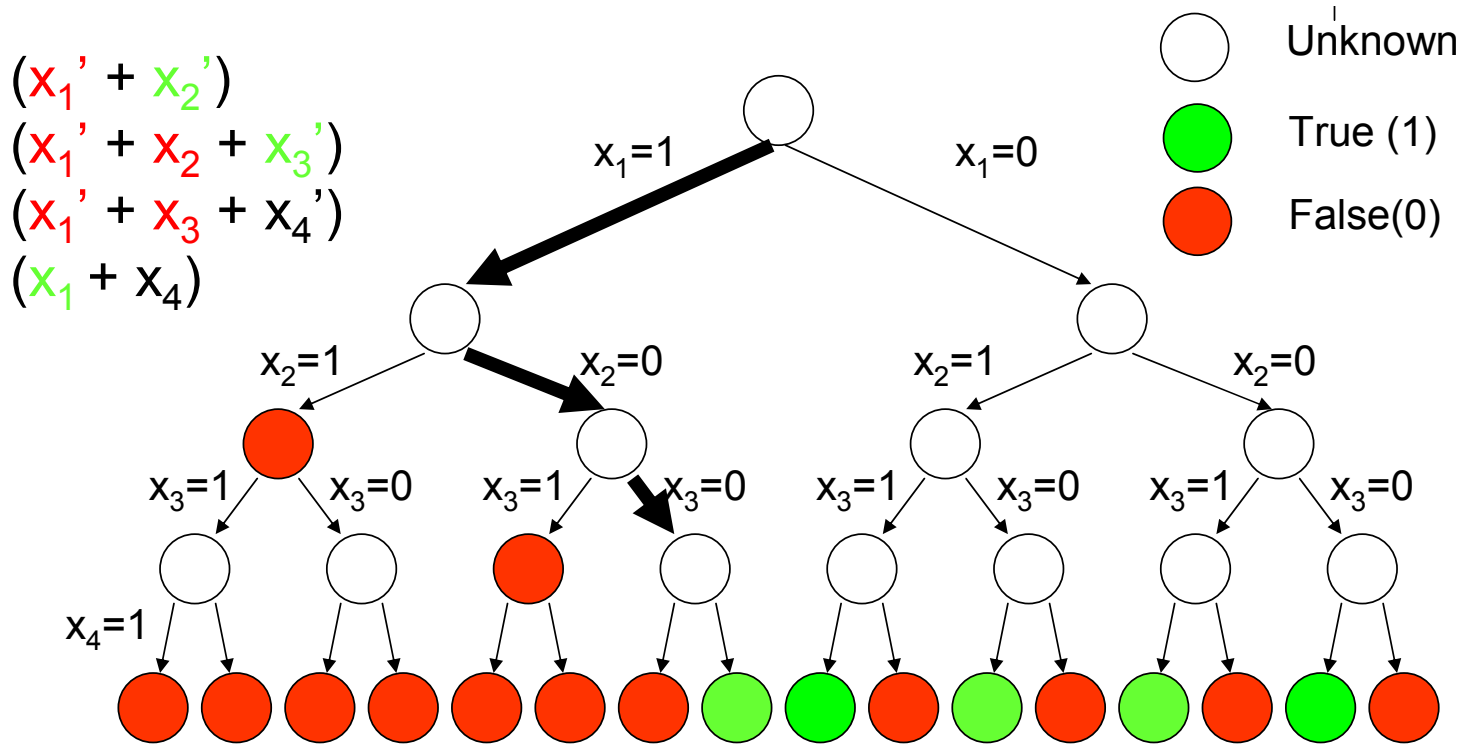
DPLL is DFS Search of SAT-Tree



Lintao Zhang

Microsoft Research

DPLL is DFS Search of SAT-Tree



Lintao Zhang

Microsoft Research

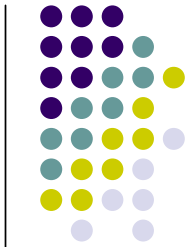
Modern DPLL

- Decision heuristics
  - Many have been tried; we'll look at VSIDS
- Efficient BCP
  - BCP is the workhorse of modern SAT solvers
  - 2-literal watching
- Non-chronological backtracking
  - Can make a huge difference
- Clause learning
  - Records non-trivial implications discovered during search
  - Avoids re-exploring similar parts of state space
  - A disciplined form of resolution, but can still lead to space blow-up
- Preprocessing: limited resolution, subsumption, etc
- Restarts: clause learning helps guide SAT solver to solution

Decision Heuristics

- How do we decide what variable to split on?
- Variable State Independent Decaying Sum (VSIDS)
  - Keeps a score for each phase of a variable
  - Initially: the number of occurrences of a literal
  - Increases score by a constant whenever an added clause contains the variable
  - Periodically all the scores are divided by a constant
  - Choose free variable with the highest combined score
- VSIDS score is a literal occurrence count with higher weight on the more recently added clauses.
- VSIDS scores do not depend on the variable assignment
- Cheap to maintain (takes small percentage of the total run time)

Conflict Driven Learning and Non-chronological Backtracking

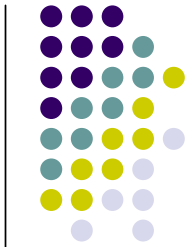


x1 + x4
x1 + x3’ + x8’
x1 + x8 + x12
x2 + x11
x7’ + x3’ + x9
x7’ + x8 + x9’
x7 + x8 + x10’
x7 + x10 + x12’

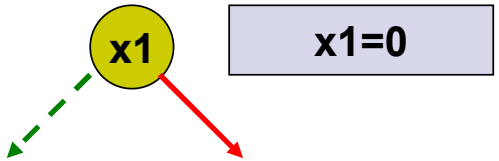
Lintao Zhang

Microsoft Research

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'

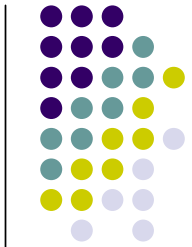


x1=0

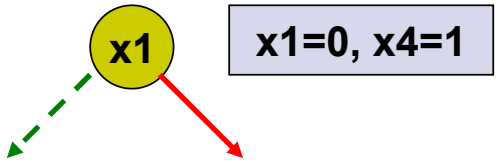
Lintao Zhang

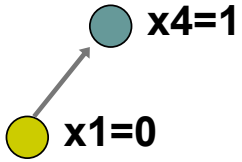
Microsoft Research

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'

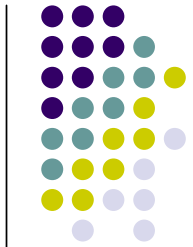




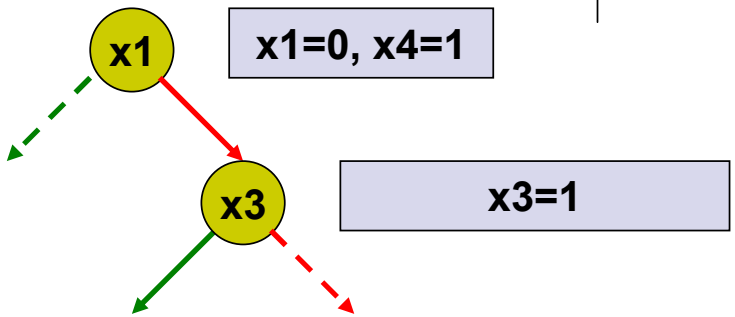
Lintao Zhang

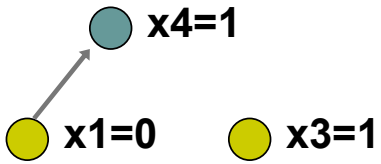
Microsoft Research

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'

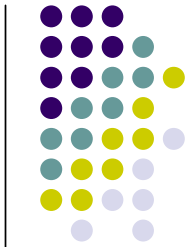




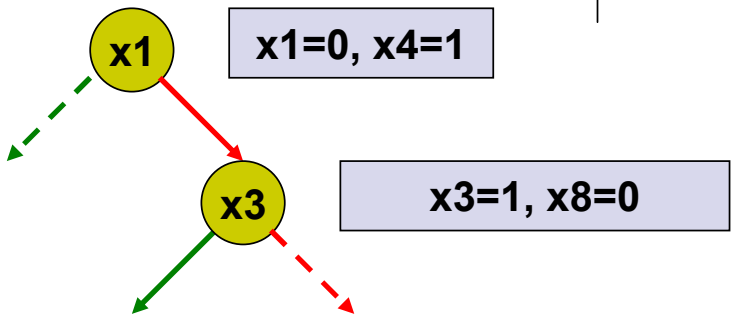
Lintao Zhang

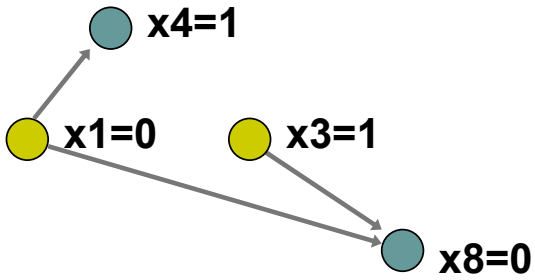
Microsoft Research

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'

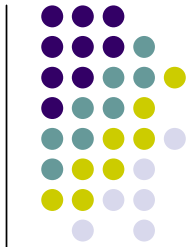




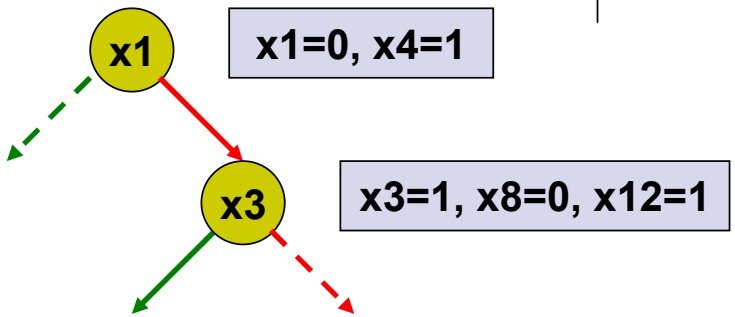
Lintao Zhang

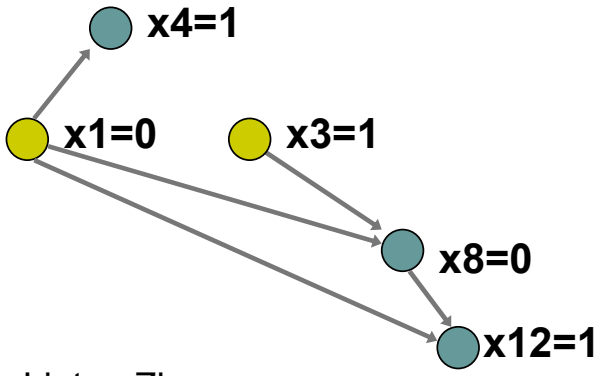
Microsoft Research

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'

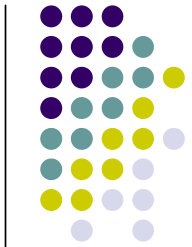




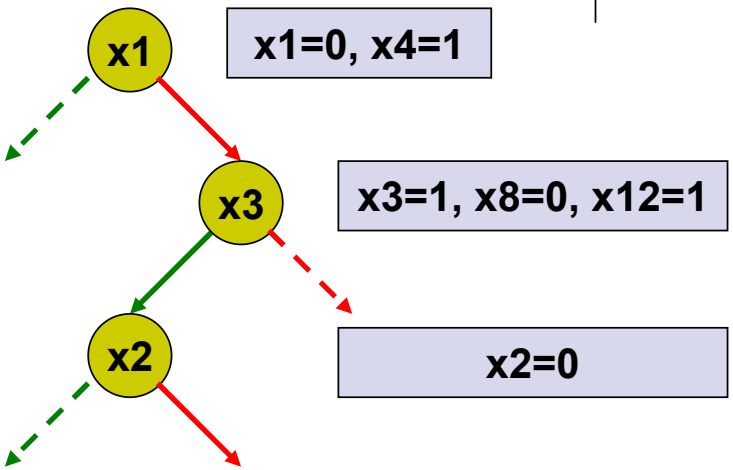
Lintao Zhang

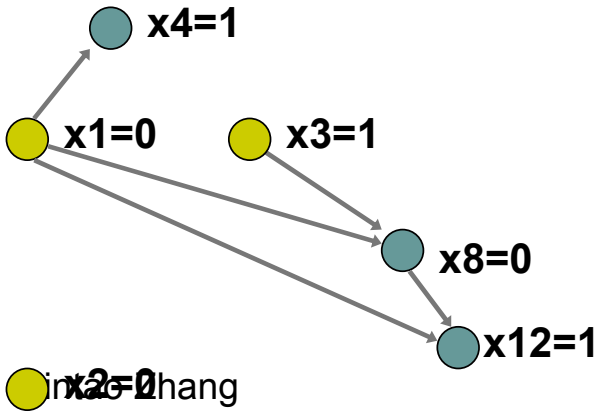
Microsoft Research

Conflict Driven Learning and Non-chronological Backtracking



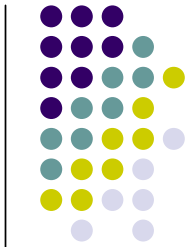
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'



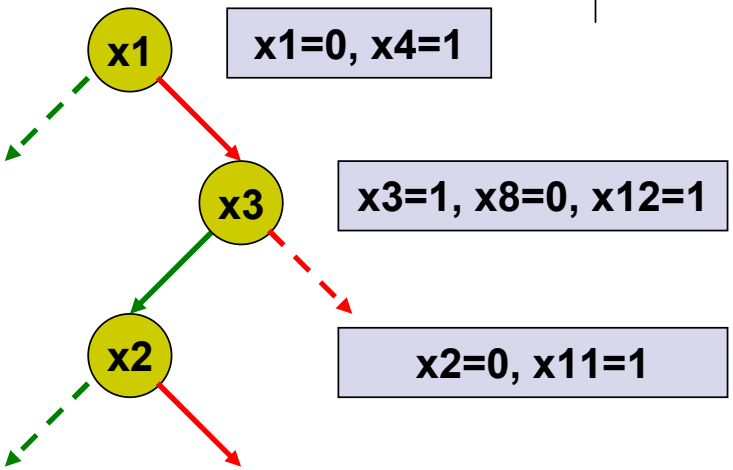


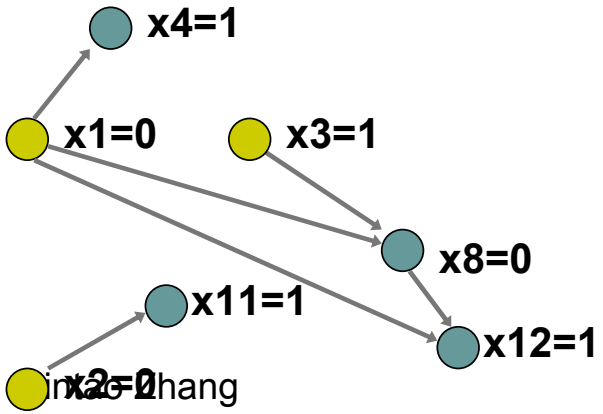
Microsoft Research

Conflict Driven Learning and Non-chronological Backtracking



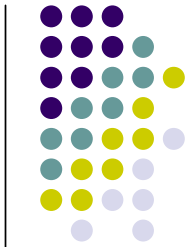
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'



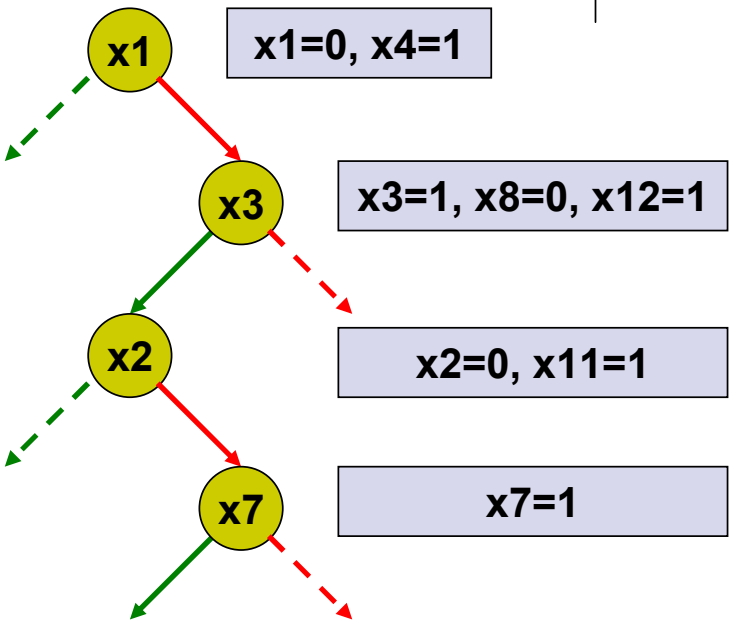


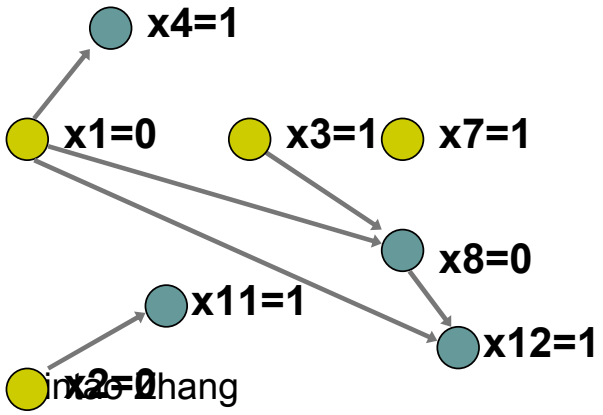
Microsoft Research

Conflict Driven Learning and Non-chronological Backtracking



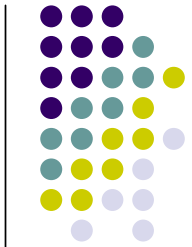
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'



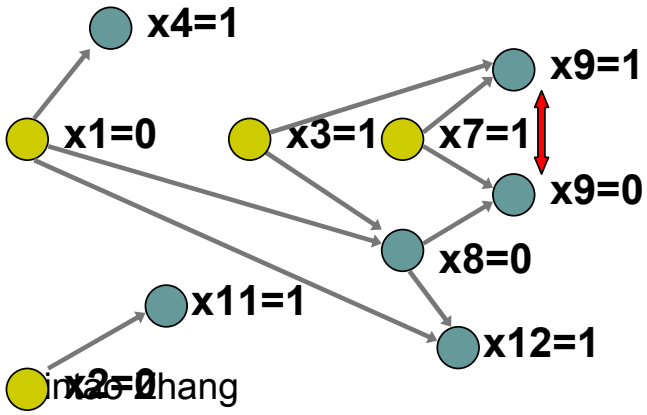


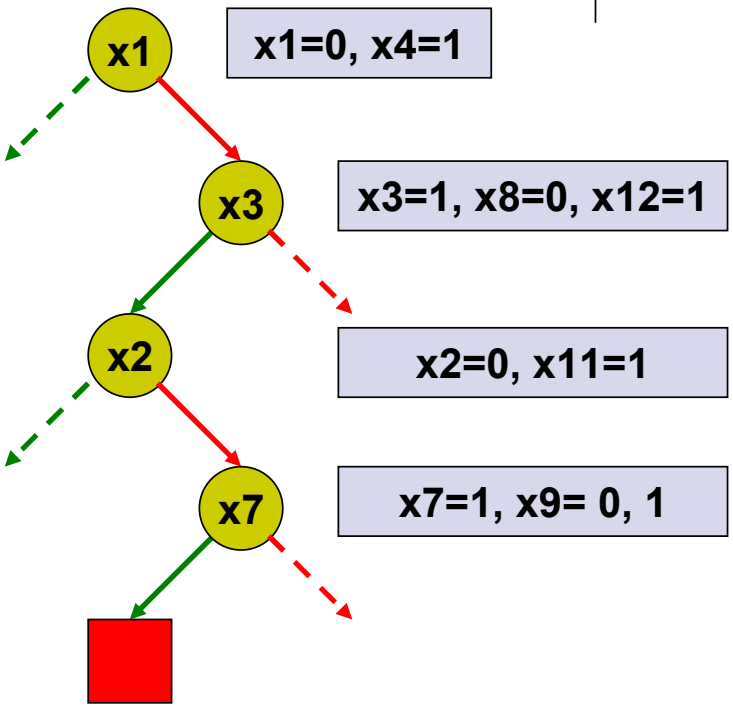
Microsoft Research

Conflict Driven Learning and Non-chronological Backtracking



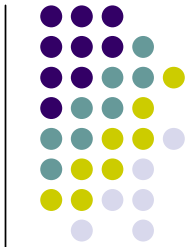
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'



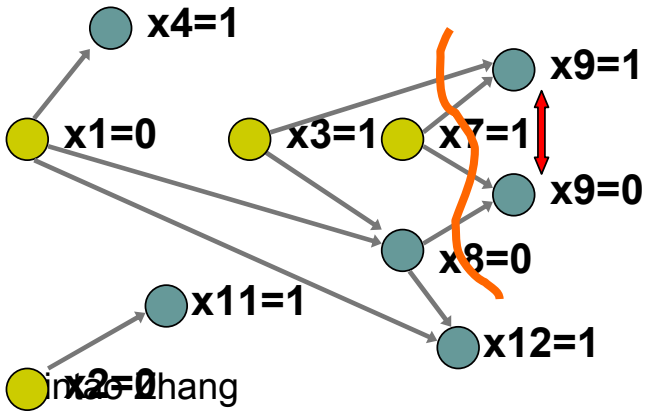


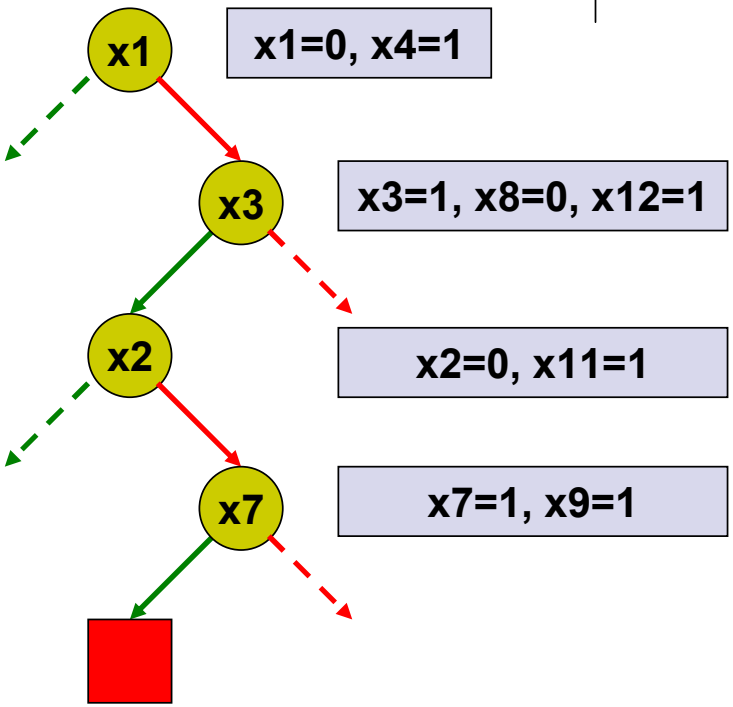
Microsoft Research

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'

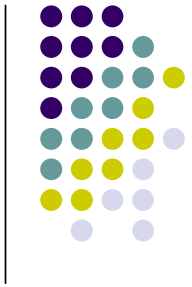




x3=1∧x7=1∧x8=0 → conflict

Microsoft Research

Contra-proposition:



- If a implies b, then b' implies a'

$$x3=1 \wedge x7=1 \wedge x8=0 \rightarrow \text{conflict}$$

$$\text{Not conflict} \rightarrow (x3=1 \wedge x7=1 \wedge x8=0)'$$

$$\text{true} \rightarrow (x3=1 \wedge x7=1 \wedge x8=0)'$$

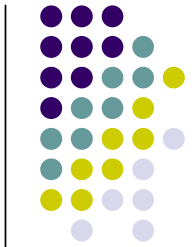
$$(x3=1 \wedge x7=1 \wedge x8=0)'$$

$$(x3' + x7' + x8)$$

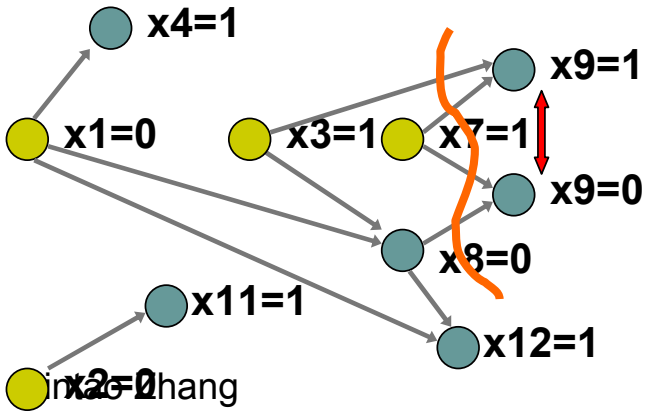
Lintao Zhang

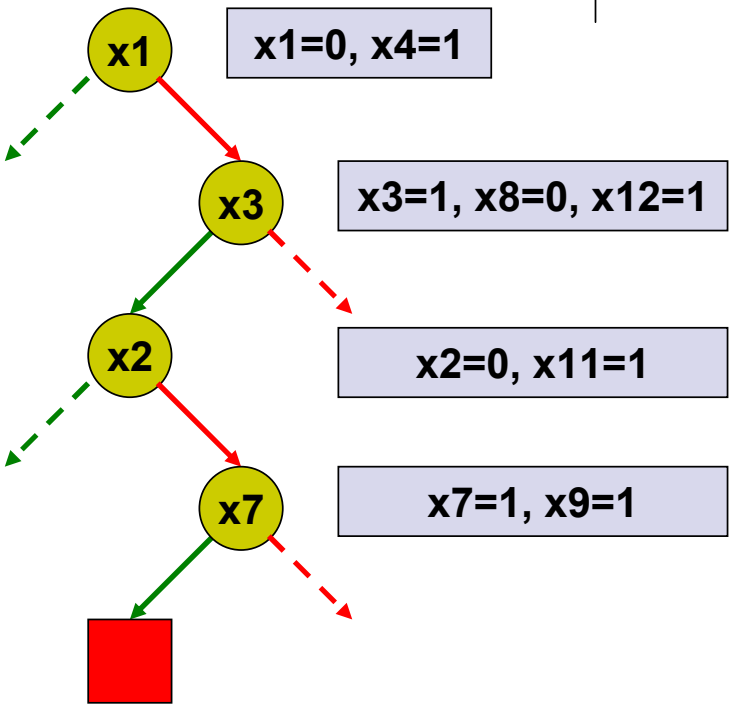
Microsoft Research

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'

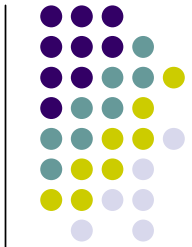




x3=1∧x7=1∧x8=0 → conflict

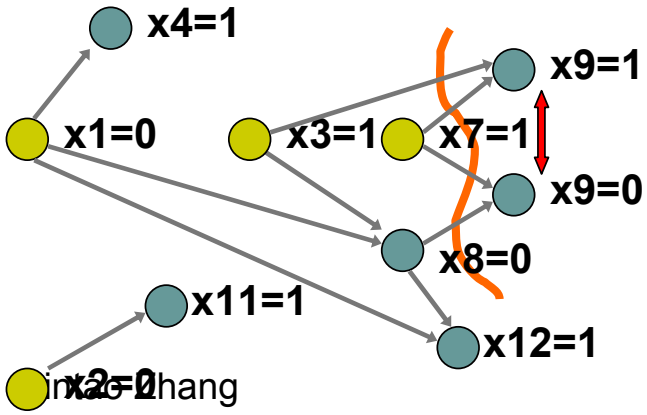
Add conflict clause: x3'+x7'+x8

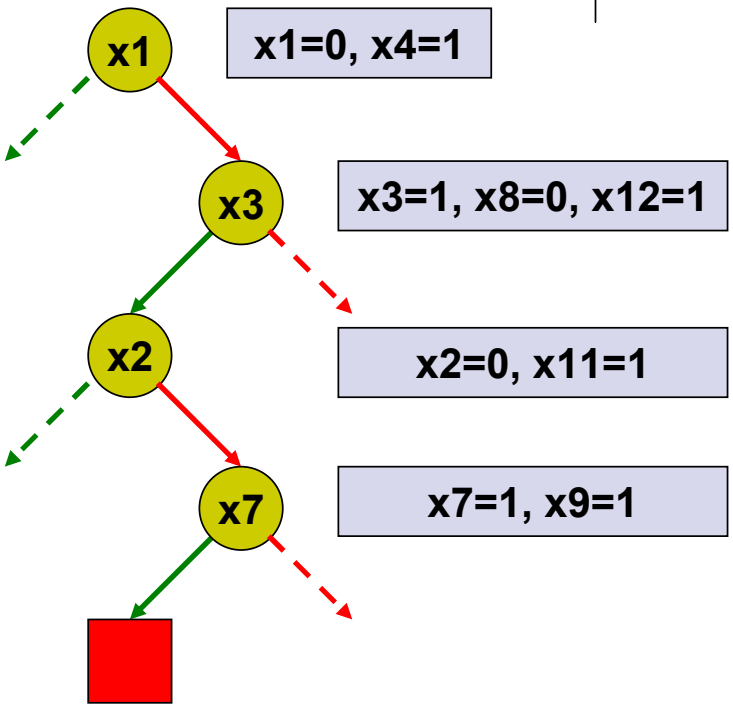
Conflict Driven Learning and Non-chronological Backtracking



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

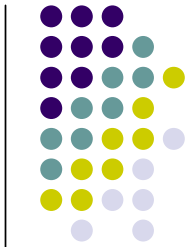




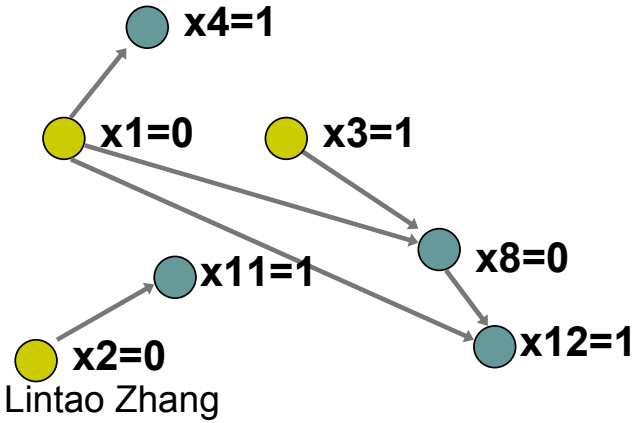
x3=1∧x7=1∧x8=0 → conflict

Add conflict clause: x3'+x7'+x8

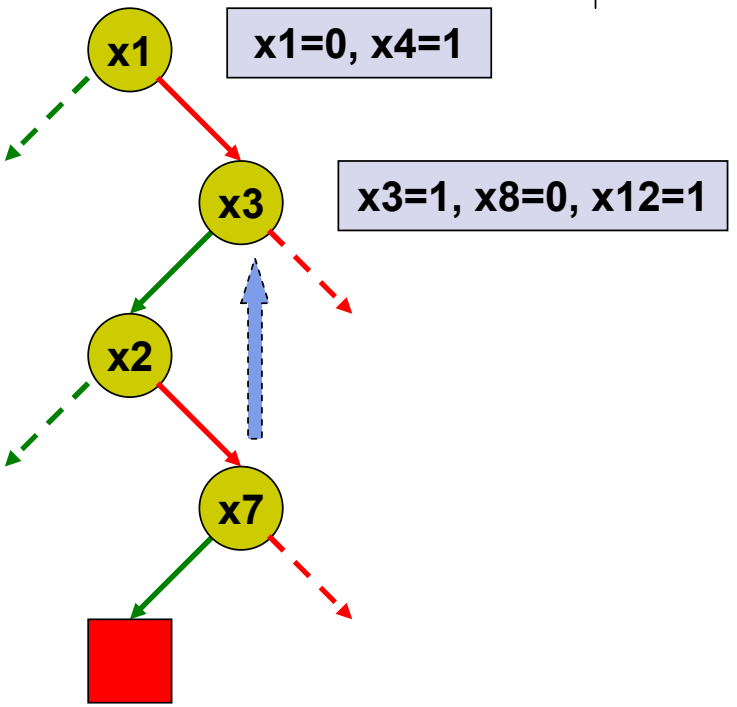
DLL with Non-Chronological Backtracking and Learning



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



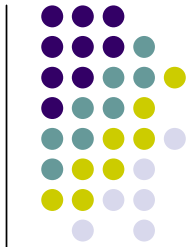
Lintao Zhang



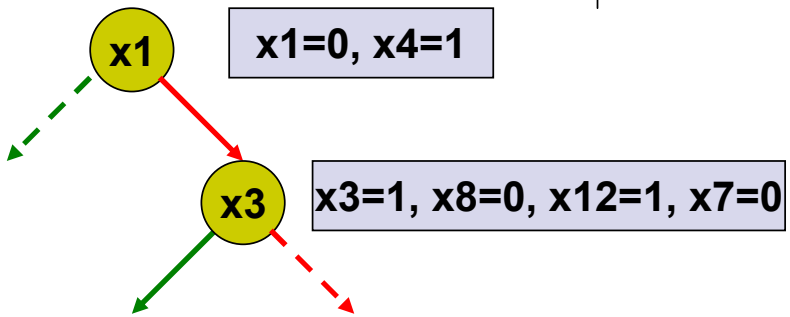
Backtrack to the decision level of x3=1: x7 = 0

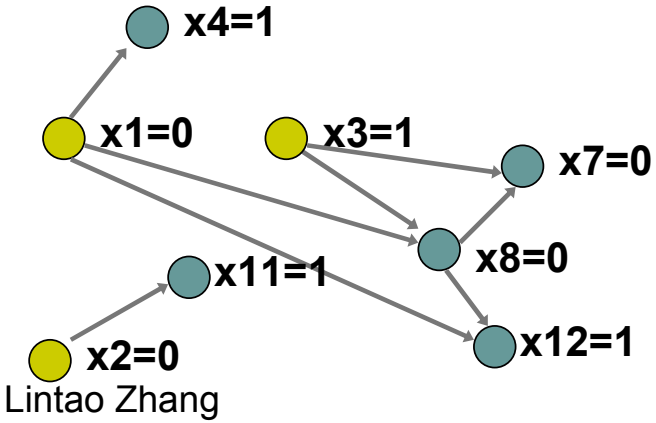
Microsoft Research

DLL with Non-Chronological Backtracking and Learning



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





Lintao Zhang

Microsoft Research