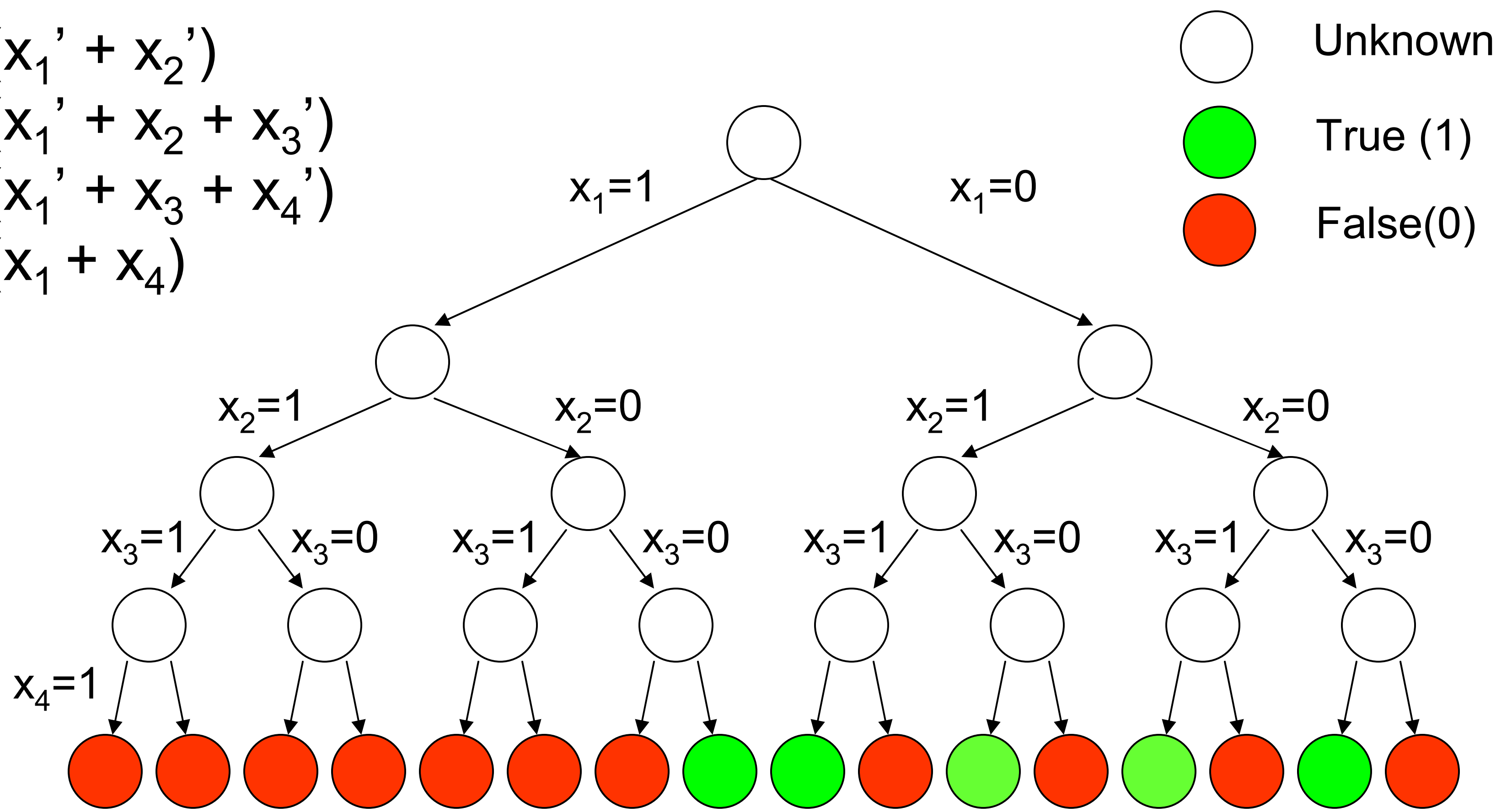
Search Tree of SAT Problem

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



Deduction Rules for SAT

- Unit Literal Rule: If an unsatisfied clause has all but one of its literals evaluate to 0, then the *free* literal must be implied to be 1.

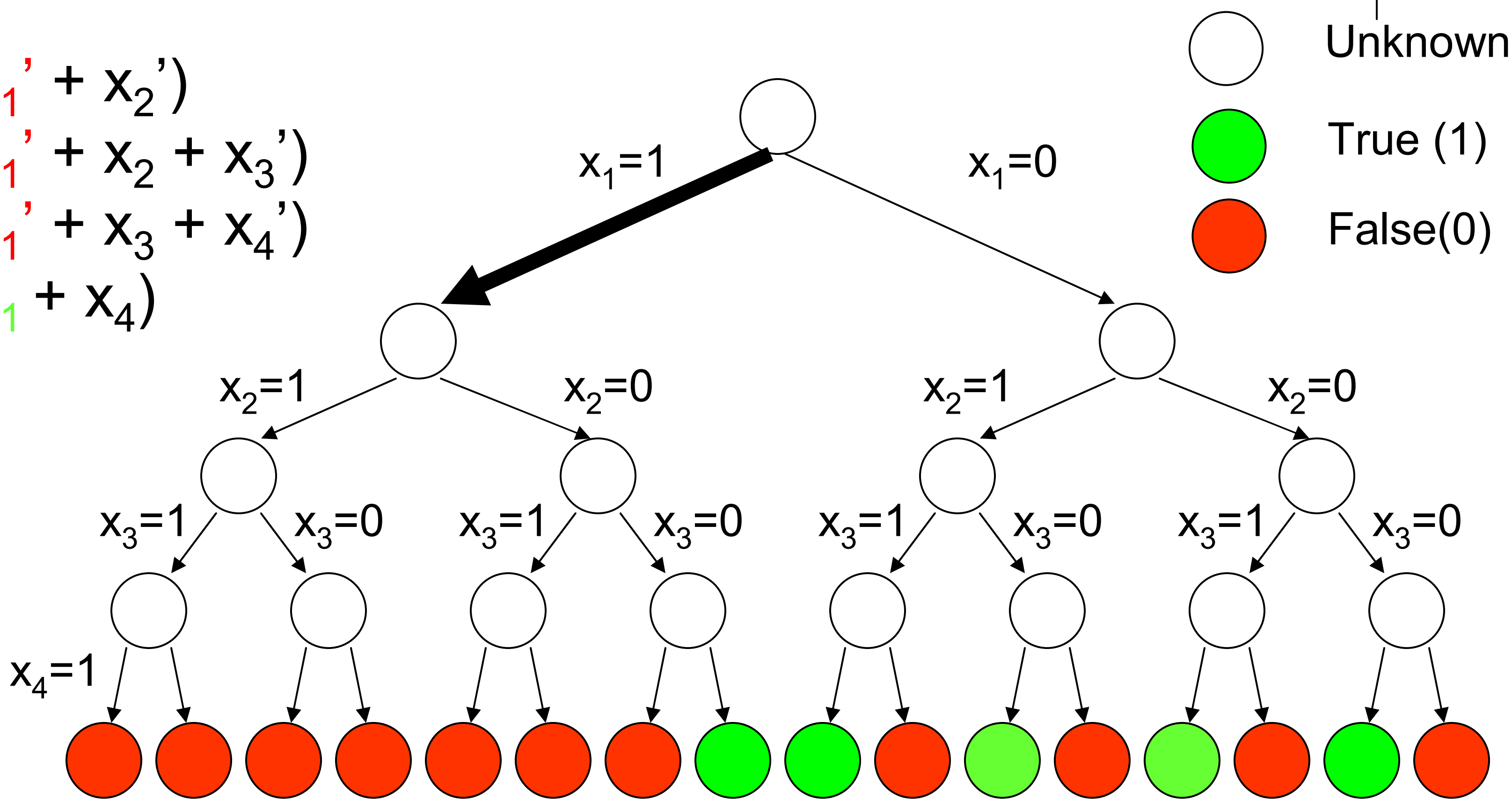
  $$(a + b + c)(d' + e)(a + b + c' + d)$$

- Conflicting Rule: If all literals in a clause evaluate to 0, then the formula is unsatisfiable in this branch.

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

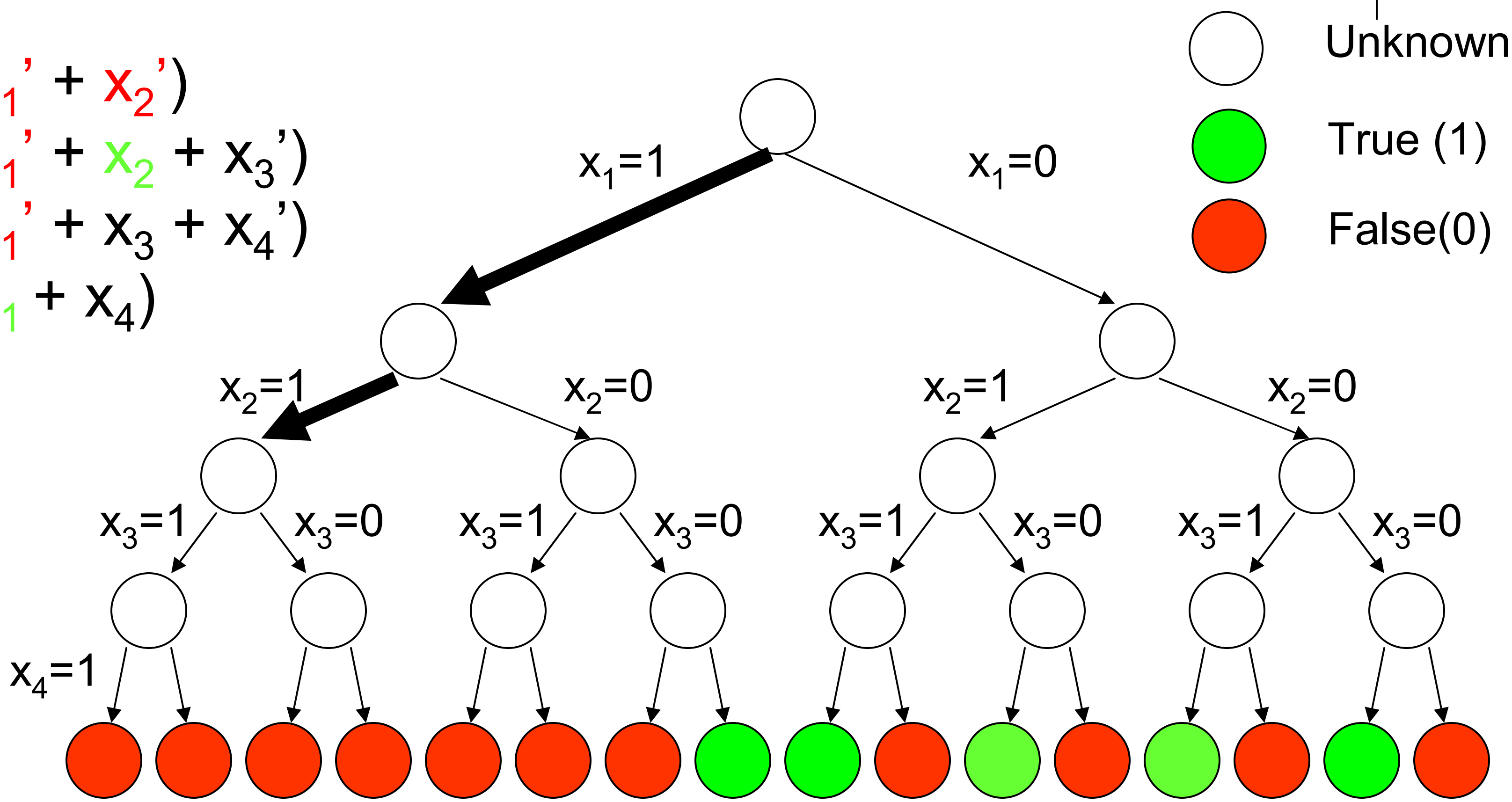
Search Tree of SAT Problem

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



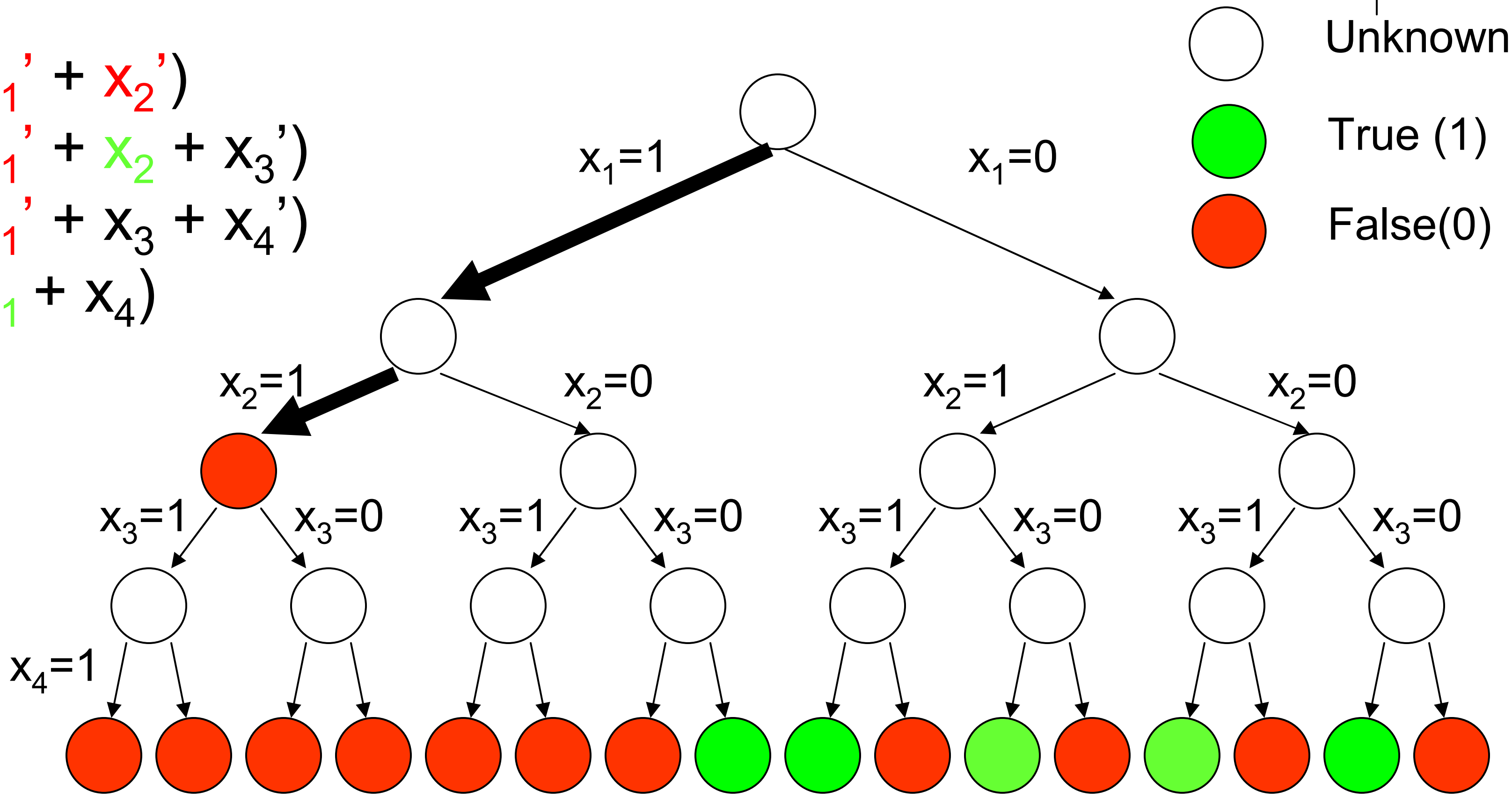
Search Tree of SAT Problem

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



Search Tree of SAT Problem

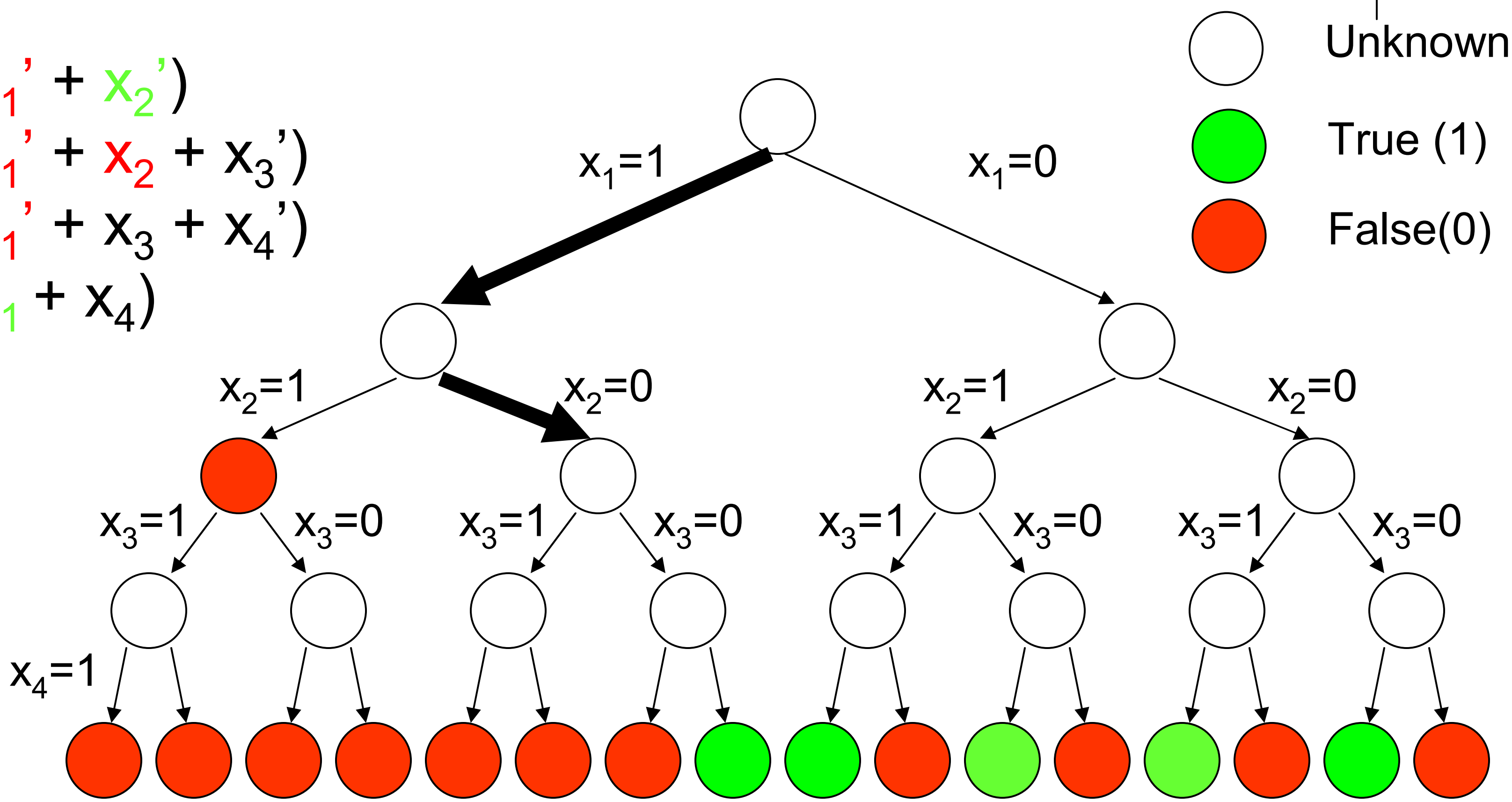
$(x_1' + x_2')$
$(x_1' + x_2 + x_3')$
$(x_1' + x_3 + x_4')$
$(x_1 + x_4)$



Lintao Zhang

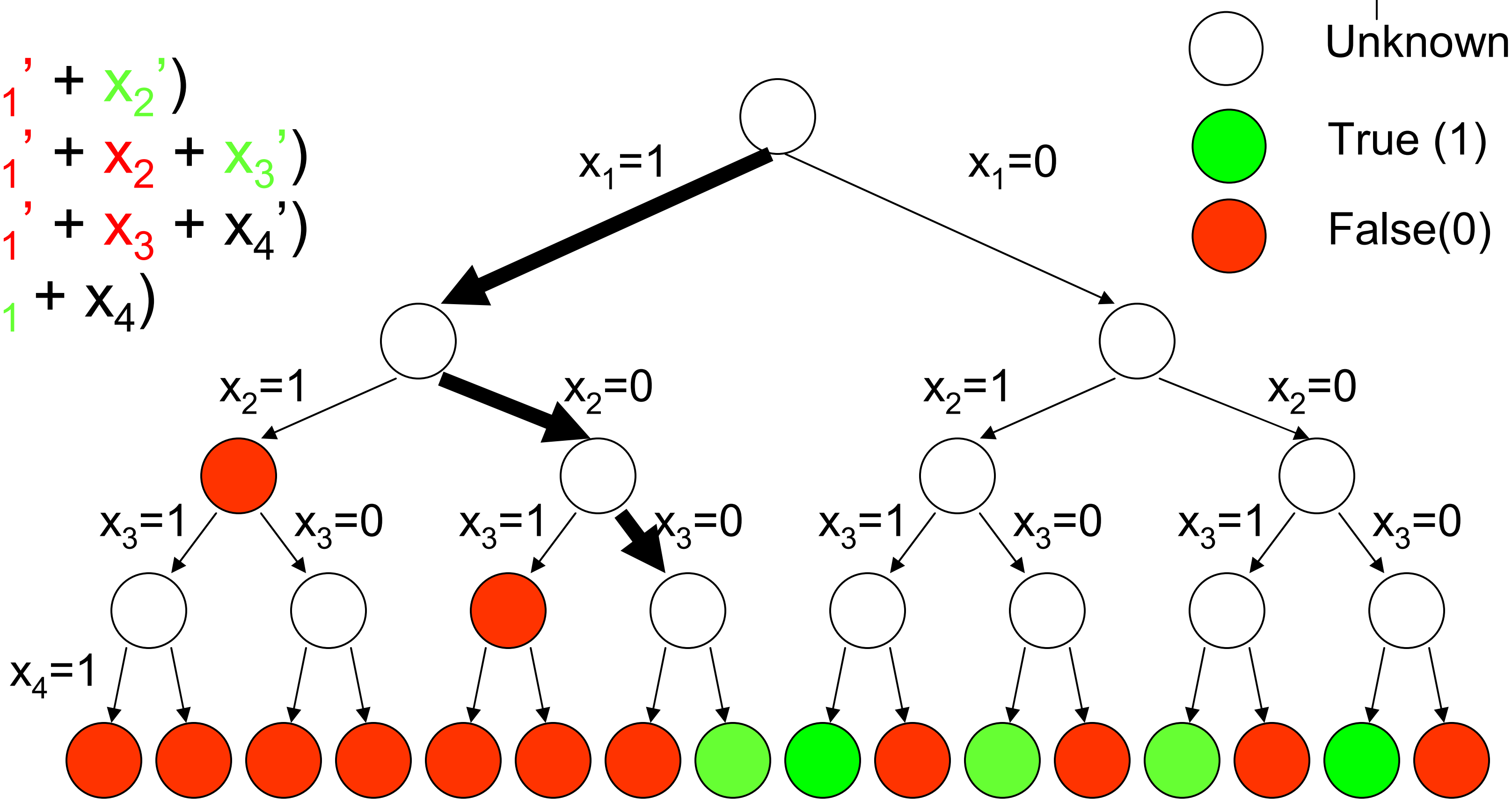
Search Tree of SAT Problem

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



Search Tree of SAT Problem

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



DLL Algorithm

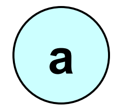
M. Davis, G. Logemann and D. Loveland, “A Machine Program for Theorem-Proving", *Communications of ACM*, Vol. 5, No. 7, pp. 394-397, 1962

- Basic framework for many modern SAT solvers
- Also known as DPLL for historical reasons

Basic DLL Procedure - DFS

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

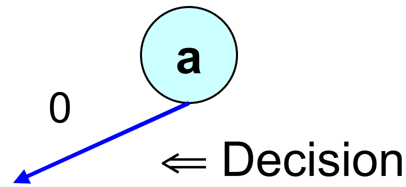
Basic DLL Procedure - DFS



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

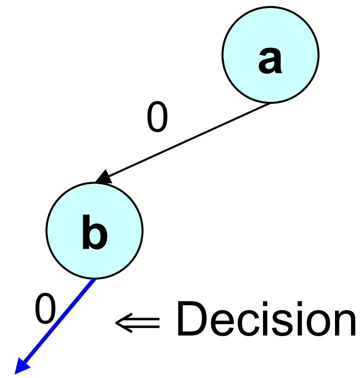
Basic DLL Procedure - DFS

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



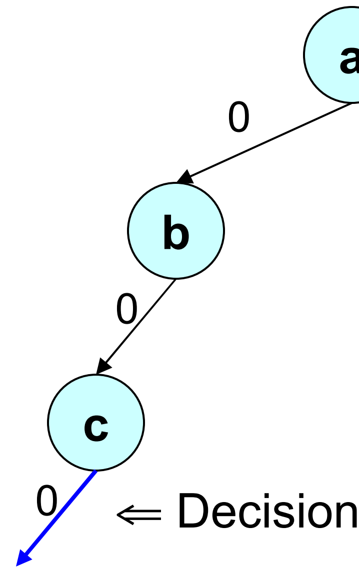
Basic DLL Procedure - DFS

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



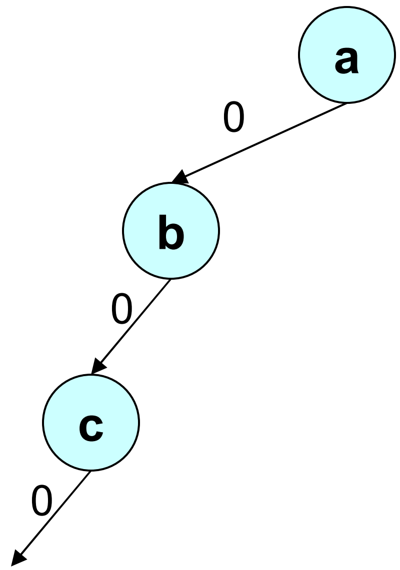
Basic DLL Procedure - DFS

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

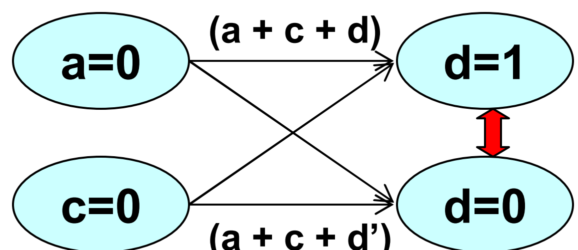


Basic DLL Procedure - DFS

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



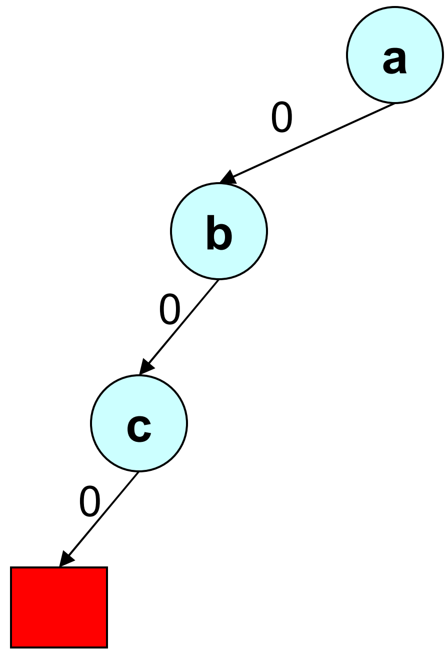
Implication Graph



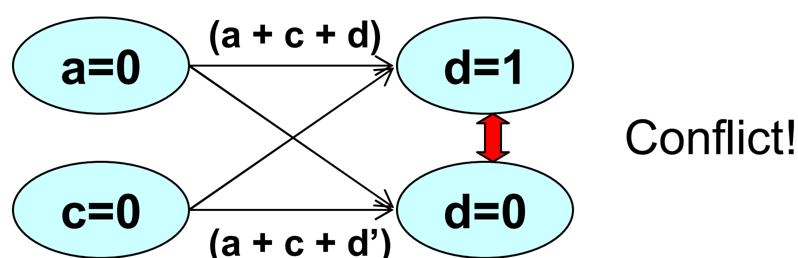
Conflict!

Basic DLL Procedure - DFS

(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)

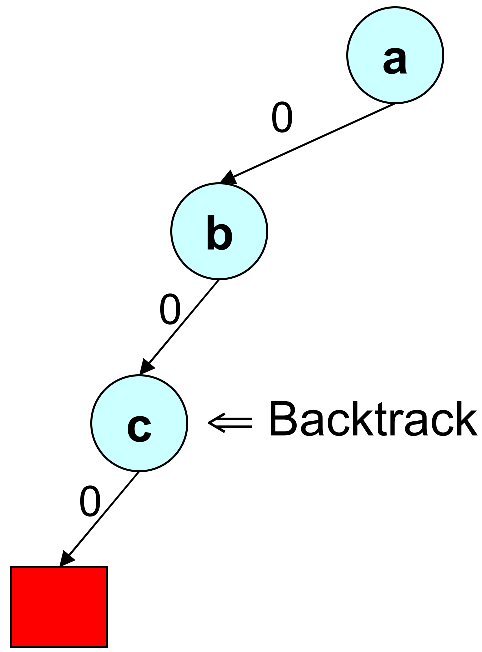


Implication Graph



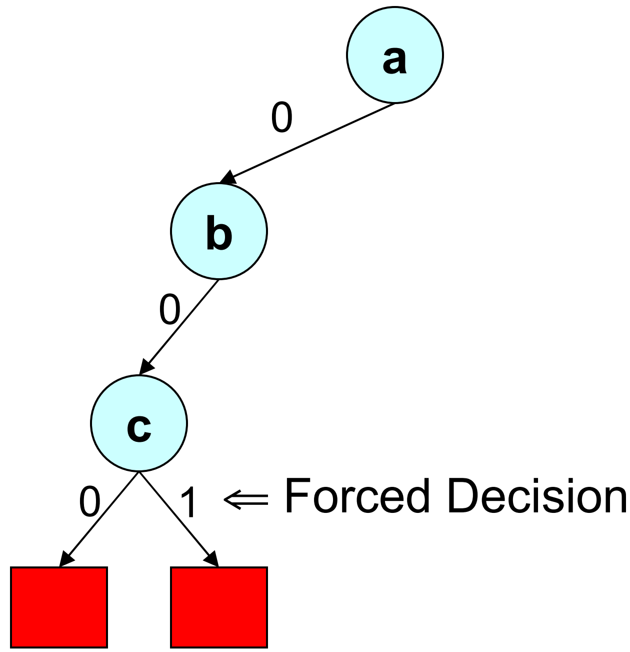
Basic DLL Procedure - DFS

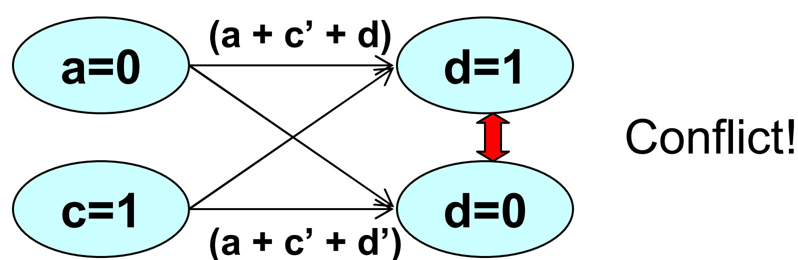
(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)



Basic DLL Procedure - DFS

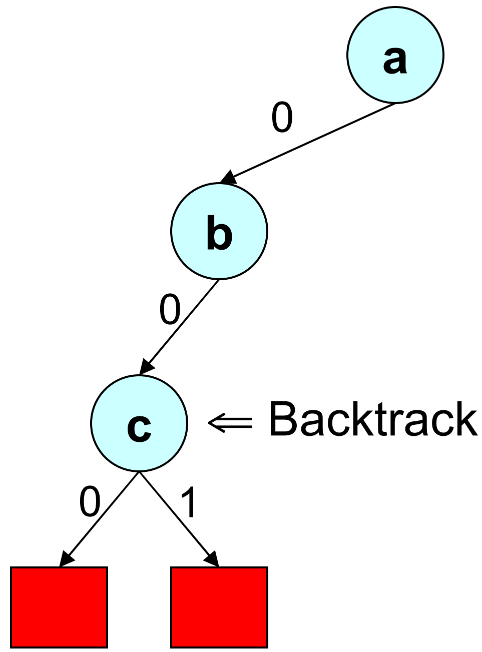
(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)





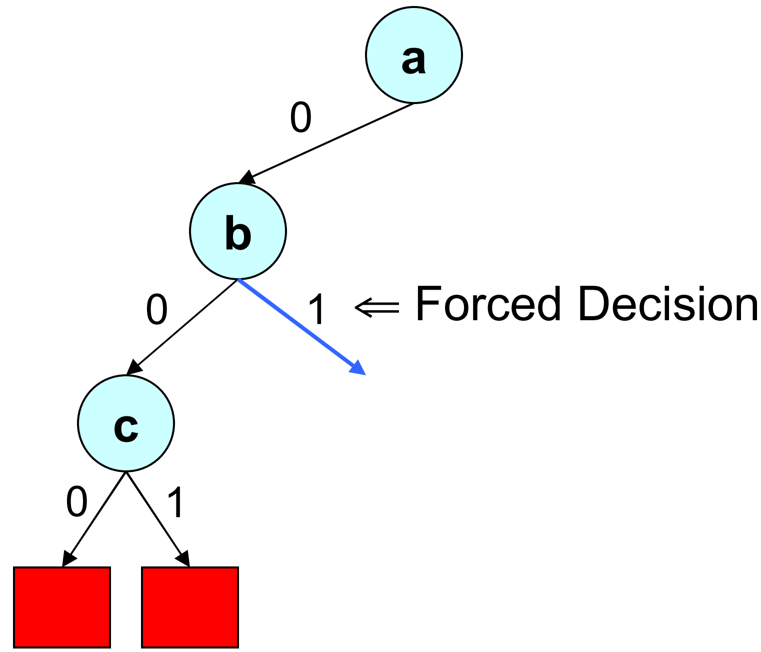
Basic DLL Procedure - DFS

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



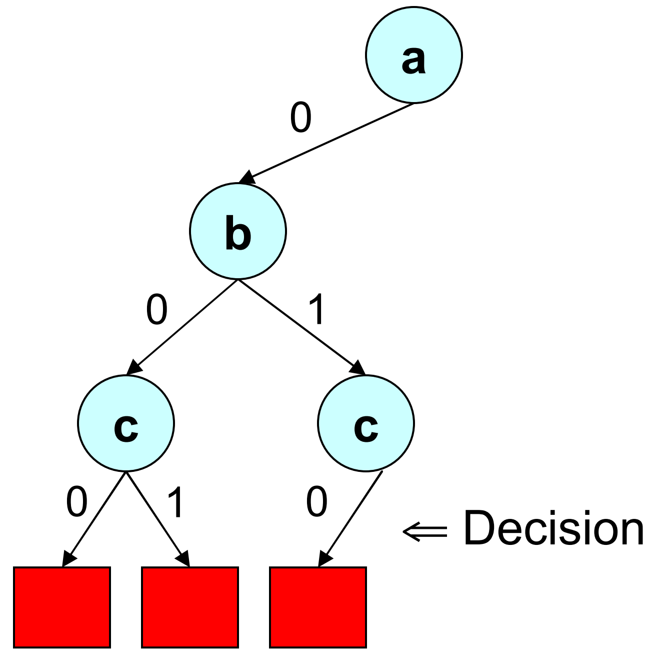
Basic DLL Procedure - DFS

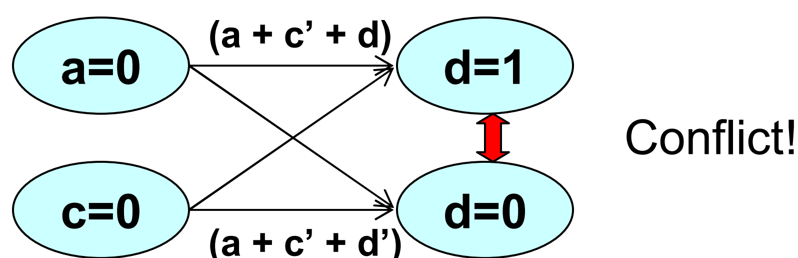
(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)



Basic DLL Procedure - DFS

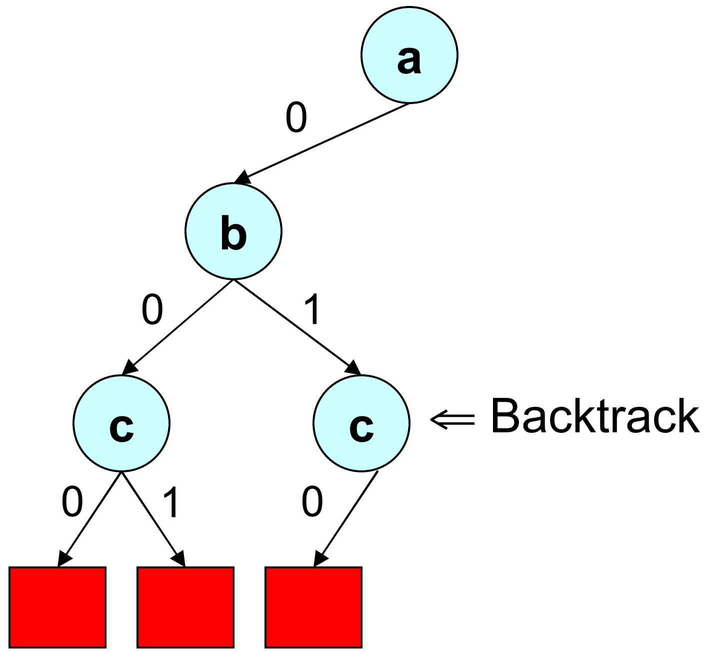
(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)





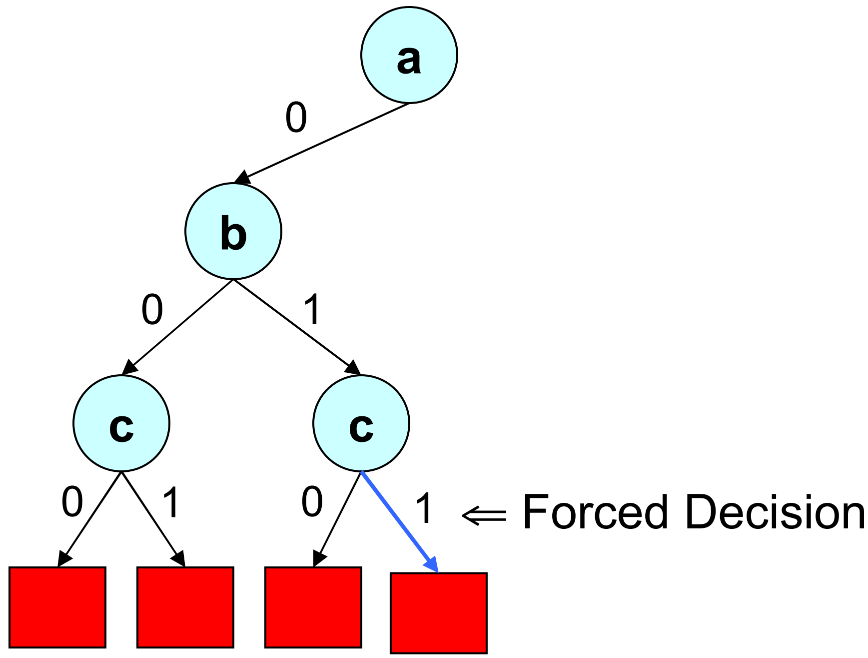
Basic DLL Procedure - DFS

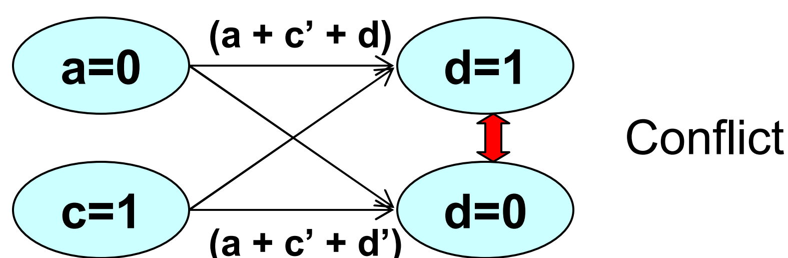
(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)



Basic DLL Procedure - DFS

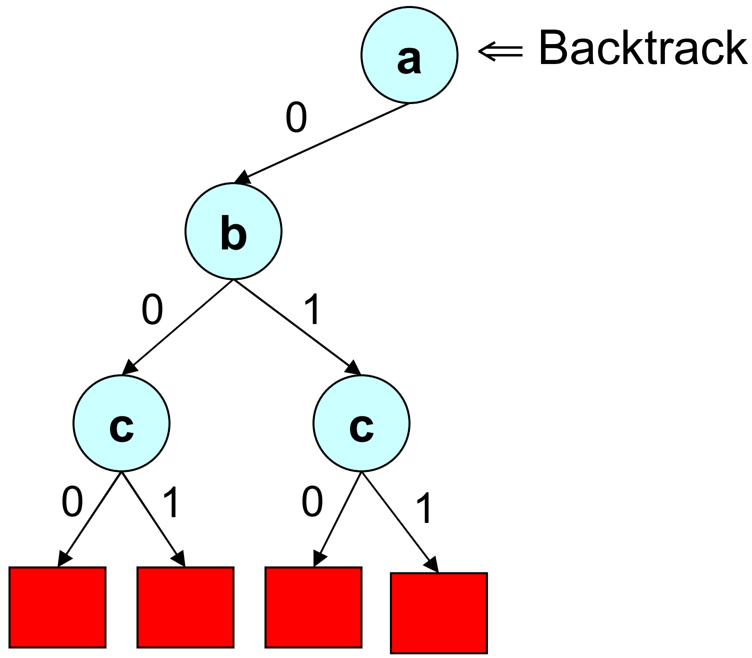
(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)





Basic DLL Procedure - DFS

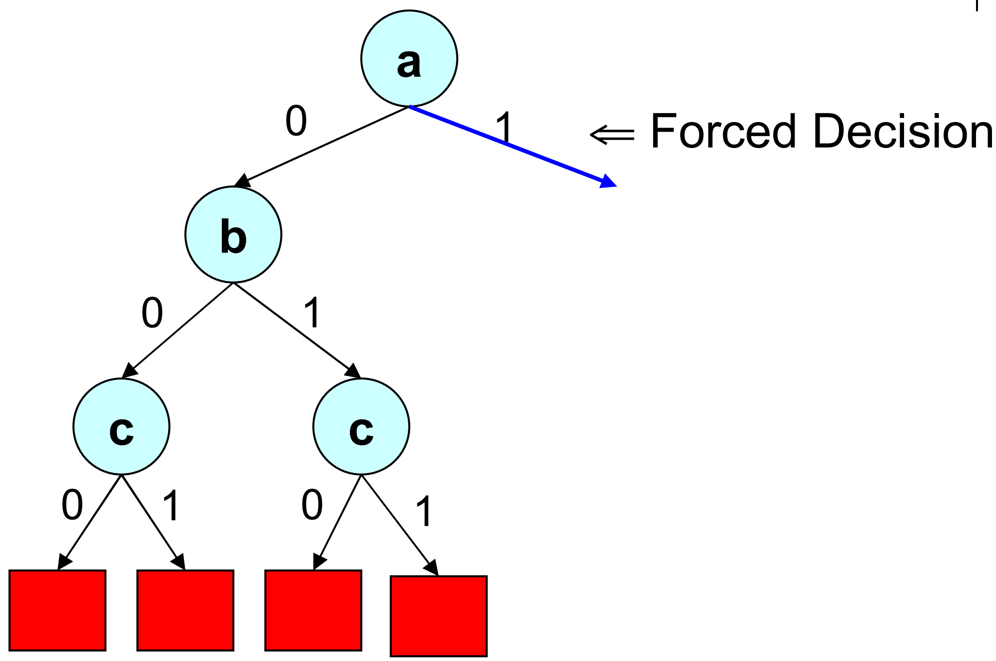
(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)



Lintao Zhang

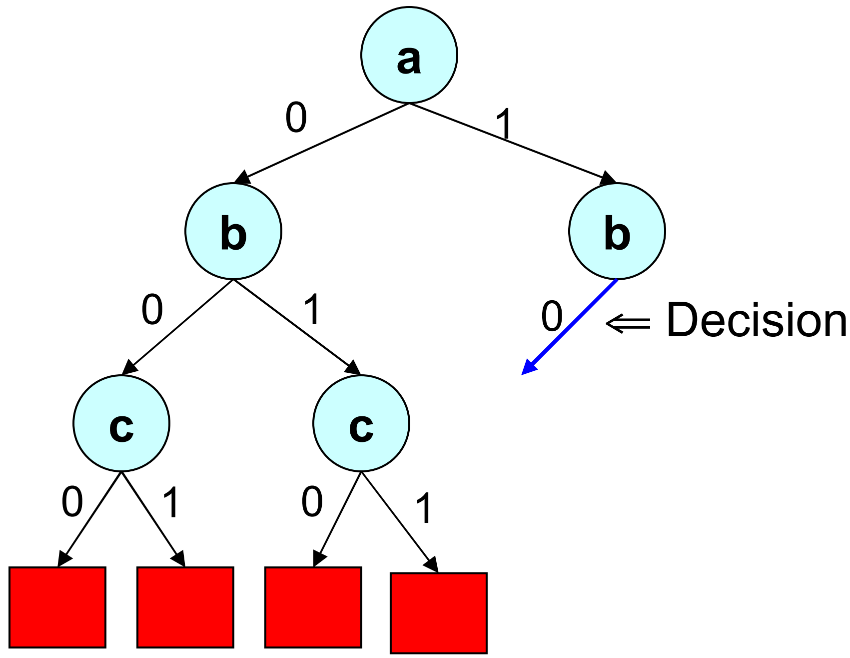
Basic DLL Procedure - DFS

(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 Procedure - DFS

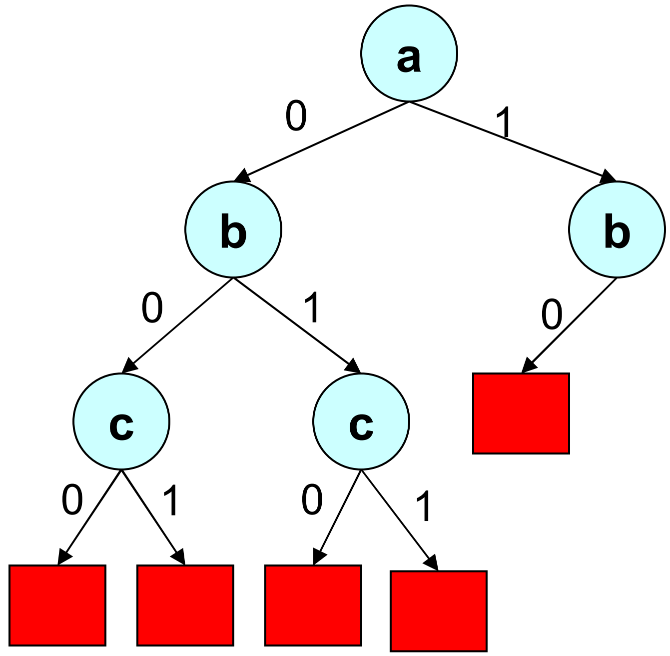
(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

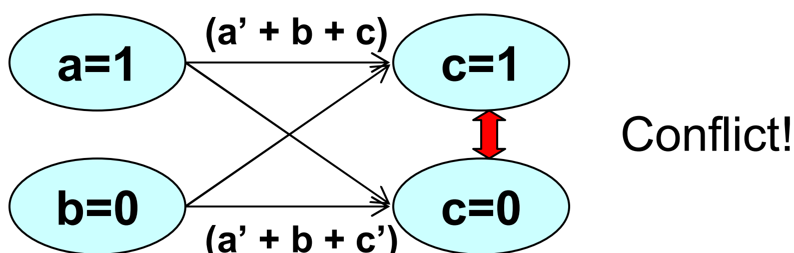


Microsoft Research

Basic DLL Procedure - DFS

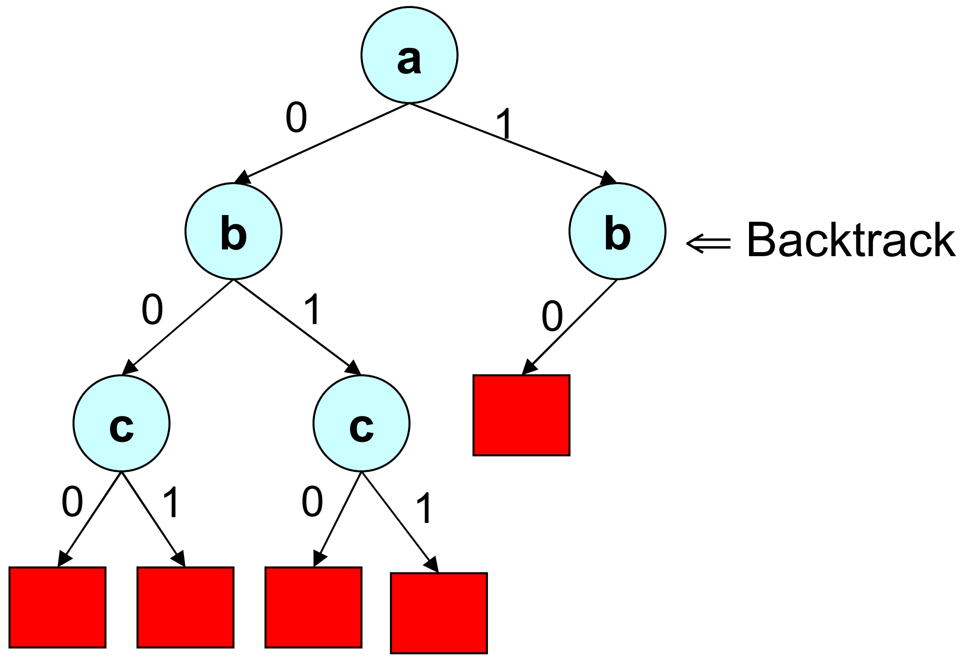
(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)





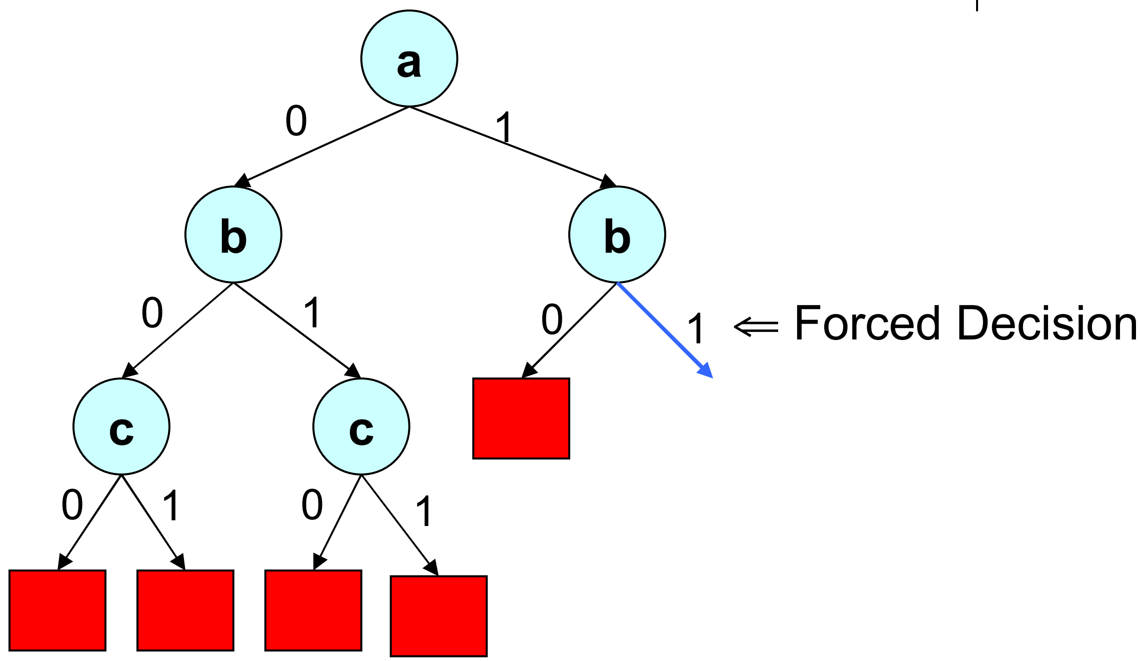
Basic DLL Procedure - DFS

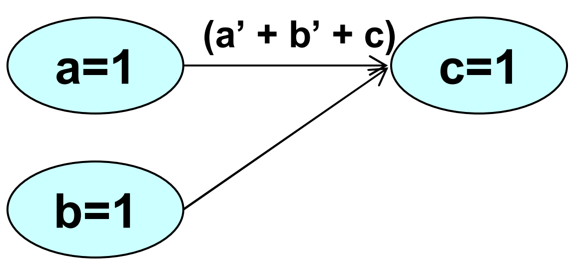
(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)



Basic DLL Procedure - DFS

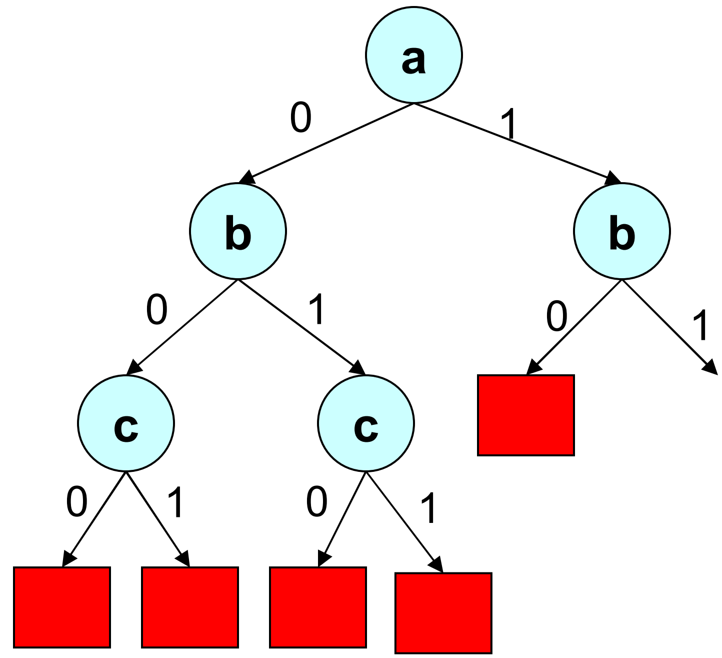
(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

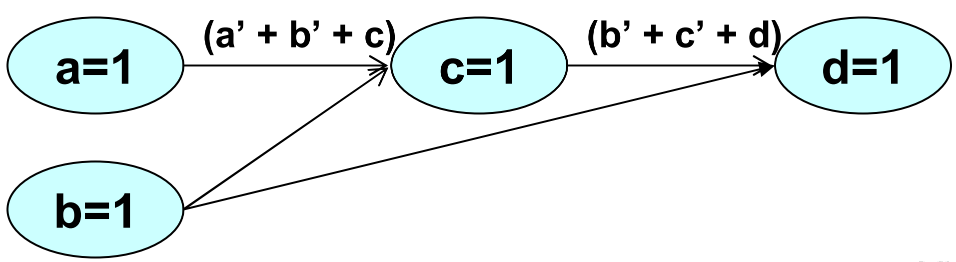




Basic DLL Procedure - DFS

(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)

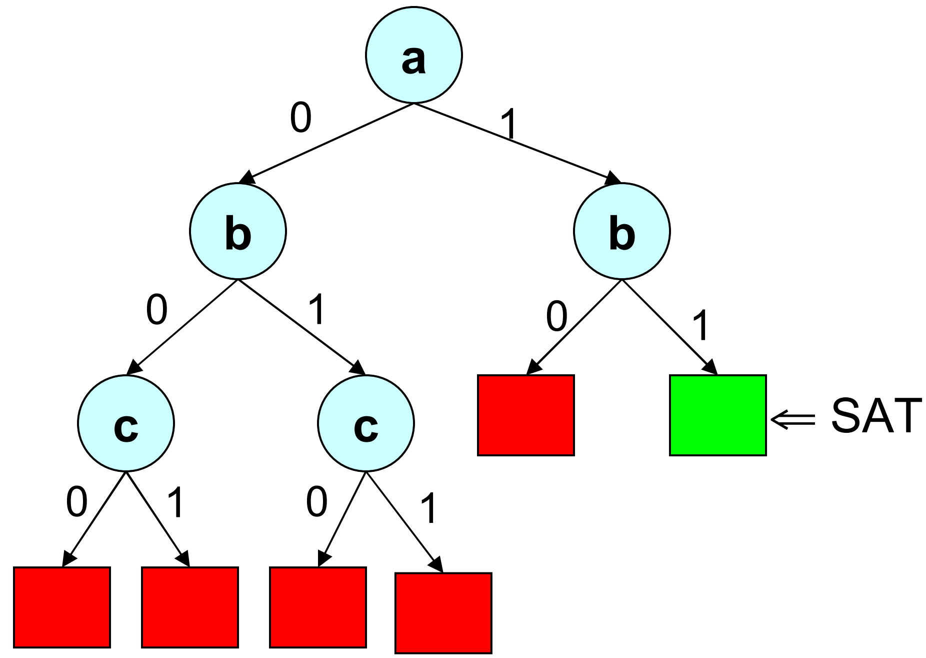


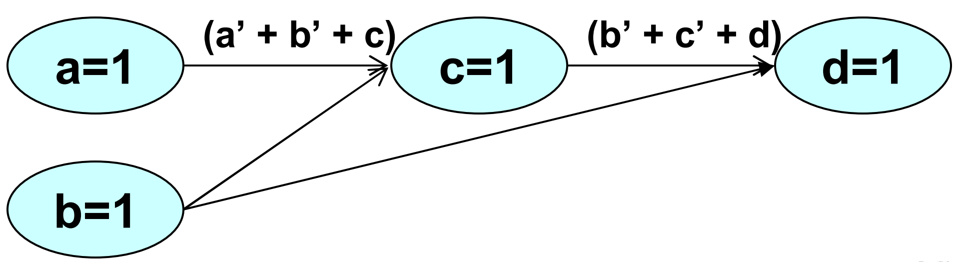


Lintao Zhang

Basic DLL Procedure - DFS

(a’ + b + c)
(a + c + d)
(a + c + d’)
(a + c’ + d)
(a + c’ + d’)
(b’ + c’ + d)
(a’ + b + c’)
(a’ + b’ + c)





Implications and Boolean Constraint Propagation

- Implication
  - A variable is forced to be assigned to be True or False based on previous assignments.
- Unit clause rule (rule for elimination of one literal clauses)
  - An unsatisfied clause is a unit clause if it has exactly one unassigned literal.

$$(a + b' + c)(b + c')(a' + c')$$

a = T, b = T, c is unassigned

Satisfied Literal

Unsatisfied Literal

Unassigned Literal

  - The unassigned literal is implied because of the unit clause.
- Boolean Constraint Propagation (BCP)
  - Iteratively apply the unit clause rule until there is no unit clause available.
- Workhorse of DLL based algorithms.

Features of DLL

- Eliminates the exponential memory requirements of DP
- Exponential time is still a problem
- Limited practical applicability – largest use seen in automatic theorem proving
- The original DLL algorithm has seen a lot of success for solving random generated instances.

Some Notes

- There are another rules proposed by the original DLL paper, which is seldom used in practice
  - Pure literal rule: if a variable only occur in one phase in the clause database, then the literal can be simply assigned with the value *true*
- The original DP paper also included the unit implication rule to simplify the clauses generated from resolution
  - Still may result in memory explosion
- DLL and DP algorithms are tightly related
  - Fundamentally, both are based on the resolution operation