Satisfiability Algorithms

- Local search (incomplete)
  - GSAT [Selman,Levesque,Mitchell 92]
  - Walksat [Kautz,Selman 96]

- Backtracking search (complete)
  - DPLL [Davis,Putnam 60] [Davis,Logeman,Loveland 62]
  - DPLL + “clause learning” GRASP, SATO, zchaff

CNF Satisfiability

$\mathbf{F} = (\mathbf{x_1} \vee \bar{\mathbf{x}}_2 \vee \mathbf{x_4}) \wedge (\bar{\mathbf{x}}_1 \vee \mathbf{x_3}) \wedge (\bar{\mathbf{x}}_3 \vee \mathbf{x_2}) \wedge (\bar{\mathbf{x}}_4 \vee \bar{\mathbf{x}}_3)$

satisfying assignment for **F**

$$\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \bar{\mathbf{x}}_4$$

Simplify(**F**, ℓ) for $\ell = \mathbf{x_3}$

$(\mathbf{x_1} \vee \bar{\mathbf{x}}_2 \vee \mathbf{x_4}) \wedge (\bar{\mathbf{x}}_1 \vee \mathbf{x_3}) \wedge (\bar{\mathbf{x}}_3 \vee \mathbf{x_2}) \wedge (\bar{\mathbf{x}}_4 \vee \bar{\mathbf{x}}_3)$

$(\mathbf{x_1} \vee \bar{\mathbf{x}}_2 \vee \mathbf{x_4}) \wedge \qquad \mathbf{x_2} \wedge \bar{\mathbf{x}}_4$

F is satisfied if all clauses disappear under simplification by the assignment

Backtracking search/DPLL

Repeat

Select a literal ℓ (some **x** or ¬**x**) — **Free step**

F ← Simplify(**F**, ℓ)

Unit propagation:

While **F** contains a 1-clause ℓ'

F ← Simplify(**F**, ℓ ')

If all clauses removed **return SAT**

If there is a 0-clause

Backtrack to last free step
and flip assignment

Recursive view of DPLL Algorithm (w/o unit propagation)

DPLL(F)

if F is empty report satisfiable and halt

if F contains the empty clause Λ

return

else choose a literal x

DPLL(Simplify(F,x))

DPLL(Simplify(F,¬x))

With unit propogation x is 1-clause

Remove all clauses containing x

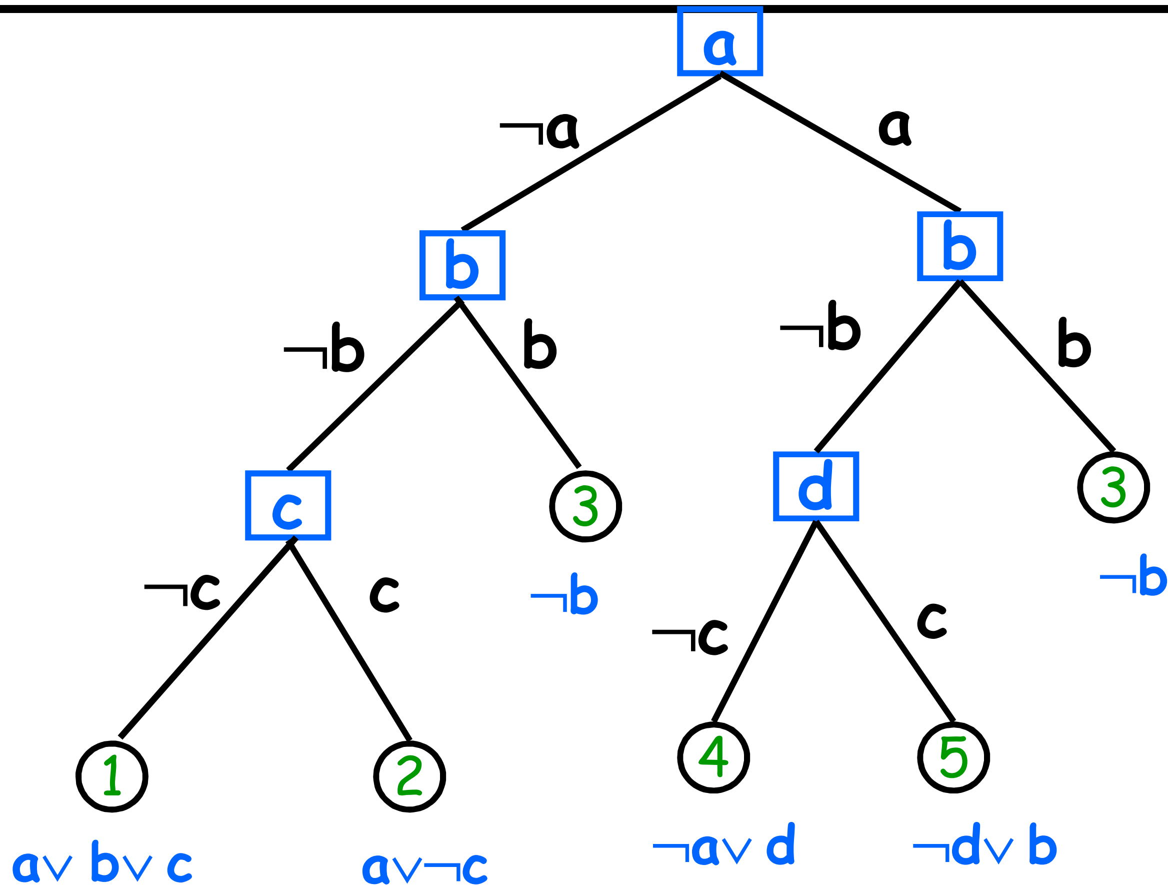
Shrink all clauses containing ¬x

DPLL on unsat formula

Clauses

1. a∨ b∨ c
2. a∨¬c
3. ¬b
4. ¬a∨ d
5. ¬d∨ b

Residual
Formula



Extending DPLL: Clause Learning

- When backtracking in DPLL, **add new clauses** corresponding to causes of failure of the search

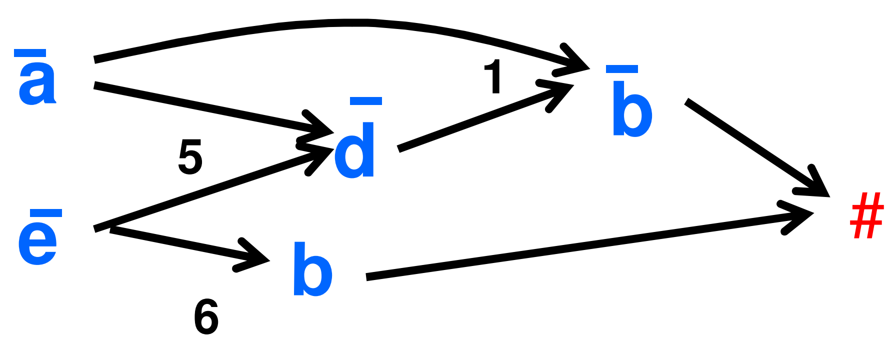
- Added conflict clauses
  - Capture *reasons* of conflicts
  - Obtained via *unit propagations* from known ones
  - Reduce future search by producing conflicts sooner

Clause Learning

- At every backtrack point derive a new clause to add to **F** that can be interpreted as a “reason” for that backtrack

$(a \vee \bar{b} \vee d)$ $(\bar{a} \vee b)$ $(\bar{c} \vee \bar{b})$ $(c \vee \bar{a})$ $(a \vee \bar{d} \vee e)$ $(b \vee e)$

1 2 3 4 5 6



Learn $(a \vee e)$

Conflict Graphs

Known Clauses

$(p \vee q \vee a)$

$(\neg a \vee \neg b \vee \neg t)$

$(t \vee \neg x_1)$

$(t \vee \neg x_2)$

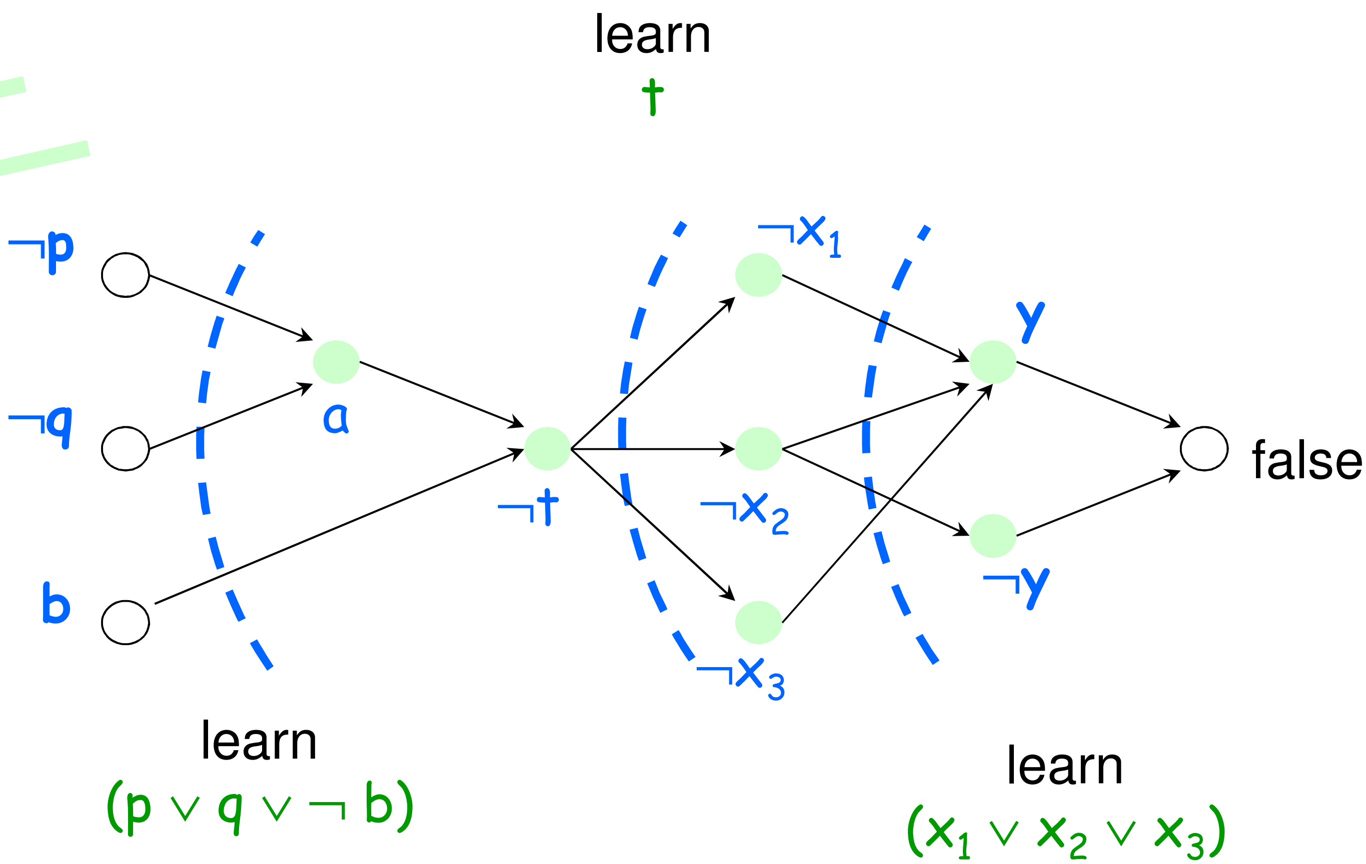
...

Current decisions

p = false

q = false

b = true



Clause Learning is Critical to Performance

- The best current SAT algorithms rely heavily on Clause Learning, e.g.

  Minisat, Glucose, MapleSAT, CaDiCaL

- Gives orders of magnitude improvement on real-world problems!