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

$F = (x_1 \lor \overline{x}_2 \lor x_4) \land (\overline{x}_1 \lor x_3) \land (\overline{x}_3 \lor x_2) \land (\overline{x}_4 \lor \overline{x}_3)$

satisfying assignment for **F**

$x_1, x_2, x_3, \overline{x}_4$

Simplify(**F**, ℓ) for $\ell = x_3$

$(x_1 \lor \overline{x}_2 \lor x_4) \land (\overline{x}_1 \lor x_3) \land (\overline{x}_3 \lor x_2) \land (\overline{x}_4 \lor \overline{x}_3)$

$(x_1 \lor \overline{x}_2 \lor x_4) \land \qquad x_2 \land \overline{x}_4$

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

Backtracking search/DPLL

```
Repeat
    Select a literal $\ell$ (some $x$ or $\neg x$)        } Free step
          F ← Simplify(F, $\ell$)
Unit          { While F contains a 1-clause $\ell'$
propagation   {       F ← Simplify(F, $\ell'$)
          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 propogation)

```
DPLL(F)
    if F is empty report satisfiable and halt
    if F contains the empty clause $\Lambda$
          return
    else choose a literal $x$
          DPLL(Simplify(F,$x$))
          DPLL(Simplify(F,$\neg x$))
```

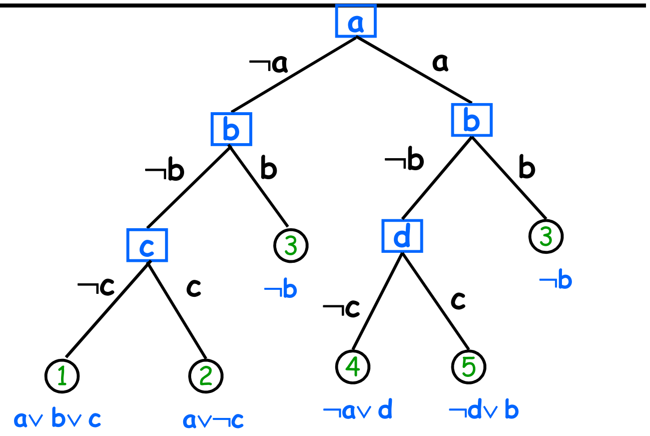
With unit propogation x is 1-clause

Remove all clauses containing x
Shrink all clauses containing $\neg x$

DPLL on unsat formula

Clauses
1.
2.
3.
4.
5.

Residual Formula



Extending DPLL: Clause Learning

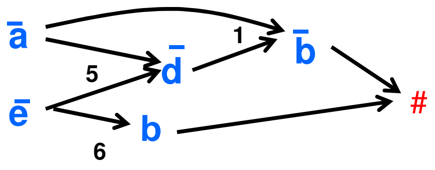
- When backtracking in DPLL, **add new clauses** corresponding to causes of failure of the search

- 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

$(a \lor \bar{b} \lor d)$	$(\bar{a} \lor b)$	$(\bar{c} \lor \bar{b})$	$(c \lor \bar{a})$	$(a \lor \bar{d} \lor e)$	$(b \lor e)$
1	2	3	4	5	6

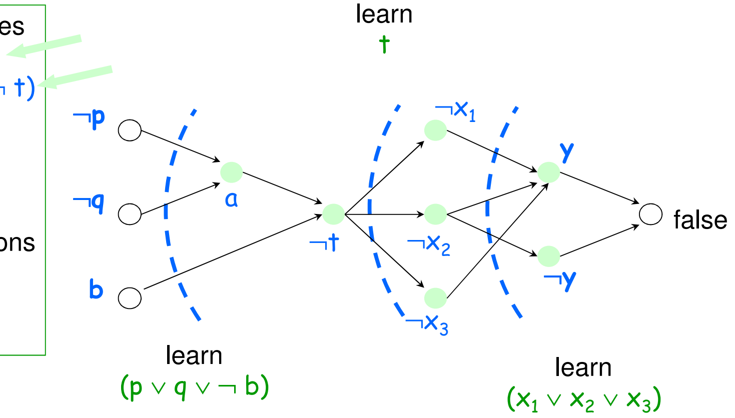


Learn $(a \lor e)$

Conflict Graphs

Known Clauses
$(p \lor q \lor a)$
$(\neg a \lor \neg b \lor \neg t)$
$(t \lor \neg x_1)$
$(t \lor \neg x_2)$
...

Current decisions
p = false
q = false
b = true



learn
t

learn
$(p \lor q \lor \neg b)$

learn
$(x_1 \lor x_2 \lor x_3)$

Clause Learning is Critical to Performance

- The best current SAT algorithms rely heavily on Clause Learning, e.g.
  - zChaff, berkmin, minisat
- Gives orders of magnitude improvement on real-world problems!