SAT Solver Heuristics

SAT-solver History

- Started with David-Putnam-Logemann-Loveland (DPLL) (1962)
  - Able to solve 10-15 variable problems
- Satz (Chu Min Li, 1995)
  - Able to solve some 1000 variable problems
- Chaff (Malik et al., 2001)
  - Intelligently hacked DPLL , Won the 2004 competition
  - Able to solve some 10000 variable problems
- Current state-of-the-art
  - MiniSAT and SATELITEGTI (Chalmer's university, 2004-2006)
  - Jerusat and Haifasat (Intel Haifa, 2002)
  - Ace (UCLA, 2004-2006)

MiniSAT

- MiniSat is a fast SAT solver developed by Niklas Eén and Niklas Sörensson
  - MiniSat won all industrial categories in SAT 2005 competition
  - MiniSat won SAT-Race 2006

- MiniSat is simple and well-documented
  - Well-defined interface for general use
  - Helpful implementation documents and comments
  - Minimal but efficient heuristic
    - Main.C (344 lines)
    - Solver.C (741 lines)

Overview (1/2)

- A set of propositional variables and CNF clauses involving variables
  - $(x_1 \vee x_1' \vee x_3) \wedge (x_2 \vee x_1' \vee x_4)$
  - x_1, x_2, x_3 and x_4 are variables (true or false)

- Literals: Variable and its negation
  - x_1 and x_1'

- A clause is satisfied if one of the literals is true
  - x_1=true satisfies clause 1
  - x_1=false satisfies clause 2

- Solution: An assignment that satisfies all clauses

Overview (2/2)

- Unit clause is a clause in which all but one of literals is assigned to False
- Unit literal is the unassigned literal in a unit clause

$$
\begin{array}{l}
\ldots\ldots \\
(x_0) \wedge \\
(-x_0 \vee x_1) \wedge \\
(-x_2 \vee -x_3 \vee -x_4) \wedge \\
\ldots\ldots
\end{array}
$$

  - (x_0) is a unit clause and 'x_0' is a unit literal
  - ($-x_0 \vee x_1$) is a unit clause since x_0 has to be True
  - ($-x_2 \vee -x_3 \vee -x_4$) can be a unit clause if the current assignment is that x_3 and x_4 are True
- Boolean Constrain Propagation(BCP) is the process of assigning the True value to all unit literals

DPLL Overview (1/3)

```
/* The Quest for Efficient Boolean Satisfiability Solvers
 *  by L.Zhang and S.Malik, Computer Aided Verification 2002 */
DPLL(a formula ' , assignment) {
     necessary = deduction(' , assignment);
     new_asgnment = union(necessary, assignment);
     if (is_satisfied(' , new_asgnment))
          return SATISFIABLE;
     else if (is_conflicting(' , new_asgnmnt))
          return UNSATISFIABLE;
     var = choose_free_variable(' , new_asgnmnt);
     asgn1 = union(new_asgnmnt, assign(var, 1));
     if (DPLL(' , asgn1) == SATISFIABLE)
          return SATISFIABLE;
     else {
          asgn2 = union (new_asgnmnt, assign(var,0));
          return DPLL (' , asgn2);
     }
}
```

Three techniques added to modern SAT solvers

1. Learnt clauses
2. Non-chronological backtracking
3. Restart

DPLL Overview (2/3)

$\{p \vee r\} \wedge \{\neg p \vee \neg q \vee r\} \wedge \{p \vee \neg r\}$

p=T

p=F

$\{T \vee r\} \wedge \{\neg T \vee \neg q \vee r\} \wedge \{T \vee \neg r\}$

$\{F \vee r\} \wedge \{\neg F \vee \neg q \vee r\} \wedge \{F \vee \neg r\}$

SIMPLIFY

$\{\neg q , r\}$

SIMPLIFY

$\{r\} \wedge \{\neg r\}$

SIMPLIFY

$\{\}$

DPLL Overview (3/3)

```
/* overall structure of Minisat solve procedure */
Solve(){
    while(true){
        boolean_constraint_propagation();
        if(no_conflict){
            if(no_unassigned_variable) return SAT;
            make_decision();
        }else{
            if (no_decisions_made) return UNSAT;
            analyze_conflict();
            undo_assignments();
            add_conflict_clause();
        }
    }
}
```

Conflict Clause Analysis (1/10)

- A conflict happens when one clause is falsified by unit propagation

Assume x_4 is False
$(x_1 \vee x_4) \wedge$
$(-x_1 \vee x_2) \wedge$
$(-x_2 \vee x_3) \wedge$
$(-x_3 \vee -x_2 \vee -x_1)$ Falsified!
Omitted clauses

- Analyze the conflicting clause to infer a clause
  - $(-x_3 \vee -x_2 \vee -x_1)$ is conflicting clause
- The inferred clause is a new knowledge
  - A new learnt clause is added to constraints

Conflict Clause Analysis (2/10)

- Learnt clauses are inferred by conflict analysis

$(x_1 \vee x_4)$ ∧
$(-x_1 \vee x_2)$ ∧
$(-x_2 \vee x_3)$ ∧
$(-x_3 \vee -x_2 \vee -x_1)$ ∧
omitted clauses ∧
(x_4) learnt clause

- They help prune future parts of the search space
  - Assigning False to x_4 is the casual of conflict
  - Adding (x_4) to constraints prohibit conflict from $–x_4$
- Learnt clauses actually drive backtracking

Conflict Clause Analysis (3/10)

```
/* conflict analysis algorithm */
Analyze_conflict(){
    cl = find_conflicting_clause();
    /* Loop until cl is falsified and one literal whose value is determined in current
    decision level is remained */
    While(!stop_criterion_met(cl)){
        lit = choose_literal(cl); /* select the last propagated literal */
        Var = variable_of_literal(lit);
        ante = antecedent(var);
        cl = resolve(cl, ante, var);
    }
    add_clause_to_database(cl);
    /* backtrack level is the lowest decision level for which the learnt clause is unit
    clause */
    back_dl = clause_asserting_level(cl);
    return back_dl;
}
```

Algorithm from Lintao Zhang and Sharad malik
"The Quest for Efficient Boolean Satisfiability Solvers"

Conflict Clause Analysis (4/10)

- Example of conflict clause analysis
  - a, b, c, d, e, f, g, and h: 8 variables (2^8 cases)

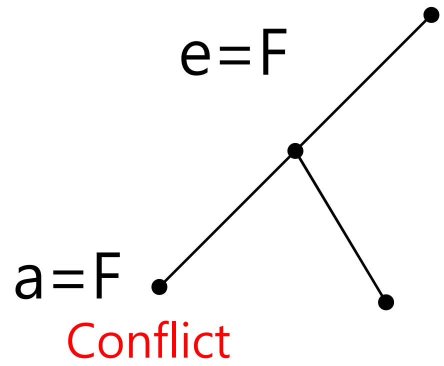
(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g)
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d)

Satisfiable?

Unsatisfiable?

Conflict Clause Analysis (5/10)

Assignments	antecedent
e=F	assumption
f=F	-f∨e
g=F DLevel=1	-g∨f
h=F	-h∨g
a=F	assumption
b=T DLevel=2	b∨a∨e
c=T	c∨e∨f∨-b
d=T	d∨-b∨h

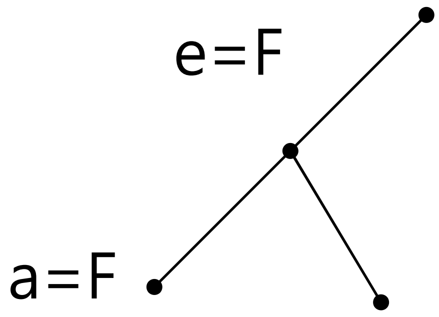


-b∨-c∨-d

Example slides are from CMU 15-414 course ppt

Conflict Clause Analysis (6/10)

Assignments	antecedent	
e=F	assumption	
f=F	-f∨e	DLevel=1
g=F	-g∨f	
h=F	-h∨g	
a=F	assumption	
b=T	b∨a∨e	DLevel=2
c=T	c∨e∨f∨-b	
d=T	d∨-b∨h	



-b∨-c∨-d

Resolution

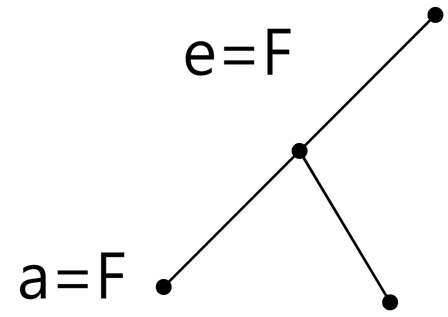
- Resolution is a process to generate a clause from two clauses
- Given two clauses $(x \vee y)$ and $(-y \vee z)$, the resolvent of these two clauses is $(x \vee z)$
  - $(x \vee y) \wedge (-y \vee z)$ is satisfiable iff $(x \vee y) \wedge (-y \vee z) \wedge (x \vee z)$ is satisfiable
  - The resolvent is redundant

Conflict Clause Analysis (7/10)

Assignments	antecedent
e=F	assumption
f=F	-f∨e
g=F	-g∨f
h=F	-h∨g
a=F	assumption
b=T	b∨a∨e
c=T	c∨e∨f∨-b
d=T	d∨-b∨h

DLevel=1 (e=F through h=F)

DLevel=2 (a=F through d=T)



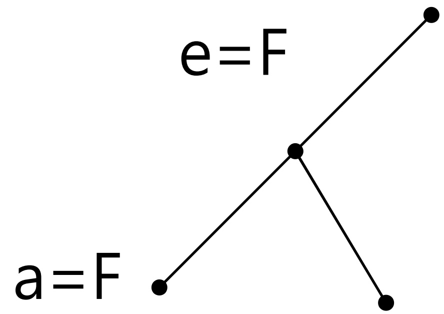
-b∨-c∨h
(a resolvent of
-b∨-c∨-d
and d∨-b∨h)

Conflict Clause Analysis (8/10)

Assignments	antecedent
e=F	assumption
f=F	-f∨e
g=F	-g∨f
h=F	-h∨g
a=F	assumption
b=T	b∨a∨e
c=T	c∨e∨f∨-b
d=T	d∨-b∨h

DLevel=1 (e=F, f=F, g=F, h=F)

DLevel=2 (a=F, b=T, c=T, d=T)



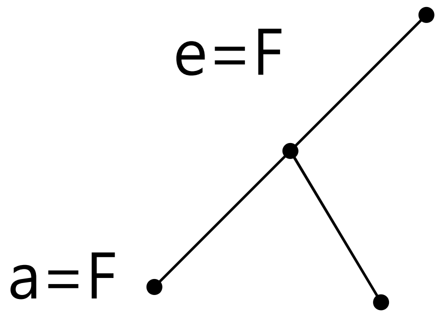
-b∨-c∨h

Conflict Clause Analysis (9/10)

Assignments	antecedent
e=F	assumption
f=F	-f∨e
g=F	-g∨f
h=F	-h∨g
a=F	assumption
b=T	b∨a∨e
c=T	c∨e∨f∨-b
d=T	d∨-b∨h

DLevel=1

DLevel=2



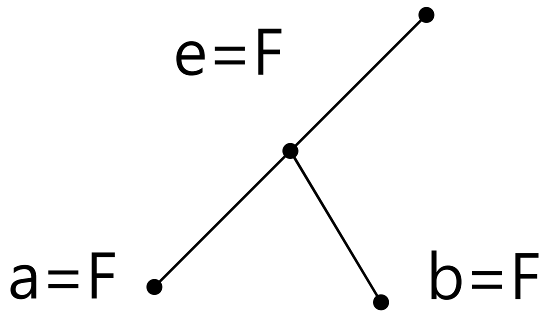
-b∨e∨f∨h learnt clause

Conflict Clause Analysis (10/10)

Assignments	antecedent
e=F	assumption
f=F	-f∨e
g=F	-g∨f
h=F	-h∨g
b=F	-b∨e∨f∨h
…	…

DLevel=1

New assignment@ level 1



-b∨-c∨-d

-b∨-c∨h

-b∨e∨f∨h

Variable State Independent Decaying Sum(VSIDS)

- Decision heuristic to determine what variable will be assigned next
- Decision is independent from the current assignment of each variable
- VSIDS makes decisions based on activity
  - Activity is a literal occurrence count with higher weight on the more recently added clauses
  - MiniSAT does not consider any polarity in VSIDS
    - Each variable, not literal has score

activity description from Lintao Zhang and Sharad malik
“The Quest for Efficient Boolean Satisfiability Solvers”

VSIDS Decision Heuristic – MiniSAT style (1/8)

- Initially, the score for each variable is 0
- First make a decision e = False
  - The order between same score is unspecified.
  - MiniSAT always assigns False to variables.

Initial constraints
(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g) ∧
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d)

Variable	Score	Value
a	0	
b	0	
c	0	
d	0	
e	0	F
f	0	
g	0	
h	0	

VSIDS Decision Heuristic (2/8)

- f, g, h are False after BCP

(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g) ∧
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d)

Variable	Score	Value
a	0	
b	0	
c	0	
d	0	
e	0	F
f	0	F
g	0	F
h	0	F

VSIDS Decision Heuristic (3/8)

- a is next decision variable

(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g) ∧
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d)

Variable	Score	Value
a	**0**	**F**
b	**0**	
c	**0**	
d	**0**	
e	**0**	**F**
f	**0**	**F**
g	**0**	**F**
h	**0**	**F**

VSIDS Decision Heuristic (4/8)

- b, c are True after BCP
- Conflict occurs on variable d
  - Start conflict analysis

(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g) ∧
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d)

Variable	Score	Value
a	0	F
b	0	T
c	0	T
d	0	T
e	0	F
f	0	F
g	0	F
h	0	F

VSIDS Decision Heuristic (5/8)

- The score of variable in resolvents is increased by 1
  - Even if a variable appears in resolvents two or mores increase the score just once

(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g) ∧
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d)

Resolvent on d
-b∨-c∨h

Variable	Score	Value
a	0	F
b	1	T
c	1	T
d	0	T
e	0	F
f	0	F
g	0	F
h	1	F

VSIDS Decision Heuristic (6/8)

- The end of conflict analysis
- The scores are decaying 5% for next scoring

(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g) ∧
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d)

Resolvents
-b∨-c∨h
-b∨e∨f∨h ← learnt clause

Variable	Score	Value
a	**0**	**F**
b	**0.95**	**T**
c	**0.95**	**T**
d	**0**	**T**
e	**0.95**	**F**
f	**0.95**	**F**
g	**0**	**F**
h	**0.95**	**F**

VSIDS Decision Heuristic (7/8)

- b is now False and a is True after BCP
- Next decision variable is c with 0.95 score

(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g) ∧
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d) ∧

Learnt clause **(-b∨e∨f∨h)**

Variable	Score	Value
a	**0**	**T**
b	**0.95**	**F**
c	**0.95**	
d	**0**	
e	**0.95**	**F**
f	**0.95**	**F**
g	**0**	**F**
h	**0.95**	**F**

VSIDS Decision Heuristic (8/8)

- Finally we find a model!

(-f∨e) ∧
(-g∨f) ∧
(b∨a∨e) ∧
(c∨e∨f∨-b) ∧
(-h∨g) ∧
(d∨-b∨h) ∧
(-b∨-c∨-d) ∧
(c∨d) ∧
Learnt clause **(-b∨e∨f∨h)**

Variable	Score	Value
a	**0**	**T**
b	**0.95**	**F**
c	**0.95**	**F**
d	**0**	**T**
e	**0.95**	**F**
f	**0.95**	**F**
g	**0**	**F**
h	**0.95**	**F**