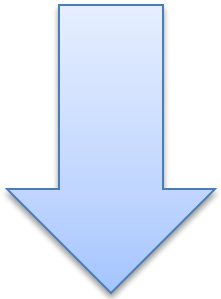
Satisfiability Modulo Theories

Bruno Dutertre
SRI International

Leonardo de Moura
Microsoft Research

Satisfiability

$$a > b + 2, \qquad a = 2c + 10, \qquad c + b \leq 1000$$



SAT

Model

$$a = 0, \qquad b = -3, \qquad c = -5$$

$$0 > -3 + 2, \qquad 0 = 2(-5) + 10, \qquad (-5) + (-3) \leq 1000$$

Satisfiability

$$b + 2 = c, \quad f(read(write(a,b,3), c-2)) \neq f(c-b+1)$$

Satisfiability

$$b + 2 = c, \quad f(read(write(a,b,3), c-2)) \neq f(c-b+1)$$

Arithmetic

Satisfiability

$$b + 2 = c, \quad f(read(write(a,b,3), c-2)) \neq f(c-b+1)$$

Array Theory

Satisfiability

$$b + 2 = c, \quad f(read(write(a,b,3), c-2)) \neq f(c-b+1)$$

Uninterpreted Functions

Satisfiability

$$b + 2 = c, \quad f(read(write(a,b,3), c-2)) \neq f(c-b+1)$$

Satisfiability

$$b + 2 = c, \quad f(read(write(a,b,3), b+2-2)) \neq f(b+2-b+1)$$

Satisfiability

$$b + 2 = c, \quad f(read(write(a,b,3), b+2-2)) \neq f(b+2-b+1)$$

Satisfiability

$$b + 2 = c, \quad f(read(write(a,b,3), b)) \neq f(b+2-b+1)$$

Satisfiability

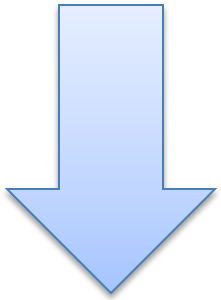
$$b + 2 = c, \quad f(read(write(a,b,3), b)) \neq f(3)$$

Array Theory Axiom

$$\forall a,i,v : read(write(a, i, v), i) = v$$

Satisfiability

$b + 2 = c, \quad f(3) \neq f(3)$

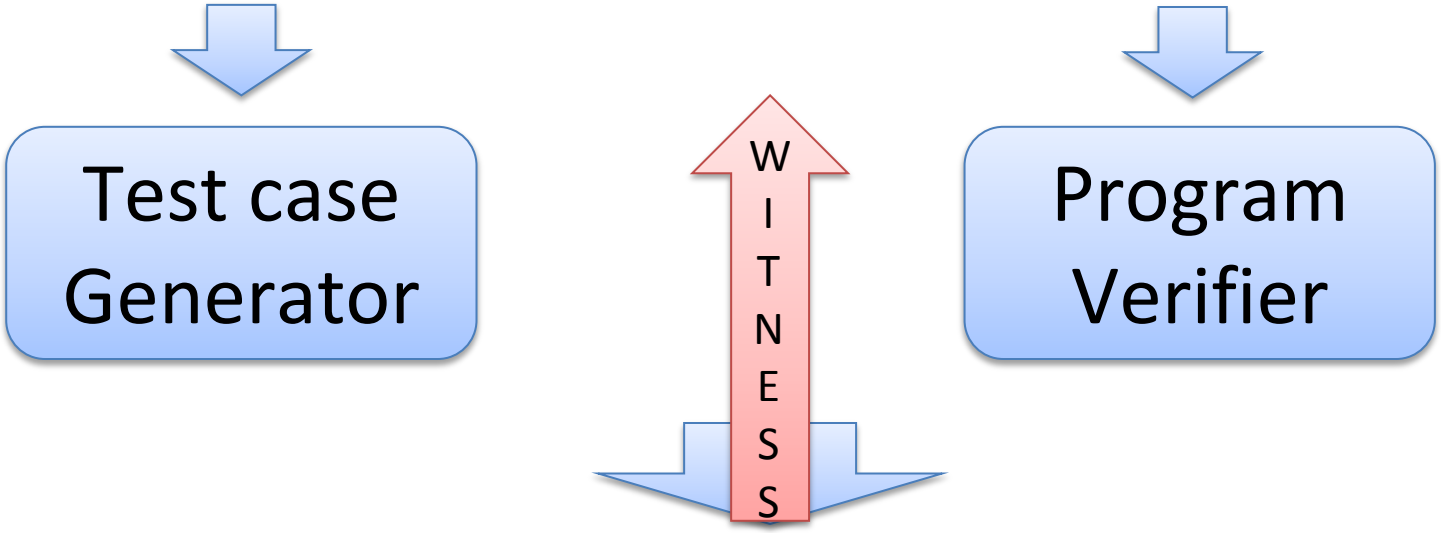


UNSAT

Applications

Is execution path *P* feasible?

Is assertion *X* violated?



Is Formula *F* Satisfiable?

Test Case Generation

```
unsigned GCD(x, y) {
  requires(y > 0);
  while (true) {
    unsigned m = x % y;
    if (m == 0) return y;
    x = y;
    y = m;
  }
}
```

SSA

$(y_0 > 0)$ and

$(m_0 = x_0 \% y_0)$ and

not $(m_0 = 0)$ and

$(x_1 = y_0)$ and

$(y_1 = m_0)$ and

$(m_1 = x_1 \% y_1)$ and

$(m_1 = 0)$

Solver

$x_0 = 2$

$y_0 = 4$

$m_0 = 2$

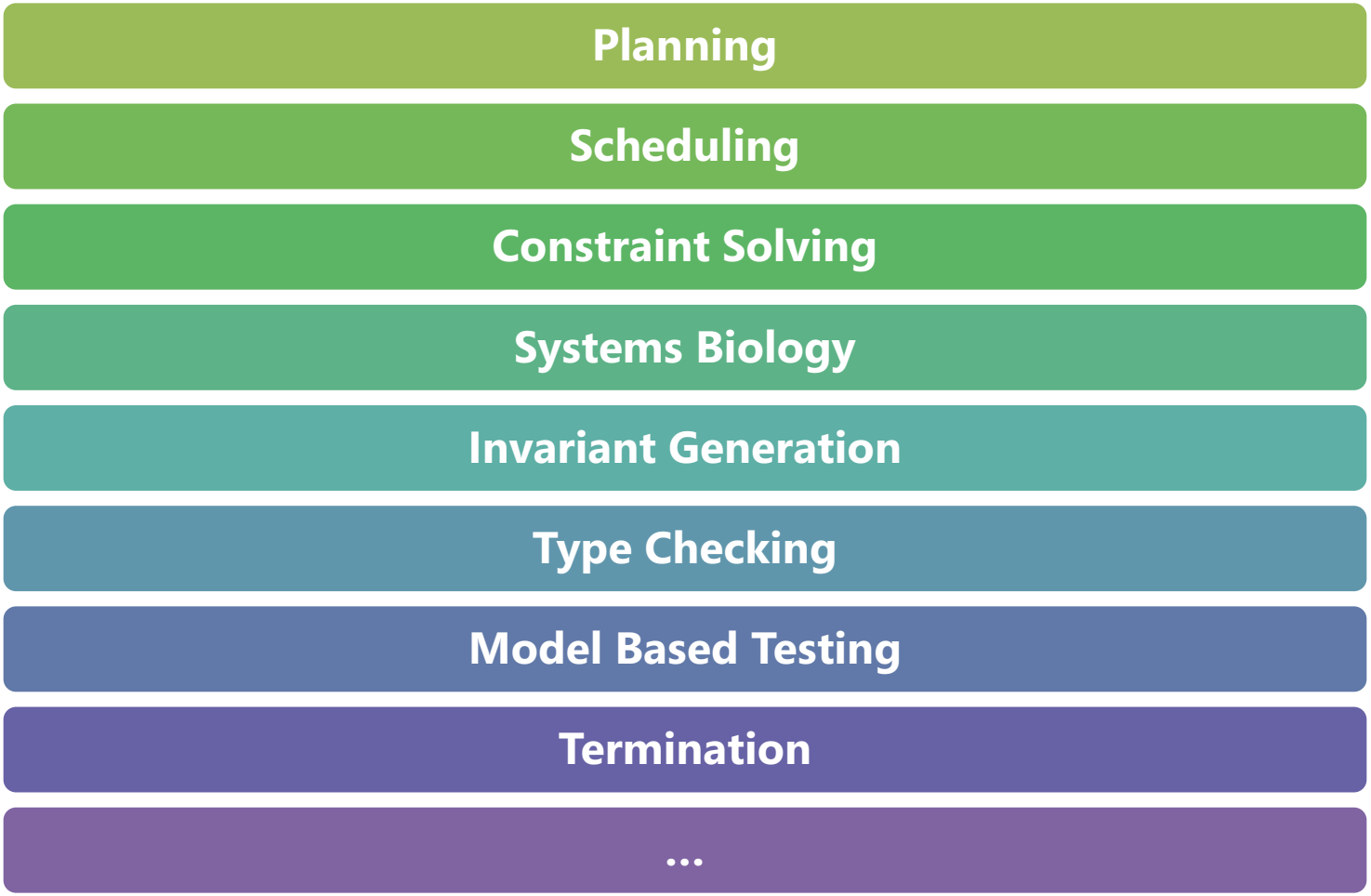
$x_1 = 4$

$y_1 = 2$

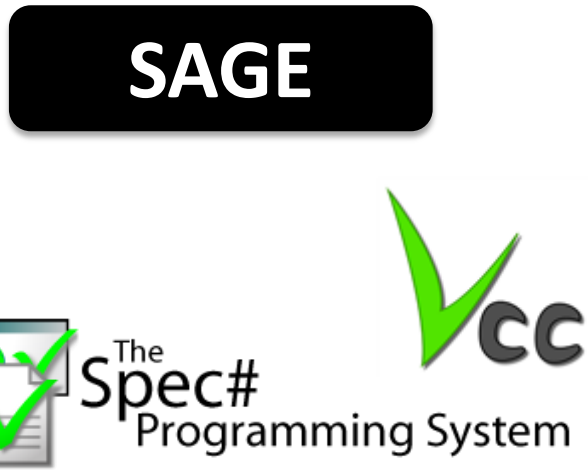
$m_1 = 0$

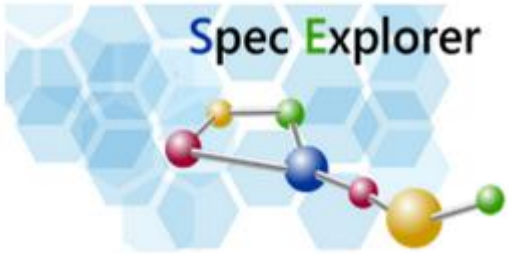
We want a trace where the loop is executed twice.

More Applications



Some Applications at Microsoft





Validity

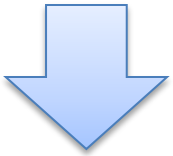
F is VALID

iff

$not\ F$ is UNSATISFIABLE

Prove that

$$x \geq 1, y \geq 1 \Rightarrow x + y \geq 2$$



Is

$$x \geq 1, y \geq 1, not(x + y \geq 2)$$

UNSAT?

Download

Yices

http://yices.csl.sri.com

Z3

http://research.microsoft.com/projects/z3

Available for Windows, OSX and Linux

SMT-Lib

http://www.smtlib.org

Online Tutorials

http://rise4fun.com/z3/tutorial

http://rise4fun.com/z3py/tutorial

Roadmap

Lecture 1: Introduction, SAT

Lecture 2: SMT, EUF, Linear Arithmetic

Lecture 3: Quantifiers

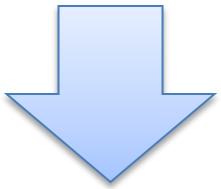
Lecture 4: Applications and Challenges

SAT

Propositional Logic

CNF

$$p_1 \vee \neg p_2, \qquad \neg p_1 \vee p_2 \vee p_3, \qquad p_3$$



$$p_1 = true, \qquad p_2 = true, \qquad p_3 = true$$

CNF is a set (conjunction) set of clauses
Clause is a disjunction of literals
Literal is an atom or the negation of an atom

Conversion to CNF

$$
\begin{aligned}
CNF(p, \Delta) &= \langle p, \Delta \rangle \\
CNF(\neg\phi, \Delta) &= \langle \neg l, \Delta' \rangle, \text{ where } \langle l, \Delta' \rangle = CNF(\phi, \Delta) \\
CNF(\phi_1 \wedge \phi_2, \Delta) &= \langle p, \Delta' \rangle, \text{ where} \\
&\quad \langle l_1, \Delta_1 \rangle = CNF(\phi_1, \Delta) \\
&\quad \langle l_2, \Delta_2 \rangle = CNF(\phi_2, \Delta_1) \\
&\quad p \text{ is fresh} \qquad p \Leftrightarrow l_1 \vee l_2 \\
&\quad \Delta' = \Delta_2 \cup \{\neg p \vee l_1, \neg p \vee l_2, \neg l_1 \vee \neg l_2 \vee p\} \\
CNF(\phi_1 \vee \phi_2, \Delta) &= \langle p, \Delta' \rangle, \text{ where } \ldots \\
&\quad \Delta' = \Delta_2 \cup \{\neg p \vee l_1 \vee l_2, \neg l_1 \vee p, \neg l_2 \vee p\}
\end{aligned}
$$

Theorem: ϕ and $l \wedge \Delta$ are equisatisfiable, where $CNF(\phi, \emptyset) = \langle l, \Delta \rangle$.

Conversion to CNF

$$CNF(\neg(\underbrace{q_1 \wedge \overbrace{(q_2 \vee \neg q_3)}^{p_1}}_{p_2}), \emptyset)$$

Conversion to CNF

$$CNF(\neg(\underbrace{q_1 \wedge \overbrace{(q_2 \vee \neg q_3)}^{p_1}}_{p_2}), \emptyset)$$

$$CNF(q_2 \vee \neg q_3, \emptyset) = \langle p_1, \{\underbrace{\neg p_1 \vee q_2 \vee \neg q_3, \neg q_2 \vee p_1, q_3 \vee p_1}\}\rangle$$

$$p_1 \Leftrightarrow q_2 \vee \neg q_3$$

$$CNF(q_1, \emptyset) = \langle q_1, \emptyset \rangle$$

Conversion to CNF

$$CNF(\neg(\underbrace{q_1 \wedge \overbrace{(q_2 \vee \neg q_3)}^{p_1}}_{p_2}), \emptyset) =$$

$$\begin{aligned}
\langle \neg p_2, \{ \quad & \neg p_1 \vee q_2 \vee \neg q_3, \\
& \neg q_2 \vee p_1, \\
& q_3 \vee p_1, \\
& \neg p_2 \vee q_1, \\
& \neg p_2 \vee p_1, \\
& \neg q_1 \vee \neg p_1 \vee p_2 \} \rangle
\end{aligned}$$

Conversion to CNF: Improvements

Maximize sharing & canonicity in the input formula F.

Cache $\phi \mapsto l$, when $CNF(\phi, \Delta) = \langle l, \Delta' \rangle$.

Support for multiary $\vee$ and $\wedge$.

...

Two procedures

Resolution	DPLL
Proof-finder	Model-finder
Saturation	Search

Resolution

$$C \vee l, \quad D \vee \neg l \quad \Rightarrow \quad C \vee D$$

$$l, \neg l \quad \Rightarrow \quad \textbf{unsat}$$

Improvements

Delete tautologies $l \vee \neg l \vee C$

Ordered Resolution

Subsumption (delete redundant clauses)

C *subsumes* $C \vee D$

…

Resolution: Example

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r$$

Resolution: Example

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r$$

Resolution: Example

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r,\ q \vee r$$

Resolution: Example

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r,\ q \vee r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r,\ q \vee r,\ r$$

Resolution: Example

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r,\ q \vee r \quad \Rightarrow$$

$$\neg p \vee \neg q \vee r,\ \neg p \vee q,\ p \vee r,\ \neg r,\ \neg q \vee r,\ q \vee r,\ r \quad \Rightarrow$$

unsat

Resolution: Correctness

Progress: Bounded number of clauses. Each application of resolution generates a new clause.

Conservation: For any model M, if $M \models C \vee l$ and $M \models D \vee \bar{l}$, then $M \models C \vee D$.

Canonicity: Given an irreducible non-**unsat** state in the atoms $p_1, \ldots, p_n$ with $p_i \prec p_{i+1}$, build a series of partial interpretations M_i as follows:

1. Let $M_0 = \emptyset$
2. If p_{i+1} is not the maximal atom in some clause that is not already satisfied in M_i, then $M_{i+1} = M_i[p_{i+1} := \mathit{false}]$.
3. If some $p_{i+1} \vee C$ is not already satisfied in M_i, then $M_{i+1} = M_i[p_{i+1} := \mathit{true}]$.

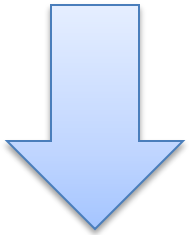
Resolution: Correctness

Suppose C and D are false in M_i

Let $j = i + 1$

p_j is maximal in

$p_j \vee C, \qquad \neg pj \vee D$



$C \vee D$ is false in M_i

Resolution: Problem

Exponential time and space

Unit Resolution

$$C \vee l,\ \neg l \quad \Rightarrow \quad C$$

C subsumes $C \vee l$

Complete for Horn Clauses

$$\neg q_1 \vee \ \ldots \vee \neg qn \vee p$$

DPLL

Split rule

S

S, p $S, \neg p$

DPLL = Unit Resolution + Split rule

Pure Literals

A literal is pure if only occurs positively or negatively.

Example :

$\varphi = (\neg x_1 \vee x_2) \wedge (x_3 \vee \neg x_2) \wedge (x_4 \vee \neg x_5) \wedge (x_5 \vee \neg x_4)$

$\neg x_1$ and x_3 are pure literals

Pure literal rule :

Clauses containing pure literals can be removed from the formula (i.e. just satisfy those pure literals)

$$\varphi_{\neg x_1, x_3} = (x_4 \vee \neg x_5) \wedge (x_5 \vee \neg x_4)$$

Preserve satisfiability, not logical equivalency !

DPLL

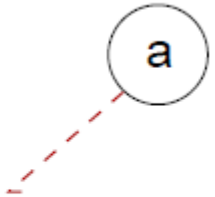
- Standard backtrack search
- DPLL(F) :
  - Apply unit propagation
  - If conflict identified, return UNSAT
  - Apply the pure literal rule
  - If F is satisfied (empty), return SAT
  - Select decision variable x
    - If DPLL($F \wedge x$)=SAT return SAT
    - return DPLL($F \wedge \neg x$)

DPLL : Example

$$
\begin{aligned}
\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\
& (\neg b \vee \neg d \vee \neg e) \wedge \\
& (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\
& (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)
\end{aligned}
$$

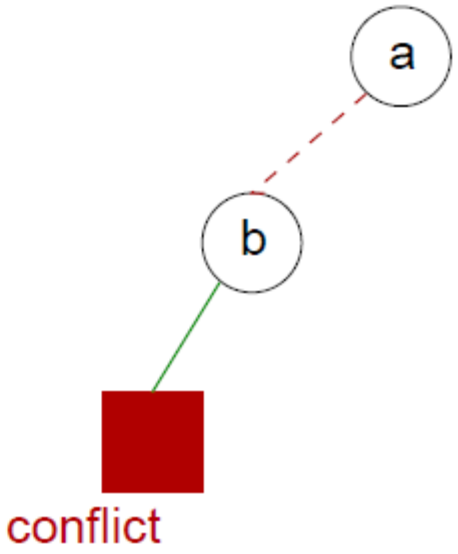
DPLL : Example

$$\begin{aligned}\varphi \quad = \quad & (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge \\ & (\neg b \vee \neg d \vee \neg e) \wedge \\ & (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge \\ & (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)\end{aligned}$$



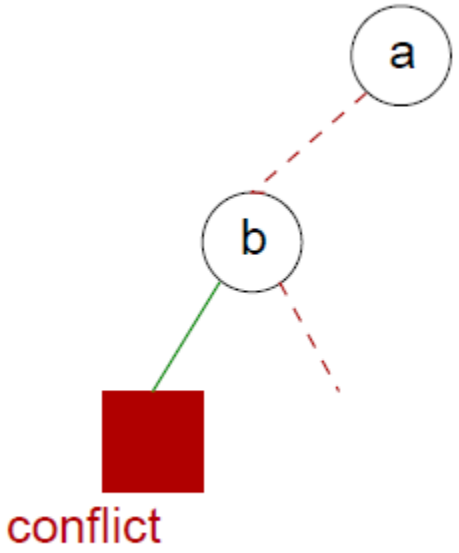
DPLL : Example

$$\varphi = (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge (\neg b \vee \neg d \vee \neg e) \wedge (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)$$



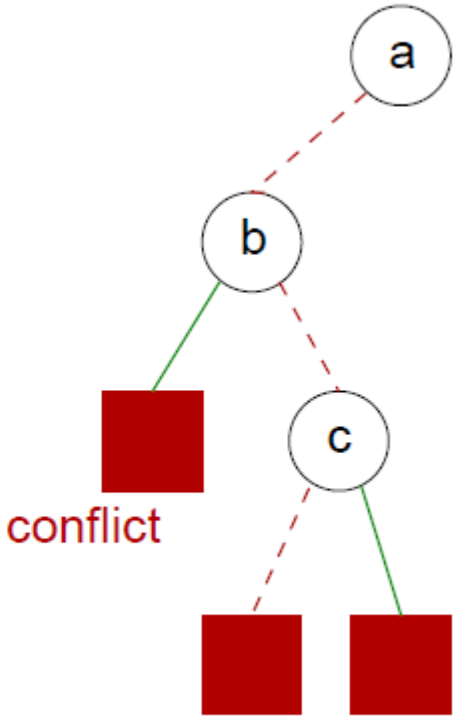
DPLL : Example

$$\varphi = (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge (\neg b \vee \neg d \vee \neg e) \wedge (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)$$



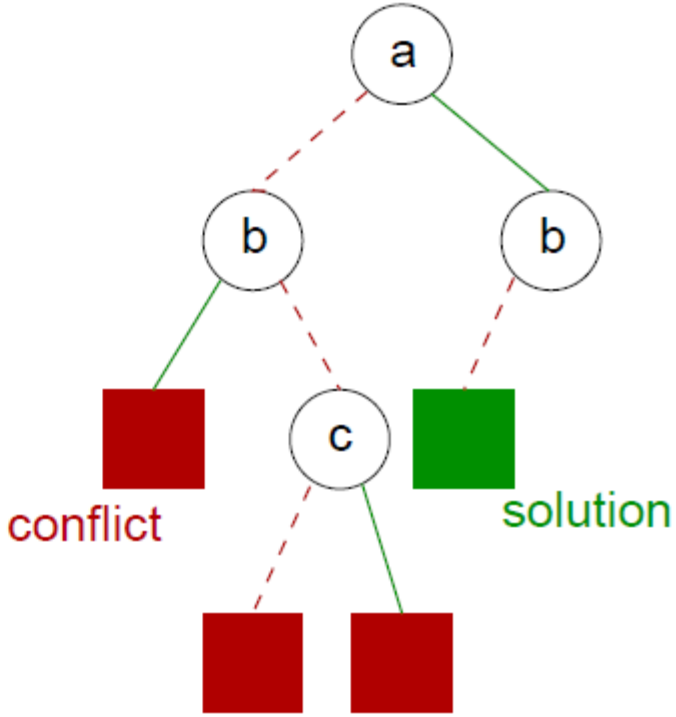
DPLL : Example

$$\varphi = (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge (\neg b \vee \neg d \vee \neg e) \wedge (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)$$

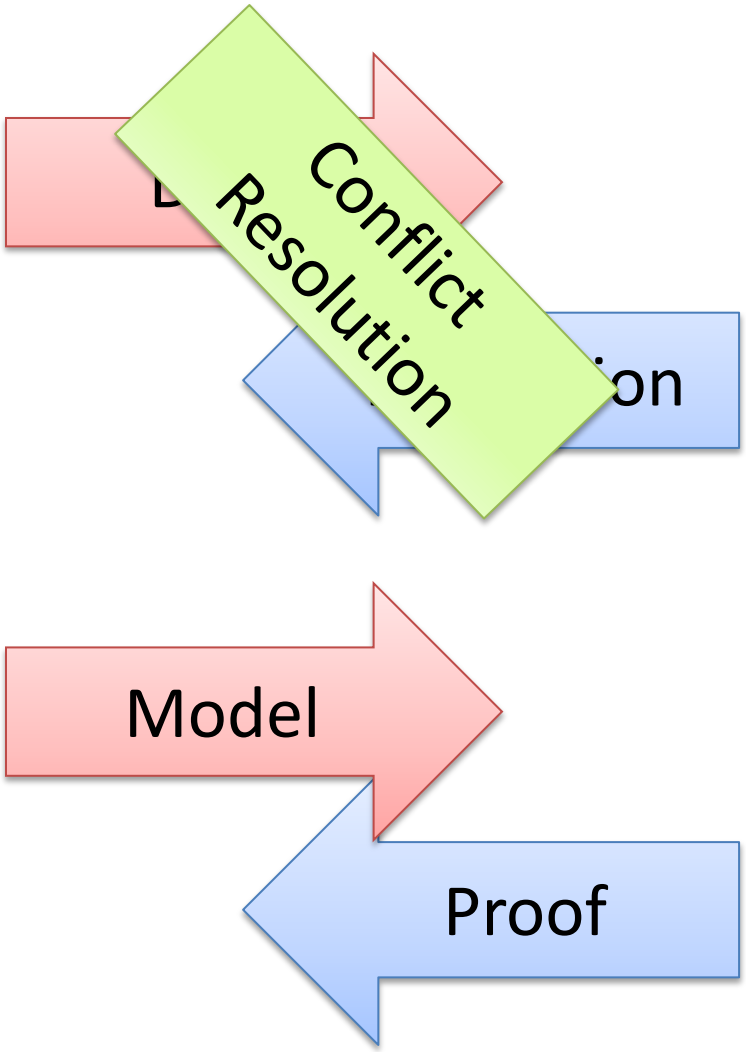


DPLL : Example

$$\varphi = (a \vee \neg b \vee d) \wedge (a \vee \neg b \vee e) \wedge (\neg b \vee \neg d \vee \neg e) \wedge (a \vee b \vee c \vee d) \wedge (a \vee b \vee c \vee \neg d) \wedge (a \vee b \vee \neg c \vee e) \wedge (a \vee b \vee \neg c \vee \neg e)$$



CDCL: Conflict Driven Clause Learning



CDCL: Conflict Driven Clause Learning

Modern SAT solvers are based on CDCL

- Backjumping
- Learning
- Restarts
- Indexing

Abstract CDCL/DPLL

$$M \parallel F$$

Partial model

Set of clauses

Abstract CDCL/DPLL

$$M \parallel F \implies M\, l \parallel F \quad \text{if} \quad \begin{cases} l \text{ or } \bar{l} \text{ occurs in } F, \\ l \text{ is undefined in } M \end{cases} \quad \text{(Decide)}$$

$$M \parallel F, C \vee l \implies M\, l_{C \vee l} \parallel F, C \vee l \quad \text{if} \quad \begin{cases} M \models \neg C, \\ l \text{ is undefined in } M \end{cases} \quad \text{(UnitPropagate)}$$

$$M \parallel F, C \implies M \parallel F, C \parallel C \quad \text{if} \quad M \models \neg C \quad \text{(Conflict)}$$

$$M \parallel F \parallel C \vee \bar{l} \implies M \parallel F \parallel D \vee C \quad \text{if} \quad l_{D \vee l} \in M, \quad \text{(Resolve)}$$

$$M \parallel F \parallel C \implies M \parallel F, C \parallel C \quad \text{if} \quad C \notin F \quad \text{(Learn)}$$

$$M\, l'\, M' \parallel F \parallel C \vee l \implies M\, l_{C \vee l} \parallel F \quad \text{if} \quad \begin{cases} M \models \neg C, \\ l \text{ is undefined in } M \end{cases} \quad \text{(Backjump)}$$

$$M \parallel F \parallel \square \implies \textit{unsat} \quad \text{(Unsat)}$$

Abstract CDCL : Example

$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$

Abstract CDCL : Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract CDCL : Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract CDCL : Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract CDCL : Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract CDCL : Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract CDCL : Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract CDCL : Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2^{}_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2^{}_{\overline{1} \vee 2}\ 3 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2^{}_{\overline{1} \vee 2}\ 3\ 4^{}_{\overline{3} \vee 4} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2^{}_{\overline{1} \vee 2}\ 3\ 4^{}_{\overline{3} \vee 4}\ 5 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2^{}_{\overline{1} \vee 2}\ 3\ 4^{}_{\overline{3} \vee 4}\ 5\ \overline{6}^{}_{\overline{5} \vee \overline{6}} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Conflict)}$$

$$1\ 2^{}_{\overline{1} \vee 2}\ 3\ 4^{}_{\overline{3} \vee 4}\ 5\ \overline{6}^{}_{\overline{5} \vee \overline{6}} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \| \quad 6 \vee \overline{5} \vee \overline{2}$$

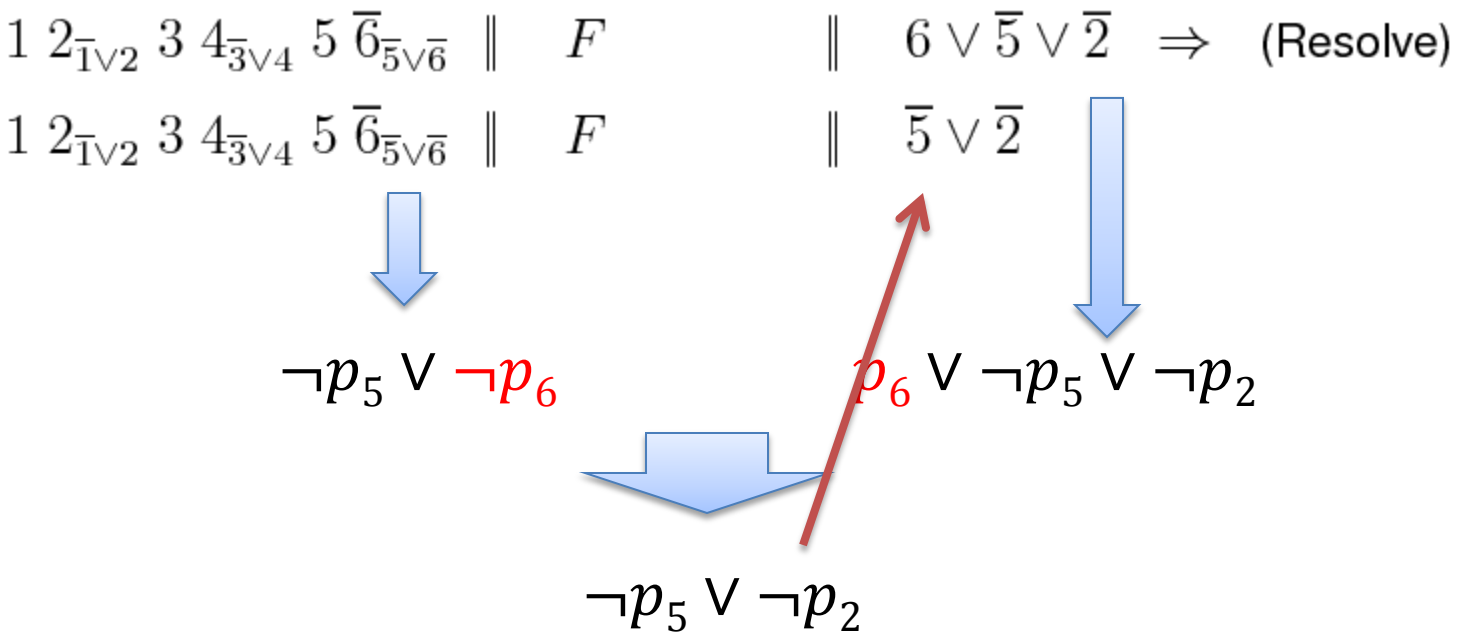
Abstract CDCL : Example

$$
\begin{array}{rcl}
 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)} \\
1 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)} \\
1\ 2^{}_{\overline{1} \vee 2} & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)} \\
1\ 2^{}_{\overline{1} \vee 2}\ 3 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)} \\
1\ 2^{}_{\overline{1} \vee 2}\ 3\ 4^{}_{\overline{3} \vee 4} & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)} \\
1\ 2^{}_{\overline{1} \vee 2}\ 3\ 4^{}_{\overline{3} \vee 4}\ 5 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)} \\
1\ 2^{}_{\overline{1} \vee 2}\ 3\ 4^{}_{\overline{3} \vee 4}\ 5\ \overline{6}^{}_{\overline{5} \vee \overline{6}} & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Conflict)} \\
1\ 2^{}_{\overline{1} \vee 2}\ 3\ 4^{}_{\overline{3} \vee 4}\ 5\ \overline{6}^{}_{\overline{5} \vee \overline{6}} & \| & \underbrace{\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}}_{F} \quad \| \quad 6 \vee \overline{5} \vee \overline{2}
\end{array}
$$

Abstract CDCL : Example

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \quad \| \quad F \quad \| \quad 6 \vee \overline{5} \vee \overline{2}$$

Abstract CDCL : Example



Abstract CDCL : Example

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F \parallel 6\vee\overline{5}\vee\overline{2} \Rightarrow \text{(Resolve)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F \parallel \overline{5}\vee\overline{2} \Rightarrow \text{(Learn)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F,\ \overline{5}\vee\overline{2} \parallel \overline{5}\vee\overline{2}$$

Abstract CDCL : Example

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F \parallel 6\vee\overline{5}\vee\overline{2} \Rightarrow \text{(Resolve)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F \parallel \overline{5}\vee\overline{2} \Rightarrow \text{(Learn)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F, \overline{5}\vee\overline{2} \parallel \overline{5}\vee\overline{2} \Rightarrow \text{(Resolve)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F, \overline{5}\vee\overline{2} \parallel \overline{5}\vee\overline{1}$$

Abstract CDCL : Example

$$
\begin{array}{rlll}
1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} & \| \quad F & \| \quad 6\vee\overline{5}\vee\overline{2} & \Rightarrow \text{ (Resolve)} \\
1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} & \| \quad F & \| \quad \overline{5}\vee\overline{2} & \Rightarrow \text{ (Learn)} \\
1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} & \| \quad F,\ \overline{5}\vee\overline{2} & \| \quad \overline{5}\vee\overline{2} & \Rightarrow \text{ (Resolve)} \\
1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} & \| \quad F,\ \overline{5}\vee\overline{2} & \| \quad \overline{5}\vee\overline{1} & \Rightarrow \text{ (Backjump)} \\
1\ 2_{\overline{1}\vee 2}\ \overline{5}_{\overline{5}\vee\overline{1}} & \| \quad F,\ \overline{5}\vee\overline{2} & &
\end{array}
$$

Abstract CDCL

- Support different strategies.
  - Example: learn 0 or several clauses per conflict.
- Does it terminate?
  - Each decision defines a new scope level.
  - Metric: number of assigned literals per level.

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \quad \mapsto \quad (2,2,2)$$
$$1\ 2_{\overline{1}\vee 2}\ \overline{5}_{\overline{5}\vee\overline{1}} \quad \mapsto \quad (3)$$

  - **Decide**, **UnitPropagate**, and **Backjump** increase the metric.
  - It can not increase forever (finite number of variables).
  - Conflict resolution rules (**Conflict**, **Resolve**, **Learn**) are also terminating.

Abstract CDCL : Strategy

- Abstract DPLL is very flexible.
- Basic Strategy:
  - Only apply **Decide** if **UnitPropagate** and **Conflict** cannot be applied.
- Conflict Resolution:
  - Learn only one clause per conflict (the clause used in **Backjump**).
  - Use **Backjump** as soon as possible (FUIP).

Abstract CDCL : Decision Strategy

- Decision heuristic:
  - Associate a score with each boolean variable.
  - Select the variable with highest score when **Decide** is used.
  - Increase by δ the score of $var(l)$ when **Resolve** is used:

    $M \parallel F \parallel C \vee \bar{l} \quad \Longrightarrow \quad M \parallel F \parallel D \vee C \quad$ **if** $l_{D \vee l} \in M,$ **(Resolve)**
  - Increase the score of every variable in the clause $C \vee l$ when **Backjump** is used:

    $$M\, l'\, M' \parallel F \parallel C \vee l \implies M\, l_{C \vee l} \parallel F' \quad \text{if} \begin{cases} M \models \neg C, \\ l \text{ is undefined in } M \end{cases} \quad \textbf{(Backjump)}$$
  - After each conflict: slightly increase the value of δ.
  - From time to time renormalize the scores and δ to avoid overflows.

Abstract CDCL : Phase Selection

- Assume p was selected by a decision strategy.

  Should we assign p or $\neg p$ in **Decide**?

  Always False Guess $\neg p$ (works well in practice).

  Always True Guess p.

  Score Associate a score with each literal instead of each variable.
  Pick the phase with highest score.

  Caching Caches the last phase of variables during conflict resolution. Improvement: except for variables in the last decision level.

  Greedy Select the phase that satisfies most clauses.

Abstract CDCL : Extra Rules

- Extra rules:

$$M \parallel F, C \quad \Longrightarrow \quad M \parallel F \qquad \text{if } C \text{ is a learned clause} \qquad \textbf{(Forget)}$$

$$M \parallel F \quad \Longrightarrow \quad \parallel F \qquad \textbf{(Restart)}$$

- **Forget** in practice:
  - Associate a score with each learned clause C.
  - Increase by δ_c the score of $D \vee l$ when **Resolve** is used.

$$M \parallel F \parallel C \vee \bar{l} \quad \Longrightarrow \quad M \parallel F \parallel D \vee C \qquad \text{if } l_{D \vee l} \in M, \qquad \textbf{(Resolve)}$$

  - From time to time use **Forget** to delete learned clauses with low score.

Abstract CDCL : Restart Strategies

No restarts

Linear Restart after every k conflicts, update $k := k + \delta$.

Geometric Restart after every k conflicts, update $k := k \times \delta$.

Inner-Out Geometric "Two dimensional pattern" that increases in both dimensions.

- Initially $k := x$, the inner loop multiplies k by δ at each restart.
- When $k > y$, $k := x$ and $y := y \times \delta$.

Luby Restarts are performed according to the following series: $1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, \ldots$, multiplied by a constant c (e.g., 100, 256, 512).

$$\mathit{luby}(i) = \begin{cases} 2^{k-1}, & \text{if } \exists k.\ i = 2^k - 1 \\ \mathit{luby}(i - 2^{k-1} + 1), & \text{if } \exists k.\ 2^{k-1} \leq i < 2^k - 1 \end{cases}$$

Abstract CDCL : Indexing

- Indexing techniques are very important.
- How to implement **UnitPropagate** and **Conflict**?
- Scanning the set of clauses will not scale.
- Simple index: mapping from literals to clauses (occurrences).
  - *watch*$(l) = \{C_1, \ldots, C_n\}$, where $\bar{l} \in C_i$
  - If l is assigned, check each clause $C \in$ *watch*(l) for **UnitPropagate** and **Conflict**.
  - Most of the time C has more than one unassigned literal.
  - Improvement: associate a counter u with each clause (number of unassigned literals).
  - Problem: counters must be decremented when literals are assigned, and restored during **Backjump**.

Abstract CDCL : Indexing

- Indexing techniques are very important.
- How to implement **UnitPropagate** and **Conflict**?
- Scanning the set of clauses will not scale.
- Simple index: mapping from literals to clauses (occurrences).
  - *watch*$(l) = \{C_1, \ldots, C_n\}$, where $\bar{l} \in C_i$
  - If l is assigned, check each clause $C \in$ *watch*(l) for **UnitPropagate** and **Conflict**.
  - Most of the time C has more than one unassigned literal.
  - Improvement: associate a counter u with each clause (number of unassigned literals).
  - Problem: counters must be decremented when literals are assigned, and restored during **Backjump**.

Indexing : Two watch literal

- Insight:
  - No need to include clause C in every set *watch*(l) where $\bar{l} \in C$.
  - It suffices to include C in at most 2 such sets.
- Invariant:

  If some literal l in C is not assigned to false, then $C \in$ *watch*(l') of some l' that is not assigned to false.

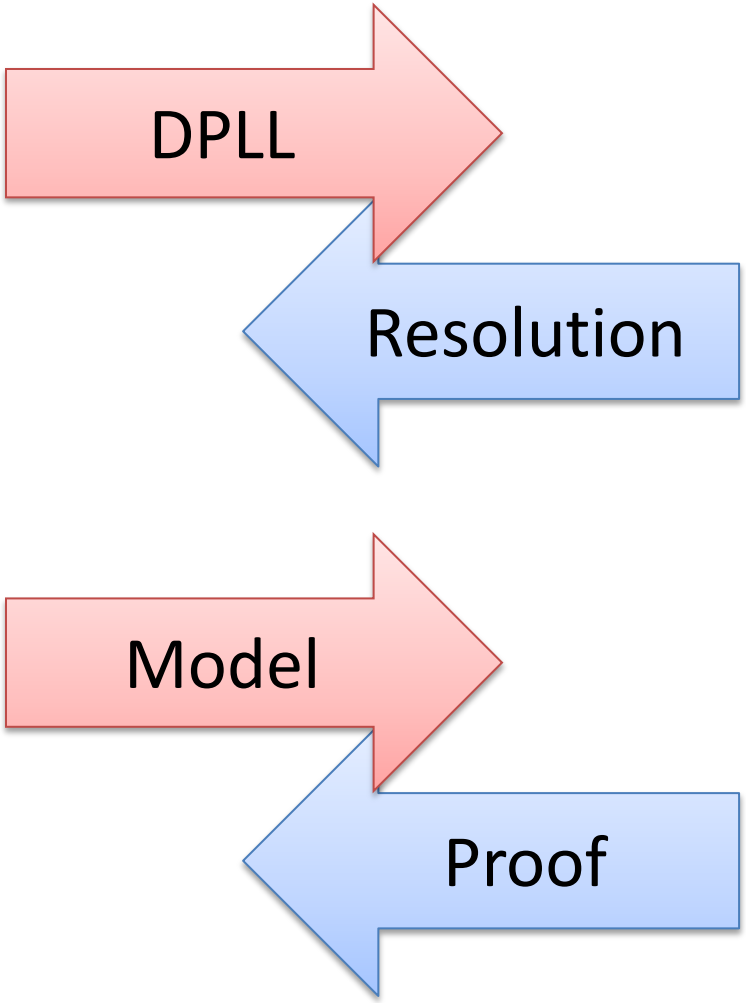
Indexing : Two watch literal

- Maintain 2-watch invariant:
  - Whenever l is assigned.
  - For each clause $C \in \mathit{watch}(l)$
    - If the other watch literal l' ($C \in \mathit{watch}(l')$) is assigned to true, then do nothing.
    - Else if some other literal l' is true or unassigned

$$\begin{aligned} \mathit{watch}(l') &:= \mathit{watch}(l') \cup \{C\} \\ \mathit{watch}(l) &:= \mathit{watch}(l) \setminus \{C\} \end{aligned}$$

    - Else if all literals in C are assigned to false, then **Backjump**.
    - Else (all but one literal in C is assigned to false) **Propagate**.

CDCL: Conflict Driven Clause Learning



Preprocessing & Inprocessing

- Preprocessing step is very important for industrial benchmarks.
- Formula $\rightsquigarrow$ CNF (already covered).
- Subsumption: C subsumes D if $C \subseteq D$.
- Resolution: eliminate cheap variables.
  - $\mathit{occs}(l) = \{\text{clauses that contain } l\}$
  - $|\mathit{occs}(p)| * |\mathit{occs}(\neg p)| < k$
  - $|\mathit{occs}(p)| = 1$ or $|\mathit{occs}(\neg p)| = 1$

Homework

Install Yices & Z3 in your notebook

http://yices.csl.sri.com

http://research.microsoft.com/projects/z3

Yices input language:

http://yices.csl.sri.com/language.shtml

Online tutorials:

http://rise4fun.com/z3/tutorial SMT 2.0

http://rise4fun.com/z3py/tutorial Python