Decision Procedures and Verification

Martin Blicha

Charles University

5.3.2018

Algorithms for Propositional Satisfiability

Why study propositional satisfiability?

- Interesting from both theoretical and practical perspective
- First problem to be proven NP-complete [Cook '71, Levin '73]
- Many industrial problems encoded as SAT
  - Hardware and software verification
  - Automated planning - Planning as Satisfiability
  - Product configuration
  - ...

Progress in SAT solving

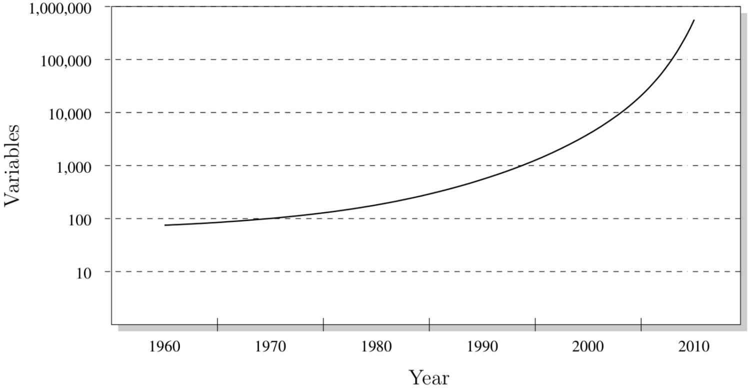


Image source: Decision Procedures. Kroening D., Strichman O.

Progress in SAT solving

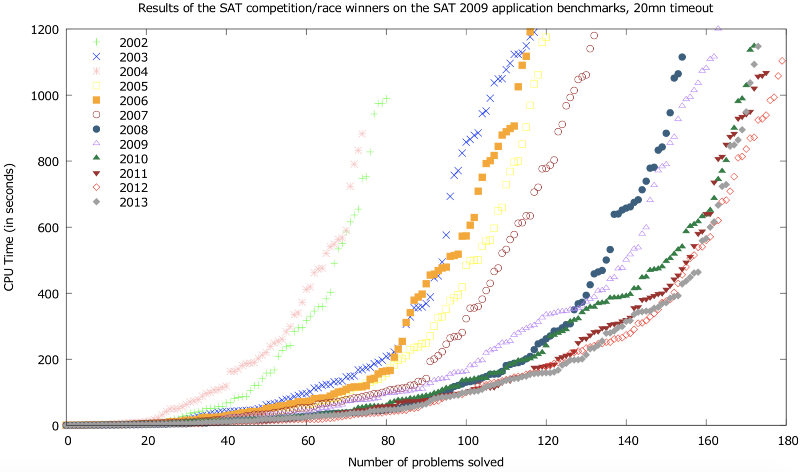


Image source: Decision Procedures. Kroening D., Strichman O.

Approaches to SAT solving

DPLL framework

- complete procedure

Stochastic search

- incomplete procedure

Approaches to SAT solving

DPLL framework

- complete procedure
- very efficient for instance with structure

Stochastic search

- incomplete procedure
- better at solving random satisfiable instances

Approaches to SAT solving

DPLL framework

- complete procedure
- very efficient for instance with structure
- important when proof of unsatiafiability required (e.g. verification)

Stochastic search

- incomplete procedure
- better at solving random satisfiable instances
- can be faster to obtain satisfying assignment

Problems of naive satisfiability algorithm

Naive algorithm

Enumerate all assignments. Check if formula is satisfied under any of them.

Problems of naive satisfiability algorithm

Naive algorithm

Enumerate all assignments. Check if formula is satisfied under any of them.

- Unnecessary repetition of partial assignment leading to conflict.
- No information preserved between tries of different assignments
- Lots of unnecessary work being done over and over again.

DPLL algorithm - overview

- *DPLL* algorithm (Davis-Putnam-Loveland-Logemann, 1962)
- Input formula assumed to be in CNF
- *Search* in a tree of partial assignments
- *Backtracking* on conflict
- *Unit propagation* prunes the tree

Basic concepts

Definition (state of a clause)

Let $\alpha : V \to \{True, False\}$ be an assignment of variables from V. Then generalization of α on a clauses of set of variables $V' \supseteq V$ is $\alpha^\star : \{c \mid c \text{ is a clause over } V'\} \to \{True, False, Undef\}$.

- c is satisfied, $\alpha^\star(c) = True$, if at least one literal in c is satisfied by α
- c is conflicting, $\alpha^\star(c) = False$, if all literals are falsified by α
- c is unresolved, $\alpha^\star(c) = Undef$, otherwise.

Basic concepts

Definition (state of a clause)

Let $\alpha : V \rightarrow \{True, False\}$ be an assignment of variables from V. Then generalization of α on a clauses of set of variables $V' \supseteq V$ is $\alpha^\star : \{c \mid c \text{ is a clause over } V'\} \rightarrow \{True, False, Undef\}$.

- c is satisfied, $\alpha^\star(c) = True$, if at least one literal in c is satisfied by α
- c is conflicting, $\alpha^\star(c) = False$, if all literals are falsified by α
- c is unresolved, $\alpha^\star(c) = Undef$, otherwise.

Definition (unit clause)

A clause c is unit under assignment α if it is not satisfied and all but one literals are falsified by α.

Basic concepts

Example

Let α be $\{x_1 \mapsto 1, x_2 \mapsto 0, x_4 \mapsto 1\}$. Then

- $x_1 \vee x_3 \vee \neg x_4$ is satisfied,
- $\neg x_1 \vee x_2$ is conflicting,
- $\neg x_1 \vee \neg x_4 \vee x_3$ is unit,
- $\neg x_1 \vee x_3 \vee x_5$ is unresolved.

Basic concepts

Example

Let α be $\{x_1 \mapsto 1, x_2 \mapsto 0, x_4 \mapsto 1\}$. Then

- $x_1 \vee x_3 \vee \neg x_4$ is satisfied,
- $\neg x_1 \vee x_2$ is conflicting,
- $\neg x_1 \vee \neg x_4 \vee x_3$ is unit,
- $\neg x_1 \vee x_3 \vee x_5$ is unresolved.

Definition (unit clause rule, antecedent clause)

Given a partial assignment α and a clause c that is unit under α, α must be extended so that is satisfies the last unassigned literal l. We say that l is implied by c (under α) and we call c the *antecedent* of l.

Basic concepts

Example

Let α be $\{x_1 \mapsto 1, x_2 \mapsto 0, x_4 \mapsto 1\}$. Then

- $x_1 \vee x_3 \vee \neg x_4$ is satisfied,
- $\neg x_1 \vee x_2$ is conflicting,
- $\neg x_1 \vee \neg x_4 \vee x_3$ is unit,
- $\neg x_1 \vee x_3 \vee x_5$ is unresolved.

Definition (unit clause rule, antecedent clause)

Given a partial assignment α and a clause c that is unit under α, α must be extended so that is satisfies the last unassigned literal l. We say that l is implied by c (under α) and we call c the *antecedent* of l.

Example

The clause $c = \neg x_1 \vee \neg x_4 \vee x_3$ and the partial assignment $\{x_1 \mapsto 1, x_4 \mapsto 1\}$ imply $x_3 \mapsto 1$ and $Antecedent(x_3) = c$.

The power of unit propagation

- The goal is to satisfy a CNF formula.
- $(\neg u \vee w) \wedge (u \vee v) \wedge (u) \wedge (\neg w \vee z)$; $\alpha = \{\}$
  - (u) is unit under α

The power of unit propagation

- The goal is to satisfy a CNF formula.

- $(\neg u \vee w) \wedge (u \vee v) \wedge (u) \wedge (\neg w \vee z)$; $\alpha = \{u\}$
  - $(\neg u \vee w)$ is unit under α

The power of unit propagation

- The goal is to satisfy a CNF formula.
- $(\neg u \vee w) \wedge (u \vee v) \wedge (u) \wedge (\neg w \vee z)$;
  $\alpha = \{u, w\}$
  - $(\neg w \vee z)$ is unit under α

The power of unit propagation

- The goal is to satisfy a CNF formula.

- $(\neg u \vee w) \wedge (u \vee v) \wedge (u) \wedge (\neg w \vee z)$;
  $\alpha = \{u, w, z\}$
  - All clauses are satisfied by α.

  Solved by unit propagation. No decisions needed.

DPLL algorithm

procedure $\mathrm{DPLL}(\varphi, \alpha)$
    if $\forall c \in \varphi$ **then** c is satisfied by α **return** TRUE
    if $\exists c \in \varphi$ **then** c is conflicting under α **return** FALSE
    $\alpha \leftarrow \alpha \cup$ *UNIT-PROPAGATION*()
    $x \leftarrow$ *SELECT-VAR*()
    if DPLL($\alpha \cup \{x \mapsto 1\}$) **then return** TRUE
    if DPLL($\alpha \cup \{x \mapsto 0\}$) **then return** TRUE
    return FALSE

DPLL algorithm

Notes

- UNIT-PROPAGATION applies unit clause rule until no clauses are unit
- After unit propagation, assignment can be extended with *pure* literals.
  - But this is not used in practice (too costly).
- The phase of unit propagation possibly with pure literals is often referred to as *BCP* - Boolean constraint propagation
- SELECT-VAR selects an unassigned variable. Both values of the variable are tried.

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
$(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
   $\underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge$
   $\underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge$
   $(\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
   $(\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
   $(\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
   $(\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
   $(\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
   $(\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL - running example

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

1. Decide $\alpha(x_1) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide $\alpha(x_2) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\alpha(x_4) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive $\alpha(x_3) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

$\alpha = \{x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0\}$ is a satisfying assignment of φ

Deficiencies of DPLL

$$\psi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4) \wedge (\neg y_1 \vee y_3 \vee y_4) \wedge (\neg y_2 \vee y_3 \vee y_4) \wedge (\neg y_3 \vee \neg y_4) \wedge (\neg y_3 \vee y_4) \wedge (y_3 \vee \neg y_4)$$

- fixed variable ordering: $y_1, x_1, x_2, x_3, x_4, y_2, y_3, y_4$
- no pure literal propagation

Deficiencies of DPLL

$$\psi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4) \wedge (\neg y_1 \vee y_3 \vee y_4) \wedge (\neg y_2 \vee y_3 \vee y_4) \wedge (\neg y_3 \vee \neg y_4) \wedge (\neg y_3 \vee y_4) \wedge (y_3 \vee \neg y_4)$$

- fixed variable ordering: $y_1, x_1, x_2, x_3, x_4, y_2, y_3, y_4$
- no pure literal propagation
- algorithm tries $x_1 \mapsto 1$ for both branches $y_1 \mapsto 1$ and $y_1 \mapsto 0$

Deficiencies of DPLL

$$\psi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4) \wedge$$
$$(\neg y_1 \vee y_3 \vee y_4) \wedge (\neg y_2 \vee y_3 \vee y_4) \wedge (\neg y_3 \vee \neg y_4) \wedge (\neg y_3 \vee y_4) \wedge (y_3 \vee \neg y_4)$$

- fixed variable ordering: $y_1, x_1, x_2, x_3, x_4, y_2, y_3, y_4$
- no pure literal propagation
- algorithm tries $x_1 \mapsto 1$ for both branches $y_1 \mapsto 1$ and $y_1 \mapsto 0$
- repeats *the same conflict* on x variables in both of these branches

Deficiencies of DPLL

$\psi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4) \wedge$
$(\neg y_1 \vee y_3 \vee y_4) \wedge (\neg y_2 \vee y_3 \vee y_4) \wedge (\neg y_3 \vee \neg y_4) \wedge (\neg y_3 \vee y_4) \wedge (y_3 \vee \neg y_4)$

- fixed variable ordering: $y_1, x_1, x_2, x_3, x_4, y_2, y_3, y_4$
- no pure literal propagation
- algorithm tries $x_1 \mapsto 1$ for both branches $y_1 \mapsto 1$ and $y_1 \mapsto 0$
- repeats *the same conflict* on x variables in both of these branches
- DPLL can repeat the same mistake over and over again

Deficiencies of DPLL

$$\psi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4) \wedge (\neg y_1 \vee y_3 \vee y_4) \wedge (\neg y_2 \vee y_3 \vee y_4) \wedge (\neg y_3 \vee \neg y_4) \wedge (\neg y_3 \vee y_4) \wedge (y_3 \vee \neg y_4)$$

- fixed variable ordering: $y_1, x_1, x_2, x_3, x_4, y_2, y_3, y_4$
- no pure literal propagation
- algorithm tries $x_1 \mapsto 1$ for both branches $y_1 \mapsto 1$ and $y_1 \mapsto 0$
- repeats *the same conflict* on x variables in both of these branches
- DPLL can repeat the same mistake over and over again
- SAT solver should learn from past mistakes

Implication graph

Definition (Implication graph)

Implication graph for α is an acyclic labeled directed graph $G = (V \cup \{K\}, E)$ where:

- Vertices V correspond to variables.
  - labeled by current assignment and decision level
  - $x@N$ ($\neg x@N$): x is assigned True (False) at decision level N.
- Edges E represents *reasons* for assigning a value.
  - $(x, y) \in E$, if $\neg x \in Antecedent(y)$ with $\alpha(x) = 1$ or $x \in Antecedent(y)$ with $\alpha(x) = 0$
  - (x, y) is labeled with $Antecedent(y)$.
- Vertex K represents a conflict
  - $(x, K) \in E$, if $\neg x \in c$ with $\alpha(x) = 1$ or $x \in c$ with $\alpha(x) = 0$ where c is a conflicting clause under α.
  - (x, K) is labeled the corresponding conflict clause.

Implication graph

Definition (Implication graph)

Implication graph for α is an acyclic labeled directed graph $G = (V \cup \{K\}, E)$ where:

- Vertices V correspond to variables.
  - labeled by current assignment and decision level
  - $x@N$ ($\neg x@N$): x is assigned True (False) at decision level N.
- Edges E represents *reasons* for assigning a value.
  - $(x, y) \in E$, if $\neg x \in Antecedent(y)$ with $\alpha(x) = 1$ or $x \in Antecedent(y)$ with $\alpha(x) = 0$
  - (x, y) is labeled with $Antecedent(y)$.
- Vertex K represents a conflict
  - $(x, K) \in E$, if $\neg x \in c$ with $\alpha(x) = 1$ or $x \in c$ with $\alpha(x) = 0$ where c is a conflicting clause under α.
  - (x, K) is labeled the corresponding conflict clause.

- *Roots* (no incoming edges) correspond to decisions, inner nodes (except K) to unit propagation. If a there is a path from roots to K we call the implication graph the *conflict* graph.

Example of implication graph

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

Example of implication graph

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
   $\underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

x_1@1

Example of implication graph

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

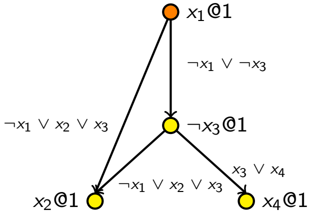
1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

$$\begin{tikzcd} x_1 @ 1 \arrow[d, "\neg x_1 \vee \neg x_3"] \\ \neg x_3 @ 1 \end{tikzcd}$$

Example of implication graph

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

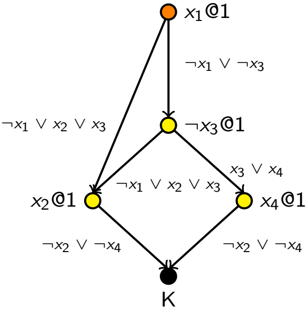
1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$



Example of implication graph

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

1. Decide $\alpha(x_1) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\alpha(x_3) = 0$:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive $\alpha(x_2) = 1, \alpha(x_4) = 1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{conflict clause}} \wedge (x_3 \vee x_4)$

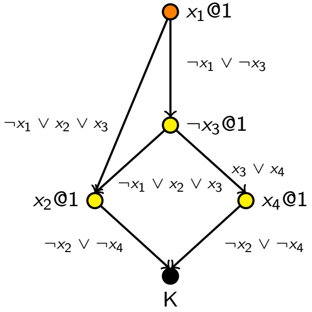


Conflict clauses as cuts in the implication graph

Definition (separating cut)

A separating cut in a conflict graph is a minimal set of edges whose removal breaks all paths from the root nodes to the conflict node.

- Each cut splits the graph to *reason* side and *conflict* side.
- The set of nodes on reason side with an edge to conflict side constitutes a sufficient condition for the conflict.
- Its negation is a conflict clause.

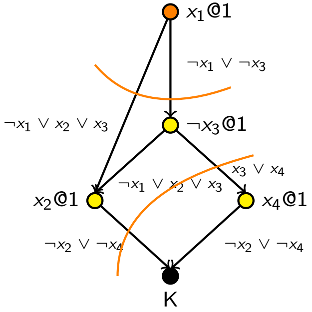


Conflict clauses as cuts in the implication graph

Definition (separating cut)

A separating cut in a conflict graph is a minimal set of edges whose removal breaks all paths from the root nodes to the conflict node.

- Each cut splits the graph to *reason* side and *conflict* side.
- The set of nodes on reason side with an edge to conflict side constitutes a sufficient condition for the conflict.
- Its negation is a conflict clause.

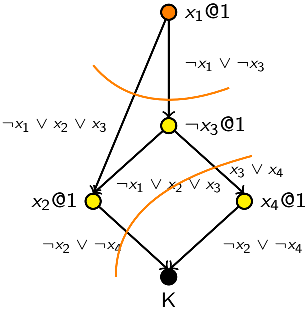


Clause learning

Observation

Every separating cut in conflict graph determines a conflict clause c such that $\varphi \to c$, where φ is the input formula.

- The conflict clause can be added to the input formula without effecting satisfiability.
- It prunes the search tree.
- This process is referred to as *learning*.
  - SAT solver is "learning" from its past mistakes.

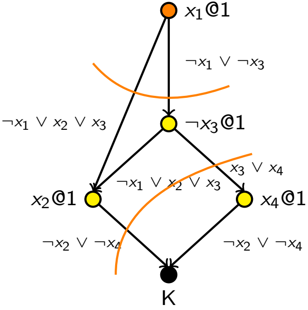


Clause learning

Observation

Every separating cut in conflict graph determines a conflict clause c such that $\varphi \rightarrow c$, where φ is the input formula.

- The conflict clause can be added to the input formula without effecting satisfiability.
- It prunes the search tree.
- This process is referred to as *learning*.
  - SAT solver is "learning" from its past mistakes.



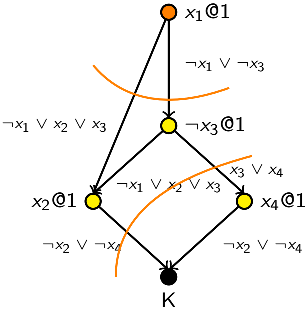
- First cut $\Rightarrow$ conflict clause $\neg x_1$.

Clause learning

Observation

Every separating cut in conflict graph determines a conflict clause c such that $\varphi \rightarrow c$, where φ is the input formula.

- The conflict clause can be added to the input formula without effecting satisfiability.
- It prunes the search tree.
- This process is referred to as *learning*.
    - SAT solver is "learning" from its past mistakes.



- First cut $\Rightarrow$ conflict clause $\neg x_1$.
- Second cut $\Rightarrow$ conflict clause $\neg x_2 \vee x_3$.

Clause learning strategies

- Different cuts correspond to different conflict clause.
- Impossible to predict if a clause will be more useful than other.
- In general smaller clauses are more desirable.
  - Less storage space
  - Earlier unit propagation
- Any number of conflict clauses could be learnt.
- Many SAT solvers learn a single clause with a special property, an *asserting* clause.

Asserting clause and UIP

Definition (asserting clause)

Asserting clause is a conflict clause that contains exactly one literal from the current decision level.

Definition

(unique implication point) Unique implication point (UIP) is any vertex other than K that is on all paths from the current decision level vertex to K.

- UIP always exists (at least the decision vertex itself)
- there may be more UIPs

Definition (first UIP)

First UIP is the UIP that is closest to K

Clause learning and backtracking

- Find the conflict clause containing the negation of first UIP as its single literal from current decision level.
  - asserting
- Backtracking with asserting clause
  - Backtrack to the *second highest decision level* from levels of literals in the conflict clause.
  - Equivalently (for asserting clause) to the highest decision level of its literals, excluding the UIP.
  - The newly learnt clause is unit at this decision level $\Rightarrow$ Unit propagation is immediately triggered.
- Notes:
  - If a conflict clause contains only literals from decision level 0, then the input formula is unsatisfiable.
  - If a conflict clause contains a single literal, the backtrack level is 0.

CDCL Algorithm

procedure $\mathrm{CDCL}(\varphi)$
    $\alpha \leftarrow \emptyset$
    if $BCP(\varphi, \alpha) = NULL$ **then return** FALSE
    while TRUE **do**
        $(x, v) \leftarrow SELECT(\varphi, \alpha)$
        if $(x, v) = NULL$ **then return** TRUE
        $\alpha \leftarrow \alpha \cup \{x \leftarrow v\}$
        $(result, \alpha) \leftarrow BCP(\varphi, \alpha)$
        while not result **do**
            $(level, \varphi) \leftarrow ANALYZE\text{-}CONFLICT(\varphi, \alpha)$
            if $level < 0$ **then return** FALSE
            $BACKTRACK(\varphi, level)$
            $(result, \alpha) \leftarrow BCP(\varphi, \alpha)$

CDCL Algorithm

- BCP. Performs unit propagation iteratively. Returns updated assignment and conflict indicator.
- SELECT. Selects unassigned variable and its polarity. Returns NULL if all variables are assigned.
- ANALYZE-CONFLICT. Determines backtrack level and extends φ with learned clause(s).
- BACKTRACK. Backtracks to the given decision level. Erases all assignments made after this level.

CDCL Algorithm

Notes

- Algorithm always terminates.
  - Idea of a proof: The algorithm never enters the same decision level with the same partial assignment twice.

CDCL Algorithm

Notes

- Algorithm always terminates.
  - Idea of a proof: The algorithm never enters the same decision level with the same partial assignment twice.
- Learned clauses can be pruned.
  - Too many learned clauses slow down the solver too much.
  - Many of them are not used more than once.

CDCL Algorithm

Notes

- Algorithm always terminates.
  - Idea of a proof: The algorithm never enters the same decision level with the same partial assignment twice.
- Learned clauses can be pruned.
  - Too many learned clauses slow down the solver too much.
  - Many of them are not used more than once.
- Implication graph can be represented implictly (decision trail with decision levels and polarity for variables, map of literals to antecedents).

Computing asserting clause

procedure $\mathrm{ANALYZE\text{-}CONFLICT}(\varphi, \alpha)$
    if *decision-level* $= 0$ **then return** $(-1, \varphi)$
    $c \leftarrow$ unsatisfied clause w.r.t. α
    while c is not asserting **do**
        $l \leftarrow$ most recently assigned literal in c
        $c \leftarrow RESOLVE(c, Antecedent(l), Var(l))$
    $\varphi \leftarrow \varphi \cup c$
    return $(LEVEL(c), \varphi)$

Computing asserting clause

procedure ANALYZE-CONFLICT(φ, α)
    if *decision-level* $= 0$ **then return** $(-1, \varphi)$
    $c \leftarrow$ unsatisfied clause w.r.t. α
    while c is not asserting **do**
        $l \leftarrow$ most recently assigned literal in c
        $c \leftarrow RESOLVE(c, Antecedent(l), Var(l))$
    $\varphi \leftarrow \varphi \cup c$
    return $(LEVEL(c), \varphi)$

- $RESOLVE(c_1, c_2, v)$ returns resolvent of c_1 and c_2 where x is the resolution variable.
- $LEVEL(c)$ returns the second highest decision level of literals in c. (Returns 0 is c has only one literal.)