



SAT and SMT Solving

Sarah Winkler

KRDB

Department of Computer Science
Free University of Bozen-Bolzano

lecture 2
WS 2022

- Summary of Last Week
- From DPLL to Conflict Driven Clause Learning
- Application: Test Case Generation

Approach

- ▶ most state-of-the-art SAT solvers use variation of Davis - Putnam - Logemann - Loveland (DPLL) procedure (1962)
- ▶ DPLL is sound and complete backtracking-based search algorithm
- ▶ can be described abstractly by transition system (Nieuwenhuis, Oliveras, Tinelli 2006)

Definition (Abstract DPLL)

- ▶ **decision literal** is annotated literal l^d
- ▶ **state** is pair $M \parallel F$ for
 - ▶ list M of (decision) literals
 - ▶ formula F in CNF
- ▶ transition rules

$$M \parallel F \quad \Longrightarrow \quad M' \parallel F' \quad \text{or} \quad \text{FailState}$$

Definition (DPLL Transition Rules)

- ▶ **unit propagation** $M \parallel F, C \vee I \implies M I \parallel F, C \vee I$
if $M \models \neg C$ and I is undefined in M
- ▶ **pure literal** $M \parallel F \implies M I \parallel F$
if I occurs in F but I^c does not occur in F , and I is undefined in M
- ▶ **decide** $M \parallel F \implies M I^d \parallel F$
if I or I^c occurs in F , and I is undefined in M
- ▶ **backtrack** $M I^d N \parallel F, C \implies M I^c \parallel F, C$
if $M I^d N \models \neg C$ and N contains no decision literals
- ▶ **fail** $M \parallel F, C \implies \text{FailState}$
if $M \models \neg C$ and M contains no decision literals
- ▶ **backjump** $M I^d N \parallel F, C \implies M I' \parallel F, C$
if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Definition

basic DPLL \mathcal{B} consists of unit propagation, decide, fail, and backjump

Theorem (Termination)

there are *no infinite derivations* $\parallel F \implies_{\mathcal{B}} S_1 \implies_{\mathcal{B}} S_2 \implies_{\mathcal{B}} \dots$

Theorem (Correctness)

for derivation with *final* state S_n :

$$\parallel F \implies_{\mathcal{B}} S_1 \implies_{\mathcal{B}} S_2 \implies_{\mathcal{B}} \dots \implies_{\mathcal{B}} S_n$$

- ▶ if $S_n = \text{FailState}$ then F is *unsatisfiable*
- ▶ if $S_n = M \parallel F'$ then F is *satisfiable* and $M \models F$

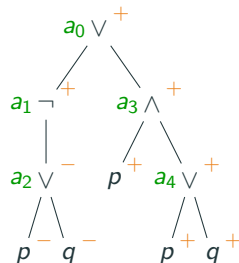
Definition

polarity of subformula φ in ψ is $+$ if number of negations above φ in ψ is even, and $-$ otherwise

Example (Efficient Transformations to CNF)

- ▶ $\varphi = \neg(p \vee q) \vee (p \wedge (p \vee q))$
- ▶ use fresh propositional variable for every connective

$$\begin{array}{ll} a_0: \neg(p \vee q) \vee (p \wedge (p \vee q)) & a_1: \neg(p \vee q) \\ a_2: p \vee q & a_3: p \wedge (p \vee q) \end{array}$$



- ▶ Tseitin: add clause a_0 plus $(a_i \leftrightarrow \dots)$ for every subformula

$$\varphi \approx a_0 \wedge (a_0 \leftrightarrow a_1 \vee a_3) \wedge (a_1 \leftrightarrow \neg a_2) \wedge (a_2 \leftrightarrow p \vee q) \wedge (a_3 \leftrightarrow p \wedge a_4)$$

- ▶ Plaisted & Greenbaum: $(a_i \rightarrow \dots)$ if polarity of a_i is $+$ and $(a_i \leftarrow \dots)$ if $-$

$$\varphi \approx a_0 \wedge (a_0 \rightarrow a_1 \vee a_3) \wedge (a_1 \rightarrow \neg a_2) \wedge (a_2 \leftarrow p \vee q) \wedge (a_3 \rightarrow p \wedge a_4) \wedge (a_4 \rightarrow p \vee q)$$

- ▶ replace \leftrightarrow and \rightarrow by 2 or 3 clauses each

- Summary of Last Week
- From DPLL to Conflict Driven Clause Learning
 - Conflict Analysis
 - Heuristics and Data Structures
- Application: Test Case Generation

Rough DPLL Algorithm

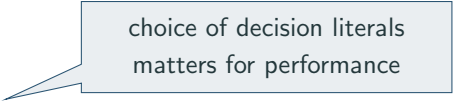
```
function dpll( $\varphi$ )
  M = []
  while (true)
    if all_variables_assigned(M)
      return satisfiable
    M = decide( $\varphi$ , M)
    M = unit_propagate( $\varphi$ , M)
    if (conflict( $\varphi$ , M))
      try
        M = backjump( $\varphi$ , M)

  catch (fail_state)
    return unsatisfiable
```


Rough DPLL Algorithm

```
function dpll( $\varphi$ )
  M = []
  while (true)
    if all_variables_assigned(M)
      return satisfiable
    M = decide( $\varphi$ , M)
    M = unit_propagate( $\varphi$ , M)
    if (conflict( $\varphi$ , M))
      try
        M = backjump( $\varphi$ , M)

      catch (fail_state)
        return unsatisfiable
```



choice of decision literals
matters for performance

Rough DPLL Algorithm

```
function dpll( $\varphi$ )  
  M = []  
  while (true)  
    if all_variables_assigned(M)  
      return satisfiable  
    M = decide( $\varphi$ , M)  
    M = unit_propagate( $\varphi$ , M)  
    if (conflict( $\varphi$ , M))  
      try  
        M = backjump( $\varphi$ , M)  
  
      catch (fail_state)  
        return unsatisfiable
```

choice of decision literals
matters for performance

more than 90% of time
spent in unit propagation

Rough DPLL Algorithm

```
function dpll( $\varphi$ )  
  M = []  
  while (true)  
    if all_variables_assigned(M)  
      return satisfiable  
    M = decide( $\varphi$ , M)  
    M = unit_propagate( $\varphi$ , M)  
    if (conflict( $\varphi$ , M))  
      try  
        M = backjump( $\varphi$ , M)  
  
      catch (fail_state)  
        return unsatisfiable
```

choice of decision literals
matters for performance

more than 90% of time
spent in unit propagation

backjump clauses are useful:
learn them!

Rough DPLL Algorithm

```
function dpll( $\varphi$ )  
  M = []  
  while (true)  
    if all_variables_assigned(M)  
      return satisfiable  
    M = decide( $\varphi$ , M)  
    M = unit_propagate( $\varphi$ , M)  
    if (conflict( $\varphi$ , M))  
      try  
        M, C = backjump( $\varphi$ , M)  
         $\varphi = \varphi \cup \{C\}$   
      catch (fail_state)  
        return unsatisfiable
```

choice of decision literals
matters for performance

more than 90% of time
spent in unit propagation

backjump clauses are useful:
learn them!

Rough DPLL Algorithm

```
function dpll( $\varphi$ )  
  M = []  
  while (true)  
    if all_variables_assigned(M)  
      return satisfiable  
  M = decide( $\varphi$ , M)  
  M = unit_propagate( $\varphi$ , M)  
  if (conflict( $\varphi$ , M))  
    try  
      M, C = backjump( $\varphi$ , M)  
       $\varphi = \varphi \cup \{C\}$   
    catch (fail_state)  
      return unsatisfiable  
   $\varphi = \text{forget}(\varphi)$ 
```

choice of decision literals
matters for performance

more than 90% of time
spent in unit propagation

backjump clauses are useful:
learn them!

forgetting implied clauses
improves performance

Rough DPLL Algorithm

```
function dpll( $\varphi$ )  
  M = []  
  while (true)  
    if all_variables_assigned(M)  
      return satisfiable  
  M = decide( $\varphi$ , M)  
  M = unit_propagate( $\varphi$ , M)  
  if (conflict( $\varphi$ , M))  
    try  
      M, C = backjump( $\varphi$ , M)  
       $\varphi = \varphi \cup \{C\}$   
    catch (fail_state)  
      return unsatisfiable  
   $\varphi = \text{forget}(\varphi)$   
  if (do_restart(M))  
    return dpll( $\varphi$ )
```

choice of decision literals
matters for performance

more than 90% of time
spent in unit propagation

backjump clauses are useful:
learn them!

forgetting implied clauses
improves performance

occasional restarts
improve performance

Conflict Driven Clause Learning (CDCL)

```
function dpll( $\varphi$ )  
  M = []  
  while (true)  
    if all_variables_assigned(M)  
      return satisfiable  
  M = decide( $\varphi$ , M)  
  M = unit_propagate( $\varphi$ , M)  
  if (conflict( $\varphi$ , M))  
    try  
      M, C = backjump( $\varphi$ , M)  
       $\varphi = \varphi \cup \{C\}$   
    catch (fail_state)  
      return unsatisfiable  
   $\varphi$  = forget( $\varphi$ )  
  if (do_restart(M))  
    return dpll( $\varphi$ )
```

choice of decision literals
matters for performance

more than 90% of time
spent in unit propagation

backjump clauses are useful:
learn them!

forgetting implied clauses
improves performance

occasional restarts
improve performance

Definition (CDCL)

CDCL system \mathcal{R} extends DPLL system \mathcal{B} by following three rules:

Definition (CDCL)

CDCL system \mathcal{R} extends DPLL system \mathcal{B} by following three rules:

- **learn**
$$M \parallel F \implies M \parallel F, C$$
 if $F \models C$ and all atoms of C occur in M or F

Definition (CDCL)

CDCL system \mathcal{R} extends DPLL system \mathcal{B} by following three rules:

- ▶ learn $M \parallel F \implies M \parallel F, C$
if $F \models C$ and all atoms of C occur in M or F
- ▶ forget $M \parallel F, C \implies M \parallel F$
if $F \models C$

Definition (CDCL)

CDCL system \mathcal{R} extends DPLL system \mathcal{B} by following three rules:

- ▶ learn $M \parallel F \implies M \parallel F, C$
if $F \models C$ and all atoms of C occur in M or F
- ▶ forget $M \parallel F, C \implies M \parallel F$
if $F \models C$
- ▶ restart $M \parallel F \implies \parallel F$

Theorem (Termination)

any derivation $\parallel F \Longrightarrow_{\mathcal{R}} S_1 \Longrightarrow_{\mathcal{R}} S_2 \Longrightarrow_{\mathcal{R}} \dots$ is finite if

- ▶ it contains *no infinite subderivation of learn and forget steps*, and

Theorem (Termination)

any derivation $\parallel F \Longrightarrow_{\mathcal{R}} S_1 \Longrightarrow_{\mathcal{R}} S_2 \Longrightarrow_{\mathcal{R}} \dots$ is finite if

- ▶ it contains no infinite subderivation of *learn* and *forget* steps, and
- ▶ *restart* is applied with *increasing periodicity*

Theorem (Termination)

any derivation $\parallel F \Longrightarrow_{\mathcal{R}} S_1 \Longrightarrow_{\mathcal{R}} S_2 \Longrightarrow_{\mathcal{R}} \dots$ is finite if

- ▶ it contains no infinite subderivation of *learn* and *forget* steps, and
- ▶ *restart* is applied with increasing periodicity

Theorem (Correctness)

for derivation with final state S_n :

$$\parallel F \Longrightarrow_{\mathcal{R}} S_1 \Longrightarrow_{\mathcal{R}} S_2 \Longrightarrow_{\mathcal{R}} \dots \Longrightarrow_{\mathcal{R}} S_n$$

- ▶ if $S_n = \text{FailState}$ then F is *unsatisfiable*

Theorem (Termination)

any derivation $\parallel F \Longrightarrow_{\mathcal{R}} S_1 \Longrightarrow_{\mathcal{R}} S_2 \Longrightarrow_{\mathcal{R}} \dots$ is finite if

- ▶ it contains no infinite subderivation of *learn* and *forget* steps, and
- ▶ *restart* is applied with increasing periodicity

Theorem (Correctness)

for derivation with final state S_n :

$$\parallel F \Longrightarrow_{\mathcal{R}} S_1 \Longrightarrow_{\mathcal{R}} S_2 \Longrightarrow_{\mathcal{R}} \dots \Longrightarrow_{\mathcal{R}} S_n$$

- ▶ if $S_n = \text{FailState}$ then F is *unsatisfiable*
- ▶ if $S_n = M \parallel F'$ then F is *satisfiable* and $M \models F$

- Summary of Last Week
- From DPLL to Conflict Driven Clause Learning
 - Conflict Analysis
 - Heuristics and Data Structures
- Application: Test Case Generation

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula
- ▶ prefix M of current literal list entails $\neg C'$

(magically detected)

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula (magically detected)
- ▶ prefix M of current literal list entails $\neg C'$

Backjump to Definition

- ▶ **backjump** $M I^d N \parallel F, C \implies M I' \parallel F, C$
if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula (magically detected)
- ▶ prefix M of current literal list entails $\neg C'$

Backjump to Definition

- ▶ backjump $M I^d N \parallel F, C \implies M I' \parallel F, C$
if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Example

$1^d 2 \quad 3^d \quad 4^d \bar{5} \parallel \bar{1} \vee 2, \bar{1} \vee \bar{3} \vee 4 \vee 5, \bar{2} \vee \bar{4} \vee \bar{5}, 4 \vee \bar{5}, \bar{4} \vee 5, \bar{1} \vee \bar{5} \vee 6, \bar{2} \vee \bar{5} \vee \bar{6}$

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula (magically detected)
- ▶ prefix M of current literal list entails $\neg C'$

Backjump to Definition

- ▶ backjump $M I^d N \parallel F, C \implies M I' \parallel F, C$
 if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Example

$$\underbrace{1^d 2}_M \underbrace{3^d}_I \underbrace{4^d \bar{5}}_N \parallel \underbrace{\bar{1} \vee 2, \bar{1} \vee \bar{3} \vee 4 \vee 5, \bar{2} \vee \bar{4} \vee \bar{5}, 4 \vee \bar{5}, \bar{4} \vee 5, \bar{1} \vee \bar{5} \vee 6, \bar{2} \vee \bar{5} \vee \bar{6}}_{F, C}$$

$$M = 1^d 2 \quad I = 3 \quad N = 4^d \bar{5}$$

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula (magically detected)
- ▶ prefix M of current literal list entails $\neg C'$

Backjump to Definition

- ▶ backjump $M I^d N \parallel F, C \implies M I' \parallel F, C$
 if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Example

$$\underbrace{1^d 2}_M \underbrace{3^d}_I \underbrace{4^d \bar{5}}_N \parallel \underbrace{\bar{1} \vee 2, \bar{1} \vee \bar{3} \vee 4 \vee 5, \bar{2} \vee \bar{4} \vee \bar{5}, 4 \vee \bar{5}, \bar{4} \vee 5, \bar{1} \vee \bar{5} \vee 6, \bar{2} \vee \bar{5} \vee \bar{6}}_{F, C}$$

$$M = 1^d 2 \quad I = 3 \quad N = 4^d \bar{5} \quad C = \bar{4} \vee 5$$

- ▶ $1^d 2 3^d 4^d \bar{5} \models \neg(\bar{4} \vee 5)$

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula (magically detected)
- ▶ prefix M of current literal list entails $\neg C'$

Backjump to Definition

- ▶ backjump $M I^d N \parallel F, C \implies M I' \parallel F, C$
 if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Example

$$\underbrace{1^d 2}_M \underbrace{3^d}_I \underbrace{4^d \bar{5}}_N \parallel \underbrace{\bar{1} \vee 2, \bar{1} \vee \bar{3} \vee 4 \vee 5, \bar{2} \vee \bar{4} \vee \bar{5}, 4 \vee \bar{5}, \bar{4} \vee 5, \bar{1} \vee \bar{5} \vee 6, \bar{2} \vee \bar{5} \vee \bar{6}}_{F, C}$$

$$M = 1^d 2 \quad I = 3 \quad N = 4^d \bar{5} \quad C = \bar{4} \vee 5 \quad C' = \bar{1} \quad I' = \bar{5}$$

- ▶ $1^d 2 3^d 4^d \bar{5} \models \neg(\bar{4} \vee 5)$
- ▶ backjump clause $C' \vee I' = \bar{1} \vee \bar{5}$ satisfies $F, C \models C' \vee I'$

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula (magically detected)
- ▶ prefix M of current literal list entails $\neg C'$

Backjump to Definition

- ▶ backjump $M I^d N \parallel F, C \implies M I' \parallel F, C$
 if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Example

$$\underbrace{1^d 2}_M \underbrace{3^d}_I \underbrace{4^d \bar{5}}_N \parallel \underbrace{\bar{1} \vee 2, \bar{1} \vee \bar{3} \vee 4 \vee 5, \bar{2} \vee \bar{4} \vee \bar{5}, 4 \vee \bar{5}, \bar{4} \vee 5, \bar{1} \vee \bar{5} \vee 6, \bar{2} \vee \bar{5} \vee \bar{6}}_{F, C}$$

$$M = 1^d 2 \quad I = 3 \quad N = 4^d \bar{5} \quad C = \bar{4} \vee 5 \quad C' = \bar{1} \quad I' = \bar{5}$$

- ▶ $1^d 2 3^d 4^d \bar{5} \models \neg(\bar{4} \vee 5)$
- ▶ backjump clause $C' \vee I' = \bar{1} \vee \bar{5}$ satisfies $F, C \models C' \vee I'$
- ▶ $1^d 2 \models 1$

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula (magically detected)
- ▶ prefix M of current literal list entails $\neg C'$

Backjump to Definition

- ▶ backjump $M I^d N \parallel F, C \implies M I' \parallel F, C$
 if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Example

$$\underbrace{1^d 2}_M \underbrace{3^d}_I \underbrace{4^d \bar{5}}_N \parallel \underbrace{\bar{1} \vee 2, \bar{1} \vee \bar{3} \vee 4 \vee 5, \bar{2} \vee \bar{4} \vee \bar{5}, 4 \vee \bar{5}, 4 \vee 5, \bar{1} \vee \bar{5} \vee 6, \bar{2} \vee \bar{5} \vee \bar{6}}_{F, C}$$

$$M = 1^d 2 \quad I = 3 \quad N = 4^d \bar{5} \quad C = \bar{4} \vee 5 \quad C' = \bar{1} \quad I' = \bar{5}$$

- ▶ $1^d 2 3^d 4^d \bar{5} \models \neg(\bar{4} \vee 5)$
- ▶ backjump clause $C' \vee I' = \bar{1} \vee \bar{5}$ satisfies $F, C \models C' \vee I'$
- ▶ $1^d 2 \models 1$, and 5 is undefined in $1^d 2$ but occurs in F

Backjump: Idea

- ▶ backjump clause $C' \vee I'$ is entailed by formula (magically detected)
- ▶ prefix M of current literal list entails $\neg C'$

Backjump to Definition

- ▶ backjump $M I^d N \parallel F, C \implies M I' \parallel F, C$
 if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models C' \vee I'$ backjump clause
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$

Example

$$1^d 2 \quad 3^d \quad 4^d \bar{5} \parallel \bar{1} \vee 2, \bar{1} \vee \bar{3} \vee 4 \vee 5, \bar{2} \vee \bar{4} \vee \bar{5}, 4 \vee \bar{5}, \bar{4} \vee 5, \bar{1} \vee \bar{5} \vee 6, \bar{2} \vee \bar{5} \vee \bar{6}$$

$$\implies 1^d 2 \bar{5} \parallel \bar{1} \vee 2, \bar{1} \vee \bar{3} \vee 4 \vee 5, \bar{2} \vee \bar{4} \vee \bar{5}, 4 \vee \bar{5}, \bar{4} \vee 5, \bar{1} \vee \bar{5} \vee 6, \bar{2} \vee \bar{5} \vee \bar{6}$$

$$M = 1^d 2 \quad I = 3 \quad N = 4^d \bar{5} \quad C = \bar{4} \vee 5 \quad C' = \bar{1} \quad I' = \bar{5}$$

- ▶ $1^d 2 3^d 4^d \bar{5} \models \neg(\bar{4} \vee 5)$
- ▶ backjump clause $C' \vee I' = \bar{1} \vee \bar{5}$ satisfies $F, C \models C' \vee I'$
- ▶ $1^d 2 \models 1$, and 5 is undefined in $1^d 2$ but occurs in F

- Summary of Last Week
- From DPLL to Conflict Driven Clause Learning
 - Conflict Analysis
 - Heuristics and Data Structures
- Application: Test Case Generation

Desirable Properties of Backjump Clauses

- ▶ small
- ▶ should trigger progress

How to Determine Backjump Clauses?

- ▶ implication graph
- ▶ resolution

Example: Implication Graph

$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge \\ (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$

decisions



Example: Implication Graph

$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge \\ (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$

decisions

1^d

level	literal	reason
1	1	decision

Example: Implication Graph

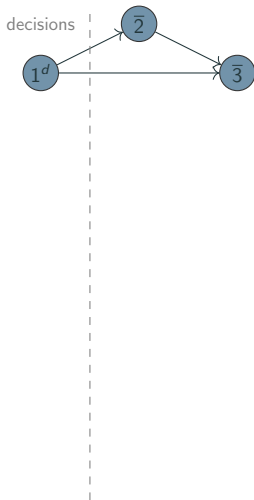
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge \\ (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$

Example: Implication Graph

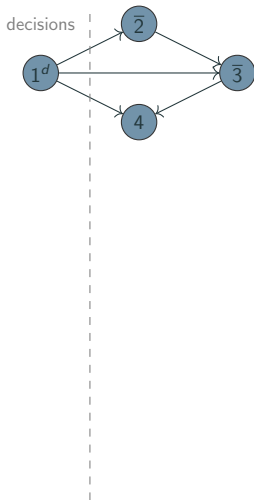
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$

Example: Implication Graph

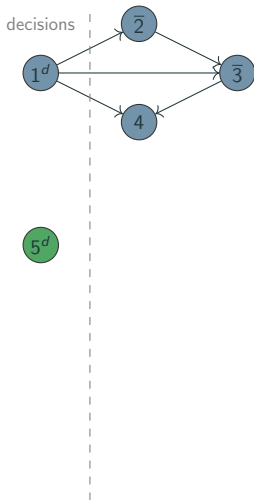
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$

Example: Implication Graph

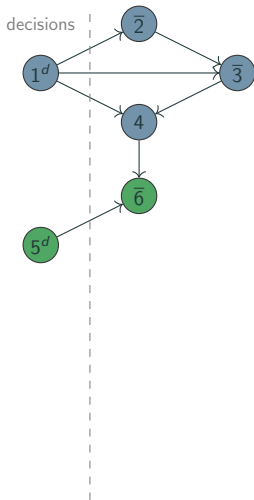
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision

Example: Implication Graph

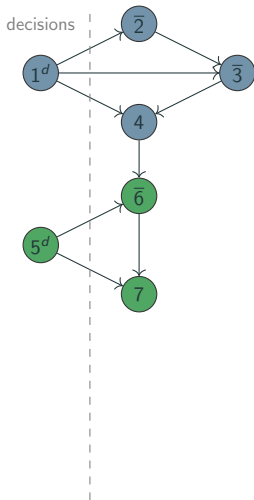
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$

Example: Implication Graph

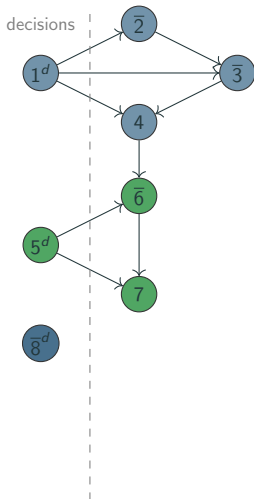
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee 9 \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$

Example: Implication Graph

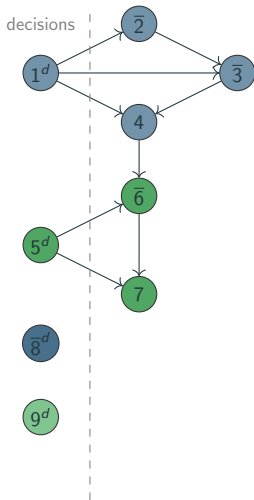
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision

Example: Implication Graph

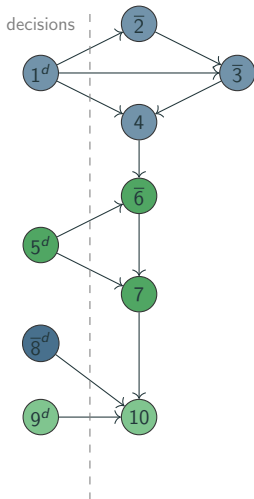
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision

Example: Implication Graph

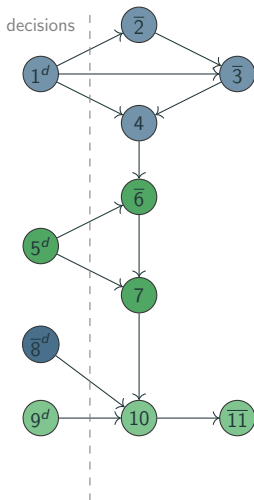
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$

Example: Implication Graph

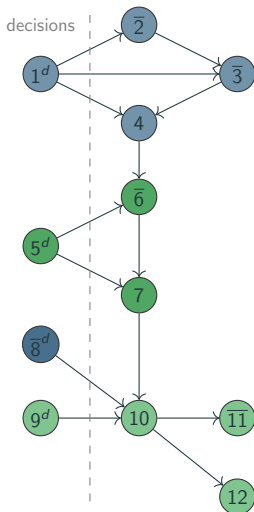
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$
	$\bar{11}$	$\bar{10} \vee \bar{11}$

Example: Implication Graph

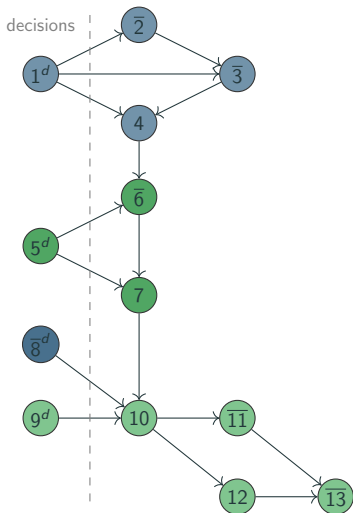
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$
	$\bar{11}$	$\bar{10} \vee \bar{11}$
	12	$\bar{10} \vee 12$

Example: Implication Graph

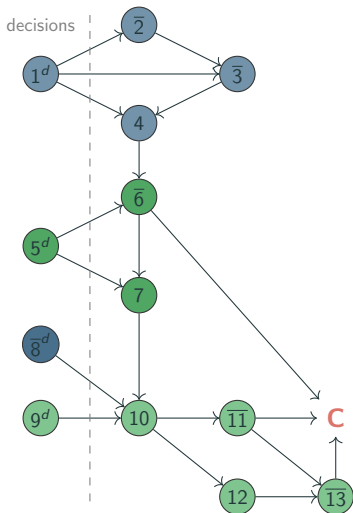
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$
	$\bar{11}$	$\bar{10} \vee \bar{11}$
	12	$\bar{10} \vee 12$
	$\bar{13}$	$\bar{12} \vee \bar{13}$

Example: Implication Graph

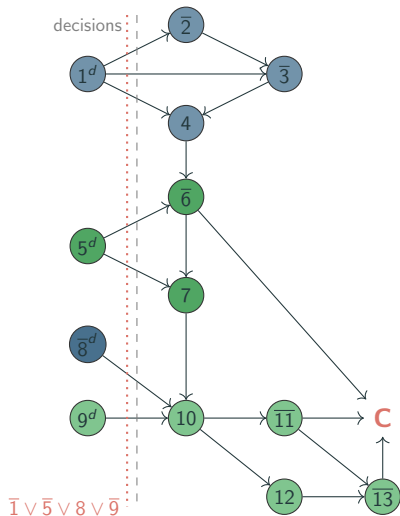
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$
	$\bar{11}$	$\bar{10} \vee \bar{11}$
	12	$\bar{10} \vee 12$
	$\bar{13}$	$\bar{12} \vee \bar{13}$

Example: Implication Graph

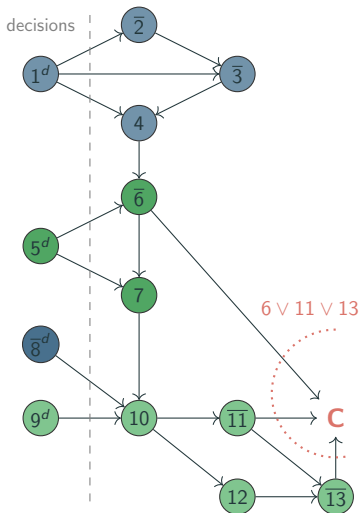
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$
	$\bar{11}$	$\bar{10} \vee \bar{11}$
	12	$\bar{10} \vee 12$
	$\bar{13}$	$\bar{12} \vee \bar{13}$

Example: Implication Graph

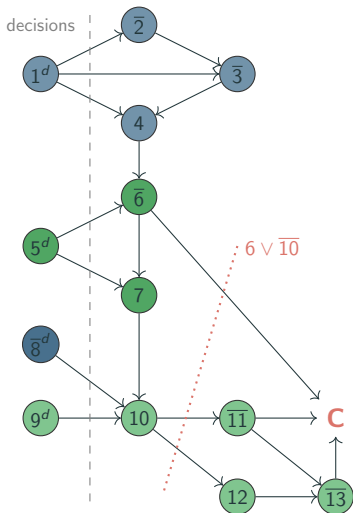
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$
	$\bar{11}$	$\bar{10} \vee \bar{11}$
	12	$\bar{10} \vee 12$
	$\bar{13}$	$\bar{12} \vee \bar{13}$

Example: Implication Graph

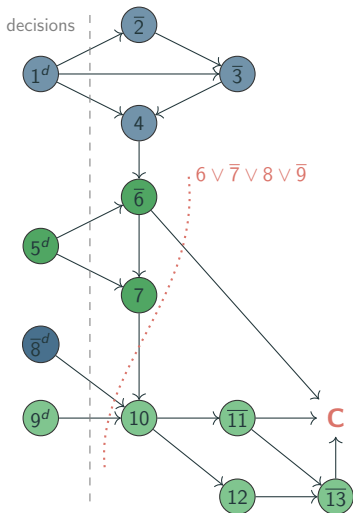
$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$
	$\bar{11}$	$\bar{10} \vee \bar{11}$
	12	$\bar{10} \vee 12$
	$\bar{13}$	$\bar{12} \vee \bar{13}$

Example: Implication Graph

$$\varphi = (\bar{1} \vee \bar{2}) \wedge (\bar{1} \vee 2 \vee \bar{3}) \wedge (\bar{1} \vee 3 \vee 4) \wedge (\bar{4} \vee \bar{5} \vee \bar{6}) \wedge (\bar{5} \vee 6 \vee 7) \wedge (\bar{7} \vee 8 \vee \bar{9} \vee 10) \wedge (\bar{10} \vee \bar{11}) \wedge (\bar{10} \vee 12) \wedge (\bar{12} \vee \bar{13}) \wedge (6 \vee 11 \vee 13)$$



level	literal	reason
1	1	decision
	$\bar{2}$	$\bar{1} \vee \bar{2}$
	$\bar{3}$	$\bar{1} \vee 2 \vee \bar{3}$
	4	$\bar{1} \vee 3 \vee 4$
2	5	decision
	$\bar{6}$	$\bar{4} \vee \bar{5} \vee \bar{6}$
	7	$\bar{5} \vee 6 \vee 7$
3	$\bar{8}$	decision
4	9	decision
	10	$\bar{7} \vee 8 \vee \bar{9} \vee 10$
	$\bar{11}$	$\bar{10} \vee \bar{11}$
	12	$\bar{10} \vee 12$
	$\bar{13}$	$\bar{12} \vee \bar{13}$

What to Learn from That?

Definitions

- ▶ **cut** of implication graph has at least all decision literals on the left, and at least the conflict node on the right

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

Example

- ▶ cuts: $\bar{1} \vee \bar{5} \vee 8 \vee \bar{9}$

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

Example

- ▶ cuts: $\bar{1} \vee \bar{5} \vee 8 \vee \bar{9} \quad 6 \vee 11 \vee 13$

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

Example

- ▶ cuts: $\bar{1} \vee \bar{5} \vee 8 \vee \bar{9}$ $6 \vee 11 \vee 13$ $6 \vee \bar{10}$

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

Example

- ▶ cuts: $\bar{1} \vee \bar{5} \vee 8 \vee \bar{9}$ $6 \vee 11 \vee 13$ $6 \vee \bar{10}$ $6 \vee \bar{7} \vee 8 \vee \bar{9}$

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right
- ▶ literal l in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through l

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right
- ▶ literal l in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through l

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

Example

- ▶ UIPs are 9 and 10

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right
- ▶ literal l in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through l
- ▶ **first UIP** is UIP closest to conflict node

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

Example

- ▶ UIPs are 9 and 10

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right
- ▶ literal l in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through l
- ▶ **first UIP** is UIP closest to conflict node

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause

Example

- ▶ UIPs are 9 and 10
- ▶ first UIP is 10

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right
- ▶ literal l in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through l
- ▶ **first UIP** is UIP closest to conflict node

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause
- ▶ last decision literal is UIP

Example

- ▶ UIPs are 9 and 10
- ▶ first UIP is 10

What to Learn from That?

Definitions

- ▶ cut of implication graph has at least all decision literals on the left, and at least the conflict node on the right
- ▶ literal l in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through l
- ▶ **first UIP** is UIP closest to conflict node

Key Observations

- ▶ if $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ are cut edges then $l_1^c \vee \dots \vee l_k^c$ is entailed clause
- ▶ last decision literal is UIP
- ▶ backjump clause: cut with exactly **one** literal l at last decision level (l is UIP)

Example

- ▶ UIPs are 9 and 10
- ▶ first UIP is 10

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- add node labelled l for every decision literal l in M

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F such that there are already nodes l_1^c, \dots, l_m^c

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F such that there are already nodes l_1^c, \dots, l_m^c
 - ▶ add node l' if not yet present

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies^*_B M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F such that there are already nodes l_1^c, \dots, l_m^c
 - ▶ add node l' if not yet present
 - ▶ add edges $l_i^c \rightarrow l'$ for all $1 \leq i \leq m$ if not yet present

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F such that there are already nodes l_1^c, \dots, l_m^c
 - ▶ add node l' if not yet present
 - ▶ add edges $l_i^c \rightarrow l'$ for all $1 \leq i \leq m$ if not yet present
 - ▶ if \exists clause $l'_1 \vee \dots \vee l'_k$ in F such that there are nodes l_1^c, \dots, l_k^c

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F such that there are already nodes l_1^c, \dots, l_m^c
 - ▶ add node l' if not yet present
 - ▶ add edges $l_i^c \rightarrow l'$ for all $1 \leq i \leq m$ if not yet present
 - if \exists clause $l'_1 \vee \dots \vee l'_k$ in F such that there are nodes l_1^c, \dots, l_k^c
 - ▶ add **conflict node** labeled C

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F such that there are already nodes l_1^c, \dots, l_m^c
 - ▶ add node l' if not yet present
 - ▶ add edges $l_i^c \rightarrow l'$ for all $1 \leq i \leq m$ if not yet present
 - if \exists clause $l'_1 \vee \dots \vee l'_k$ in F such that there are nodes $l_1'^c, \dots, l_k'^c$
 - ▶ add conflict node labeled C
 - ▶ add edges $l_i'^c \rightarrow C$

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F such that there are already nodes l_1^c, \dots, l_m^c
 - ▶ add node l' if not yet present
 - ▶ add edges $l_i^c \rightarrow l'$ for all $1 \leq i \leq m$ if not yet present
 - if \exists clause $l'_1 \vee \dots \vee l'_k$ in F such that there are nodes l_1^c, \dots, l_k^c
 - ▶ add conflict node labeled C
 - ▶ add edges $l_i^c \rightarrow C$

Lemma

if edges intersected by cut are $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ then $F \models l_1^c \vee \dots \vee l_k^c$

Definition (Implication Graph)

Consider DPLL derivation to $\parallel F \implies_{\mathcal{B}}^* M \parallel F$.

Implication graph is a directed acyclic graph constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F such that there are already nodes l_1^c, \dots, l_m^c
 - ▶ add node l' if not yet present
 - ▶ add edges $l_i^c \rightarrow l'$ for all $1 \leq i \leq m$ if not yet present
 - if \exists clause $l'_1 \vee \dots \vee l'_k$ in F such that there are nodes l_1^c, \dots, l_k^c
 - ▶ add conflict node labeled C
 - ▶ add edges $l_i^c \rightarrow C$

potential backjump clause

Lemma

if edges intersected by cut are $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ then $F \models l_1^c \vee \dots \vee l_k^c$

Remarks

- ▶ keeping track of implication graph is too expensive in practice
- ▶ compute clauses associated with cuts by resolution instead

Resolution

Remarks

- ▶ keeping track of implication graph is too expensive in practice
- ▶ compute clauses associated with cuts by resolution instead

Definition (Resolution)

$$\frac{C \vee I \quad C' \vee \neg I}{C \vee C'}$$

(assuming literals in clauses can be reordered)

Resolution

Remarks

- ▶ keeping track of implication graph is too expensive in practice
- ▶ compute clauses associated with cuts by resolution instead

Definition (Resolution)

$$\frac{C \vee I \quad C' \vee \neg I}{C \vee C'}$$

(assuming literals in clauses can be reordered)

Example

$$\frac{6 \vee 11 \vee 13 \quad \overline{12} \vee \overline{13}}{6 \vee 11 \vee \overline{12}}$$

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to **cut** in implication graph

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

- ▶ $C_0 = 6 \vee 11 \vee 13$

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

$$\begin{array}{lcl} \text{▶ } C_0 = 6 \vee 11 \vee 13 & 6 \vee 11 \vee 13 & \overline{12} \vee 13 \\ \hline & 6 \vee 11 \vee \overline{12} \end{array}$$

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

- ▶ $C_0 = 6 \vee 11 \vee 13$
 - ▶ $C_1 = 6 \vee 11 \vee \overline{12}$
- $$\frac{6 \vee 11 \vee 13 \quad \overline{12} \vee \overline{13}}{6 \vee 11 \vee \overline{12}}$$

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

- ▶ $C_0 = 6 \vee 11 \vee 13$
 - ▶ $C_1 = 6 \vee 11 \vee \overline{12}$
- | | |
|--------------------------------|------------------------------------|
| $6 \vee 11 \vee 13$ | $\overline{12} \vee \overline{13}$ |
| <hr/> | |
| $6 \vee 11 \vee \overline{12}$ | $\overline{10} \vee 12$ |
| <hr/> | |
| $6 \vee 11 \vee \overline{10}$ | |

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

- ▶ $C_0 = 6 \vee 11 \vee 13$
 - ▶ $C_1 = 6 \vee 11 \vee \overline{12}$
 - ▶ $C_2 = 6 \vee 11 \vee \overline{10}$
- | | |
|--------------------------------|------------------------------------|
| $6 \vee 11 \vee 13$ | $\overline{12} \vee \overline{13}$ |
| <hr/> | |
| $6 \vee 11 \vee \overline{12}$ | $\overline{10} \vee 12$ |
| <hr/> | |
| $6 \vee 11 \vee \overline{10}$ | |

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

- ▶ $C_0 = 6 \vee 11 \vee 13$
 - ▶ $C_1 = 6 \vee 11 \vee \overline{12}$
 - ▶ $C_2 = 6 \vee \boxed{11} \vee \overline{10}$
- | | |
|--|--|
| $6 \vee 11 \vee 13$ | $\overline{12} \vee \overline{13}$ |
| <hr/> | |
| $6 \vee 11 \vee \overline{12}$ | $\overline{10} \vee 12$ |
| <hr/> | |
| $6 \vee \boxed{11} \vee \overline{10}$ | $\overline{10} \vee \boxed{\overline{11}}$ |
| <hr/> | |
| $6 \vee \overline{10}$ | |

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

- ▶ $C_0 = 6 \vee 11 \vee 13$
 - ▶ $C_1 = 6 \vee 11 \vee \overline{12}$
 - ▶ $C_2 = 6 \vee 11 \vee \overline{10}$
 - ▶ $C_3 = 6 \vee \overline{10}$
- | | |
|--------------------------------|------------------------------------|
| $6 \vee 11 \vee 13$ | $\overline{12} \vee \overline{13}$ |
| <hr/> | |
| $6 \vee 11 \vee \overline{12}$ | $\overline{10} \vee 12$ |
| <hr/> | |
| $6 \vee 11 \vee \overline{10}$ | $\overline{10} \vee \overline{11}$ |
| <hr/> | |
| $6 \vee \overline{10}$ | |

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

- ▶ $C_0 = 6 \vee 11 \vee 13$
 - ▶ $C_1 = 6 \vee 11 \vee \overline{12}$
 - ▶ $C_2 = 6 \vee 11 \vee \overline{10}$
 - ▶ $C_3 = 6 \vee \overline{10}$
- | | | |
|--|------------------------------------|---|
| $6 \vee 11 \vee 13$ | $\overline{12} \vee \overline{13}$ | |
| <hr/> | | |
| $6 \vee 11 \vee \overline{12}$ | $\overline{10} \vee 12$ | |
| <hr/> | | |
| $6 \vee 11 \vee \overline{10}$ | $\overline{10} \vee \overline{11}$ | |
| <hr/> | | |
| $6 \vee \overline{10}$ | | $\overline{7} \vee 8 \vee \overline{9} \vee 10$ |
| <hr/> | | |
| $6 \vee \overline{7} \vee 8 \vee \overline{9}$ | | |

How to Derive Backjump Clause by Resolution

- ▶ let C_0 be the conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

Example

- ▶ $C_0 = 6 \vee 11 \vee 13$
 - ▶ $C_1 = 6 \vee 11 \vee \overline{12}$
 - ▶ $C_2 = 6 \vee 11 \vee \overline{10}$
 - ▶ $C_3 = 6 \vee \overline{10}$
 - ▶ $C_4 = 6 \vee \overline{7} \vee 8 \vee \overline{9}$
- $$\begin{array}{rcl}
 6 \vee 11 \vee 13 & & \overline{12} \vee \overline{13} \\
 \hline
 6 \vee 11 \vee \overline{12} & & \overline{10} \vee 12 \\
 \hline
 6 \vee 11 \vee \overline{10} & & \overline{10} \vee \overline{11} \\
 \hline
 6 \vee \overline{10} & & \overline{7} \vee 8 \vee \overline{9} \vee 10 \\
 \hline
 6 \vee \overline{7} \vee 8 \vee \overline{9}
 \end{array}$$

Observations

- ▶ choice of next decision variable is critical
- ▶ prefer variables that participated in recent conflict

Observations

- ▶ choice of next decision variable is critical
- ▶ prefer variables that participated in recent conflict

VSIDS: Variable State Independent Decaying Sum

- ▶ first presented in SAT solver [Chaff](#) (2001)
- ▶ variant of this heuristic nowadays implemented in most CDCL solvers
- ▶ compute score for each variable, select variable with highest score

Decision Variable Selection

Observations

- ▶ choice of next decision variable is critical
- ▶ prefer variables that participated in recent conflict

VSIDS: Variable State Independent Decaying Sum

- ▶ first presented in SAT solver [Chaff](#) (2001)
- ▶ variant of this heuristic nowadays implemented in most CDCL solvers
- ▶ compute score for each variable, select variable with highest score
 - ▶ initial variable score is **number of literal occurrences**

Decision Variable Selection

Observations

- ▶ choice of next decision variable is critical
- ▶ prefer variables that participated in recent conflict

VSIDS: Variable State Independent Decaying Sum

- ▶ first presented in SAT solver [Chaff](#) (2001)
- ▶ variant of this heuristic nowadays implemented in most CDCL solvers
- ▶ compute score for each variable, select variable with highest score
 - ▶ initial variable score is number of literal occurrences
 - ▶ learned (conflict) clause C : **increment** score for all variables in C

Decision Variable Selection

Observations

- ▶ choice of next decision variable is critical
- ▶ prefer variables that participated in recent conflict

VSIDS: Variable State Independent Decaying Sum

- ▶ first presented in SAT solver [Chaff](#) (2001)
- ▶ variant of this heuristic nowadays implemented in most CDCL solvers
- ▶ compute score for each variable, select variable with highest score
 - ▶ initial variable score is number of literal occurrences
 - ▶ learned (conflict) clause C : increment score for all variables in C
 - ▶ **periodically divide** all scores by constant

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

Example (VSIDS)

$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \quad \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \quad \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 4^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 4^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow^* 3^d 1 \bar{4} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}$$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 4^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow^* 3^d 1 \bar{4} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}$$

after adding learned clause: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 6, 4 \mapsto 5, 5 \mapsto 2\}$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 4^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow^* 3^d 1 \bar{4} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}$$

after adding learned clause: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 6, 4 \mapsto 5, 5 \mapsto 2\}$

division by 2: $\{1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto \frac{5}{2}, 5 \mapsto 1\}$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 4^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow^* 3^d 1 \bar{4} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}$$

after adding learned clause: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 6, 4 \mapsto 5, 5 \mapsto 2\}$

division by 2: $\{1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto \frac{5}{2}, 5 \mapsto 1\}$

$$\Rightarrow^* \bar{3} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}, \bar{1} \vee \bar{3} \vee 4$$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 14^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow^* 3^d 1\bar{4} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}$$

after adding learned clause: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 6, 4 \mapsto 5, 5 \mapsto 2\}$

division by 2: $\{1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto \frac{5}{2}, 5 \mapsto 1\}$

$$\Rightarrow^* \bar{3} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}, \bar{1} \vee \bar{3} \vee 4$$

after adding learned clause: $\{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 4, 4 \mapsto \frac{7}{2}, 5 \mapsto 1\}$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 14^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow^* 3^d 1\bar{4} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}$$

after adding learned clause: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 6, 4 \mapsto 5, 5 \mapsto 2\}$

division by 2: $\{1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto \frac{5}{2}, 5 \mapsto 1\}$

$$\Rightarrow^* \bar{3} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}, \bar{1} \vee \bar{3} \vee 4$$

after adding learned clause: $\{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 4, 4 \mapsto \frac{7}{2}, 5 \mapsto 1\}$

$$\Rightarrow^* \bar{3} 2 4^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}, \bar{1} \vee \bar{3} \vee 4$$

Example (VSIDS)

$$\parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

initial scores: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 5, 4 \mapsto 4, 5 \mapsto 2\}$

$$\Rightarrow 3^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow 3^d 1 4^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}$$

$$\Rightarrow^* 3^d 1 \bar{4} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}$$

after adding learned clause: $\{1 \mapsto 4, 2 \mapsto 4, 3 \mapsto 6, 4 \mapsto 5, 5 \mapsto 2\}$

division by 2: $\{1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto \frac{5}{2}, 5 \mapsto 1\}$

$$\Rightarrow^* \bar{3} \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}, \bar{1} \vee \bar{3} \vee 4$$

after adding learned clause: $\{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 4, 4 \mapsto \frac{7}{2}, 5 \mapsto 1\}$

$$\Rightarrow^* \bar{3} 2 4^d \parallel 1 \vee \bar{2}, 2 \vee \bar{3} \vee 4, \bar{1} \vee 4, \bar{4} \vee 3 \vee 5, 3 \vee \bar{5}, \bar{3} \vee 1, \bar{1} \vee \bar{2}, 2 \vee 3, \bar{4} \vee \bar{5}, \bar{4} \vee \bar{3}, \bar{1} \vee \bar{3} \vee 4$$

\Rightarrow^* FailState

Efficient Unit Propagation?

Suppose input formula φ has n clauses and m literals in total.

Unit propagation in practice

- ▶ each unit propagation step requires to traverse entire formula φ
- ▶ takes 90% of computation time when implemented naively

$\mathcal{O}(m)$

Efficient Unit Propagation?

Suppose input formula φ has n clauses and m literals in total.

Unit propagation in practice

- ▶ each unit propagation step requires to traverse entire formula φ $\mathcal{O}(m)$
- ▶ takes 90% of computation time when implemented naively

Observation

at any point of DPLL run, literal in clause is either **true**, **false**, or **unassigned**

Efficient Unit Propagation?

Suppose input formula φ has n clauses and m literals in total.

Unit propagation in practice

- ▶ each unit propagation step requires to traverse entire formula φ $\mathcal{O}(m)$
- ▶ takes 90% of computation time when implemented naively

Observation

at any point of DPLL run, literal in clause is either true, false, or unassigned

First idea

- ▶ maintain **counter** how many **false literals** are in every clause C

Efficient Unit Propagation?

Suppose input formula φ has n clauses and m literals in total.

Unit propagation in practice

- ▶ each unit propagation step requires to traverse entire formula φ $\mathcal{O}(m)$
- ▶ takes 90% of computation time when implemented naively

Observation

at any point of DPLL run, literal in clause is either true, false, or unassigned

First idea

- ▶ maintain counter how many false literals are in every clause C
- ▶ when **assigning false** to literal in clause, **increment** counter

Efficient Unit Propagation?

Suppose input formula φ has n clauses and m literals in total.

Unit propagation in practice

- ▶ each unit propagation step requires to traverse entire formula φ $\mathcal{O}(m)$
- ▶ takes 90% of computation time when implemented naively

Observation

at any point of DPLL run, literal in clause is either true, false, or unassigned

First idea

- ▶ maintain counter how many false literals are in every clause C
- ▶ when assigning false to literal in clause, increment counter
- ▶ if counter is $|C| - 1$ and remaining literal unassigned, unit propagate $\mathcal{O}(n)$

Efficient Unit Propagation?

Suppose input formula φ has n clauses and m literals in total.

Unit propagation in practice

- ▶ each unit propagation step requires to traverse entire formula φ $\mathcal{O}(m)$
- ▶ takes 90% of computation time when implemented naively

Observation

at any point of DPLL run, literal in clause is either true, false, or unassigned

First idea

- ▶ maintain counter how many false literals are in every clause C
- ▶ when assigning false to literal in clause, increment counter
- ▶ if counter is $|C| - 1$ and remaining literal unassigned, unit propagate $\mathcal{O}(n)$

Drawbacks

- ▶ upon backjump, must adjust all counters
- ▶ overhead to adjust counter if not yet $|C| - 1$

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause C

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause C
- ▶ each pointer points to a literal in the clause that is:
unassigned or true if possible, otherwise false

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause C
- ▶ each pointer points to a literal in the clause that is: unassigned or true if possible, otherwise false
- ▶ ensure invariant that $p_1(C) \neq p_2(C)$

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause
- ▶ each pointer points to a literal in the clause that is unassigned or true if possible, otherwise false
- ▶ ensure invariant that $p_1(C) \neq p_2(C)$

assume that preprocessing eliminates singleton clauses

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause
- ▶ each pointer points to a literal in the clause that is unassigned or true if possible, otherwise false
- ▶ ensure invariant that $p_1(C) \neq p_2(C)$

assume that preprocessing eliminates singleton clauses

Key properties

- ▶ clause C enables **unit propagation** if $p_1(C)$ is false and $p_2(C)$ is unassigned literal
or vice versa

$O(n)$

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause
- ▶ each pointer points to a literal in the clause that is unassigned or true if possible, otherwise false
- ▶ ensure invariant that $p_1(C) \neq p_2(C)$

assume that preprocessing eliminates singleton clauses

Key properties

- ▶ clause C enables unit propagation if $p_1(C)$ is false and $p_2(C)$ is unassigned literal
or vice versa
- ▶ clause C is **conflict clause** if $p_1(C)$ and $p_2(C)$ are false literals

$\mathcal{O}(n)$

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause
- ▶ each pointer points to a literal in the clause that is unassigned or true if possible, otherwise false
- ▶ ensure invariant that $p_1(C) \neq p_2(C)$

assume that preprocessing eliminates singleton clauses

Key properties

- ▶ clause C enables unit propagation if $p_1(C)$ is false and $p_2(C)$ is unassigned literal
or vice versa $\mathcal{O}(n)$
- ▶ clause C is conflict clause if $p_1(C)$ and $p_2(C)$ are false literals

Setting pointers

- ▶ **initialization**: set p_1 and p_2 to different (unassigned) literals in clause

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause
- ▶ each pointer points to a literal in the clause that is unassigned or true if possible, otherwise false
- ▶ ensure invariant that $p_1(C) \neq p_2(C)$

assume that preprocessing eliminates singleton clauses

Key properties

- ▶ clause C enables unit propagation if $p_1(C)$ is false and $p_2(C)$ is unassigned literal
or vice versa
- ▶ clause C is conflict clause if $p_1(C)$ and $p_2(C)$ are false literals

$\mathcal{O}(n)$

Setting pointers

- ▶ **initialization**: set p_1 and p_2 to different (unassigned) literals in clause
- ▶ **assigning variables** by decide or unit propagate:
when assigning literal l true, redirect all pointers to l^c to other literal in their clause if possible

Two-Watched Literal Scheme

Idea

- ▶ maintain two pointers p_1 and p_2 for each clause
- ▶ each pointer points to a literal in the clause that is unassigned or true if possible, otherwise false
- ▶ ensure invariant that $p_1(C) \neq p_2(C)$

assume that preprocessing eliminates singleton clauses

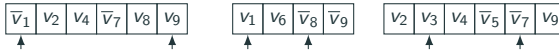
Key properties

- ▶ clause C enables unit propagation if $p_1(C)$ is false and $p_2(C)$ is unassigned literal
or vice versa $\mathcal{O}(n)$
- ▶ clause C is conflict clause if $p_1(C)$ and $p_2(C)$ are false literals

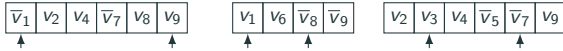
Setting pointers

- ▶ initialization: set p_1 and p_2 to different (unassigned) literals in clause
- ▶ assigning variables by decide or unit propagate:
when assigning literal l true, redirect all pointers to l^c to other literal in their clause if possible
- ▶ **backjump**: no need to change pointers!

Example (Two-Watched literal scheme)

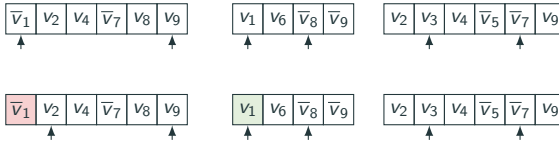


Example (Two-Watched literal scheme)



Example (Two-Watched literal scheme)

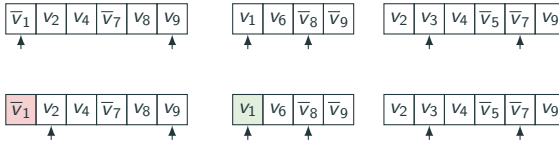
$v_1 \mapsto T$ ↓



Example (Two-Watched literal scheme)

$v_1 \mapsto T$
 $v_9 \mapsto F$
 $v_7 \mapsto T$
 $v_4 \mapsto F$

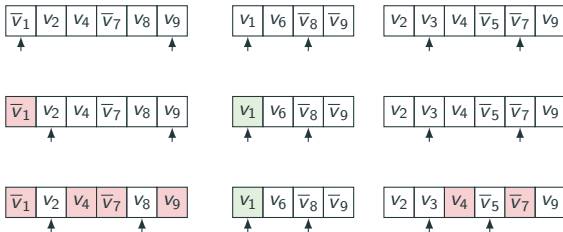
\downarrow
 \downarrow^*



Example (Two-Watched literal scheme)

$v_1 \mapsto T$ ↓

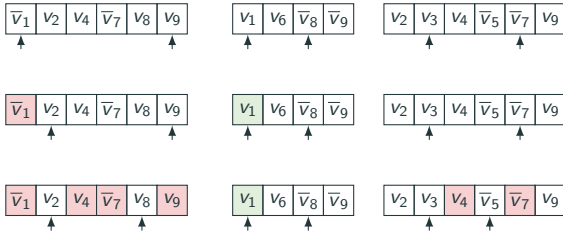
$v_9 \mapsto F$
 $v_7 \mapsto T$ * ↓
 $v_4 \mapsto F$



Example (Two-Watched literal scheme)

$v_1 \mapsto T$ ↓
 $v_9 \mapsto F$
 $v_7 \mapsto T$ * ↓
 $v_4 \mapsto F$

 $v_2 \mapsto F$ ↓

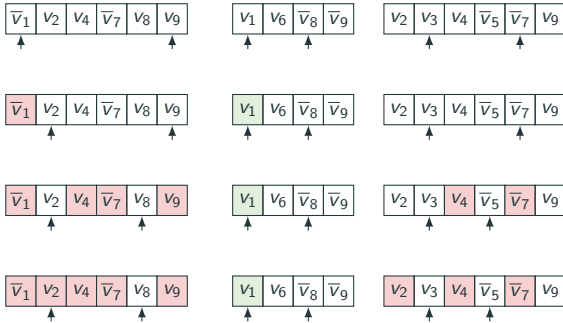


Example (Two-Watched literal scheme)

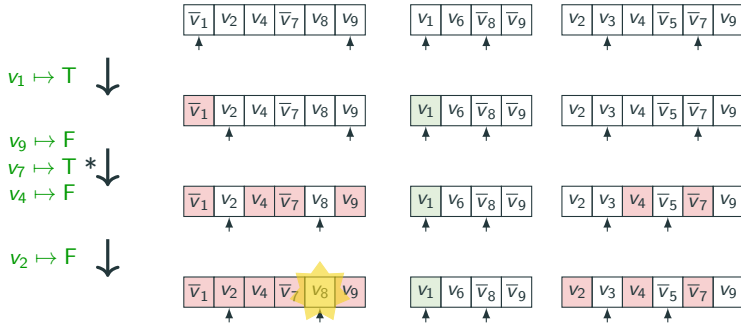
$v_1 \mapsto T$ ↓

$v_9 \mapsto F$
 $v_7 \mapsto T$ * ↓
 $v_4 \mapsto F$

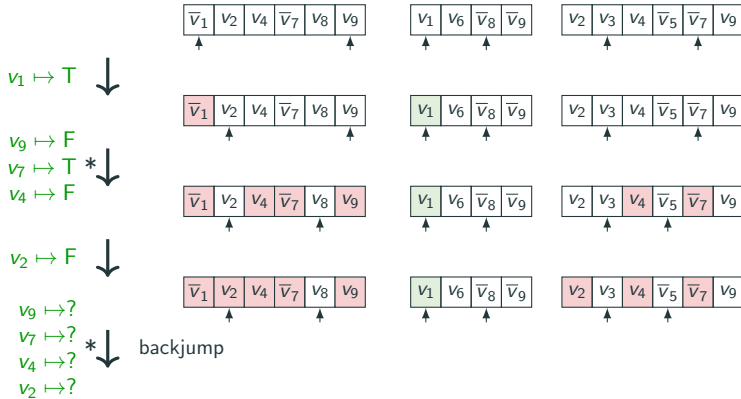
$v_2 \mapsto F$ ↓



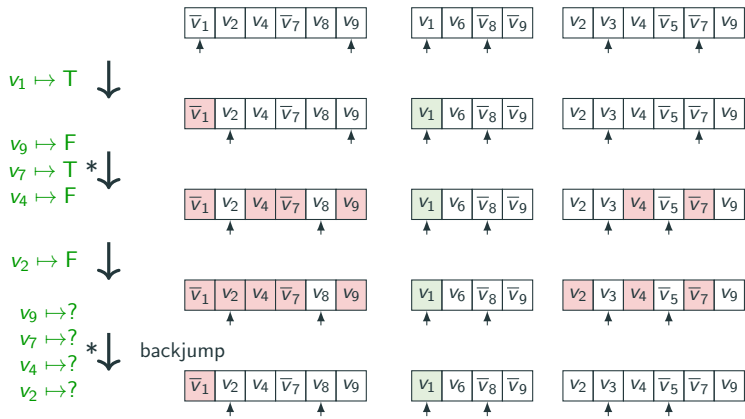
Example (Two-Watched literal scheme)



Example (Two-Watched literal scheme)



Example (Two-Watched literal scheme)



Example (Two-Watched literal scheme)



Example (Two-Watched literal scheme)



- Summary of Last Week
- From DPLL to Conflict Driven Clause Learning
- Application: Test Case Generation

Problem

given software system with n parameters, generate set of test cases which covers all problematic situations while being as small as possible

Problem

given software system with n parameters, generate set of test cases which covers all problematic situations while being as small as possible

Pairwise Testing

- ▶ well-practiced software testing method
- ▶ **observation**: most faults are caused by interaction of at most two parameters

Problem

given software system with n parameters, generate set of test cases which covers all problematic situations while being as small as possible

Pairwise Testing

- ▶ well-practiced software testing method
- ▶ **observation**: most faults are caused by interaction of at most two parameters

Example (Testing on Mobile Phones)

property	values
storage	32GB, 64GB, 128GB
cores	2, 4, 8
camera	8MP, 12MP, 16MP
SIM	single, dual
OS	Android, iOS

(a) testing model for mobile phones

Problem

given software system with n parameters, generate set of test cases which covers all problematic situations while being as small as possible

Pairwise Testing

- ▶ well-practiced software testing method
- ▶ **observation**: most faults are caused by interaction of at most two parameters

Example (Testing on Mobile Phones)

property	values						
		storage	cores	camera	SIM	OS	
storage	32GB, 64GB, 128GB	1	128GB	4	12MP	single	Android
cores	2, 4, 8	2	32GB	2	8MP	single	Android
camera	8MP, 12MP, 16MP	3	64GB	2	12MP	dual	iOS
SIM	single, dual	4	32GB	4	16MP	dual	iOS
OS	Android, iOS	5	64GB	8	16MP	single	Android
		6	128GB	8	8MP	dual	iOS
		7	128GB	2	12MP	dual	Android
		8	32GB	8	16MP	single	iOS
		9	64GB	4	8MP	single	iOS

(a) testing model for mobile phones

(b) test case set with pairwise coverage

Problem

given software system with n parameters, generate set of test cases which covers all problematic situations while being as small as possible

Pairwise Testing

- ▶ well-practiced software testing method
- ▶ **observation**: most faults are caused by interaction of at most two parameters

Example (Testing on Mobile Phones)

property	values						
		storage	cores	camera	SIM	OS	
storage	32GB, 64GB, 128GB	1	128GB	4	12MP	single	Android
cores	2, 4, 8	2	32GB	2	8MP	single	Android
camera	8MP, 12MP, 16MP	3	64GB	2	12MP	dual	iOS
SIM	single, dual	4	32GB	4	16MP	dual	iOS
OS	Android, iOS	5	64GB	8	16MP	single	Android
		6	128GB	8	8MP	dual	iOS
		7	128GB	2	12MP	dual	Android
		8	32GB	8	16MP	single	iOS
		9	64GB	4	8MP	single	iOS

some combinations may be infeasible

(a) testing model for mobile phones

(b) test case set with pairwise coverage

Encode Test Set of Fixed Size in SAT

- ▶ have n parameters, and parameter i has C_i values

Encode Test Set of Fixed Size in SAT

- ▶ have n parameters, and parameter i has C_i values
- ▶ for all m test cases use variables x_{ij} meaning that parameter i has value j

Encode Test Set of Fixed Size in SAT

- ▶ have n parameters, and parameter i has C_i values
- ▶ for all m test cases use variables x_{ij} meaning that parameter i has value j
- ▶ parameter j has exactly one value

$$\text{one_value}(x_{j1}, \dots, x_{jC_j}) = \bigvee_{1 \leq k \leq C_j} x_{jk} \wedge \bigwedge_{1 \leq k < k' \leq C_j} \neg x_{jk} \vee \neg x_{jk'}$$

Encode Test Set of Fixed Size in SAT

- ▶ have n parameters, and parameter i has C_i values
- ▶ for all m test cases use variables x_{ij} meaning that parameter i has value j
- ▶ parameter j has exactly one value

$$\text{one_value}(x_{j1}, \dots, x_{jC_j}) = \bigvee_{1 \leq k \leq C_j} x_{jk} \wedge \bigwedge_{1 \leq k < k' \leq C_j} \neg x_{jk} \vee \neg x_{jk'}$$

- ▶ in test case every parameter has one value

$$\text{test_case}(x_{11}, \dots, x_{nC_n}) = \bigwedge_{1 \leq j \leq n} \text{one_value}(x_{j1}, \dots, x_{jC_j})$$

Encode Test Set of Fixed Size in SAT

- ▶ have n parameters, and parameter i has C_i values
- ▶ for all m test cases use variables x_{ij} meaning that parameter i has value j
- ▶ parameter j has exactly one value

$$\text{one_value}(x_{j1}, \dots, x_{jC_j}) = \bigvee_{1 \leq k \leq C_j} x_{jk} \wedge \bigwedge_{1 \leq k < k' \leq C_j} \neg x_{jk} \vee \neg x_{jk'}$$

- ▶ in test case every parameter has one value

$$\text{test_case}(x_{11}, \dots, x_{nC_n}) = \bigwedge_{1 \leq j \leq n} \text{one_value}(x_{j1}, \dots, x_{jC_j})$$

- ▶ constraints on test case can be expressed by formula $\text{constraints}(x_{11}, \dots, x_{nC_n})$

Encode Test Set of Fixed Size in SAT

- ▶ have n parameters, and parameter i has C_i values
- ▶ for all m test cases use variables x_{ij} meaning that parameter i has value j
- ▶ parameter j has exactly one value

$$\text{one_value}(x_{j1}, \dots, x_{jC_j}) = \bigvee_{1 \leq k \leq C_j} x_{jk} \wedge \bigwedge_{1 \leq k < k' \leq C_j} \neg x_{jk} \vee \neg x_{jk'}$$

- ▶ in test case every parameter has one value

$$\text{test_case}(x_{11}, \dots, x_{nC_n}) = \bigwedge_{1 \leq j \leq n} \text{one_value}(x_{j1}, \dots, x_{jC_j})$$

- ▶ constraints on test case can be expressed by formula $\text{constraints}(x_{11}, \dots, x_{nC_n})$
- ▶ use overall encoding

$$\bigwedge_{1 \leq i \leq m} \text{test_case}(\overline{x^i}) \wedge \text{constraints}(\overline{x^i})$$

Encode Test Set of Fixed Size in SAT

- ▶ have n parameters, and parameter i has C_i values
- ▶ for all m test cases use variables x_{ij} meaning that parameter i has value j
- ▶ parameter j has exactly one value

$$\text{one_value}(x_{j1}, \dots, x_{jC_j}) = \bigvee_{1 \leq k \leq C_j} x_{jk} \wedge \bigwedge_{1 \leq k < k' \leq C_j} \neg x_{jk} \vee \neg x_{jk'}$$

- ▶ in test case every parameter has one value

$$\text{test_case}(x_{11}, \dots, x_{nC_n}) = \bigwedge_{1 \leq j \leq n} \text{one_value}(x_{j1}, \dots, x_{jC_j})$$

- ▶ constraints on test case can be expressed by formula $\text{constraints}(x_{11}, \dots, x_{nC_n})$
- ▶ use overall encoding assuming set of parameter pairs P

$$\bigwedge_{1 \leq i \leq m} \text{test_case}(\overline{x^i}) \wedge \text{constraints}(\overline{x^i}) \wedge \bigwedge_{(j,k),(j',k') \in P} \bigvee_{1 \leq i \leq m} x_{jk}^i \wedge x_{j'k'}^i$$

Encode Test Set of Fixed Size in SAT

- ▶ have n parameters, and parameter i has C_i values
- ▶ for all m test cases use variables x_{ij} meaning that parameter i has value j
- ▶ parameter j has exactly one value

$$\text{one_value}(x_{j1}, \dots, x_{jC_j}) = \bigvee_{1 \leq k \leq C_j} x_{jk} \wedge \bigwedge_{1 \leq k < k' \leq C_j} \neg x_{jk} \vee \neg x_{jk'}$$

- ▶ in test case every parameter has one value

$$\text{test_case}(x_{11}, \dots, x_{nC_n}) = \bigwedge_{1 \leq j \leq n} \text{one_value}(x_{j1}, \dots, x_{jC_j})$$

- ▶ constraints on test case can be expressed by formula $\text{constraints}(x_{11}, \dots, x_{nC_n})$
- ▶ use overall encoding assuming set of parameter pairs P

$$\bigwedge_{1 \leq i \leq m} \text{test_case}(\overline{x^i}) \wedge \text{constraints}(\overline{x^i}) \wedge \bigwedge_{(j,k),(j',k') \in P} \bigvee_{1 \leq i \leq m} x_{jk}^i \wedge x_{j'k'}^i$$

- ▶ Minimal test set can be found by repeating approach with smaller m

CDCL



João Marques-Silva, Inês Lynce, Sharad Malik.
Conflict-Driven Clause Learning SAT Solvers.
Handbook of Satisfiability 2021: 133-182.



Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, Sharad Malik.
Chaff: Engineering an Efficient SAT Solver
DAC 2001: 530-535.