universität innsbruck

# SAT and SMT Solving


**Sarah Winkler**

Department of Computer Science
University of Innsbruck


lecture 2
SS 2019

# Outline

- Summary of Last Week
- Conflict Analysis
- 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 $\quad M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\ l \parallel F,\ C \vee l$
  if $M \models \neg C$ and $l$ is undefined in $M$
- pure literal $\quad M \parallel F \quad \Longrightarrow \quad M\ l \parallel F$
  if $l$ occurs in $F$ but $l^c$ does not occur in $F$, and $l$ is undefined in $M$
- decide $\quad M \parallel F \quad \Longrightarrow \quad M\ l^d \parallel F$
  if $l$ or $l^c$ occurs in $F$, and $l$ is undefined in $M$
- backtrack $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l^c \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and $N$ contains no decision literals
- fail $\quad M \parallel F, C \quad \Longrightarrow \quad$ FailState
  if $M \models \neg C$ and $M$ contains no decision literals
- backjump $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \models C' \vee l'$ backjump clause
  - $M \models \neg C'$ and $l'$ is undefined in $M$, and $l'$ or $l'^c$ occurs in $F$ or in $M\ l^d\ N$

**Definition**

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

**Theorem (Termination)**

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

**Theorem (Correctness)**

*for derivation with final state $S_n$:*

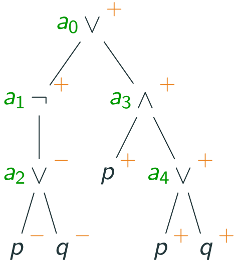
$$\| \; F \implies_{\mathcal{B}} S_1 \implies_{\mathcal{B}} S_2 \implies_{\mathcal{B}} \dots \implies_{\mathcal{B}} S_n$$

- *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 $\varphi$ in $\psi$ is $+$ if number of negations above $\varphi$ in $\psi$ is even, and $-$ otherwise

## Example (Efficient Transformations to CNF)




- $\varphi = \neg(p \lor q) \lor (p \land (p \lor q))$
- use fresh propositional variable for every connective

  $a_0 \colon \neg(p \lor q) \lor (p \land (p \lor q))$ $\quad$ $a_1 \colon \neg(p \lor q)$

  $a_2 \colon p \lor q$ $\quad$ $a_3 \colon p \land (p \lor q)$

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

  $$\varphi \approx \quad a_0 \land (a_0 \leftrightarrow a_1 \lor a_3) \land (a_1 \leftrightarrow \neg a_2) \land (a_2 \leftrightarrow p \lor q) \land (a_3 \leftrightarrow p \land a_2)$$

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

  $$\varphi \approx \quad a_0 \land (a_0 \rightarrow a_1 \lor a_3) \land (a_1 \rightarrow \neg a_2) \land (a_2 \leftarrow p \lor q) \land (a_3 \rightarrow p \land a_4) \land (a_4 \rightarrow p \lor q)$$

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

# Outline

- Summary of Last Week
- Conflict Analysis
- Conflict Driven Clause Learning
- Application: Test Case Generation

## Backjump: Idea

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

## Backjump: Idea

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

## Backjump to Definition

- backjump $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$

  if $M\ l^d\ N \vDash \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \vDash C' \vee l'$ backjump clause
  - $M \vDash \neg C'$ and $l'$ is undefined in $M$, and $l'$ or $l'^c$ occurs in $F$ or in $M\ l^d\ N$

## Backjump: Idea

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

## Backjump to Definition

- backjump $\quad M\, l^d\, N \parallel F, C \quad \Longrightarrow \quad M\, l' \parallel F, C$

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

## Example

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

## Backjump: Idea

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

## Backjump to Definition

- backjump $\quad M\, l^d\, N \parallel F, C \implies M\, l' \parallel F, C$

  if $M\, l^d\, N \vDash \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \vDash C' \vee l'$ backjump clause
  - $M \vDash \neg C'$ and $l'$ is undefined in $M$, and $l'$ or $l'^c$ occurs in $F$ or in $M\, l^d\, N$

## Example

$$\underbrace{1^d\ 2}_{M}\ \underbrace{3^d}_{l}\ \underbrace{4^d\ \overline{5}}_{N} \parallel \underbrace{\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5,\ \overline{1} \vee \overline{5} \vee 6,\ \overline{2} \vee \overline{5} \vee \overline{6}}_{F,C}$$

$M = 1^d\ 2 \qquad l = 3 \qquad N = 4^d\ \overline{5}$

## Backjump: Idea

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

## Backjump to Definition

- backjump $\quad M\, l^d\, N \parallel F, C \implies M\, l' \parallel F, C$

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

## Example

$$\underbrace{1^d\, 2}_{M}\ \underbrace{3^d}_{l}\ \underbrace{4^d\, \overline{5}}_{N} \parallel \underbrace{\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5,\ \overline{1} \vee \overline{5} \vee 6,\ \overline{2} \vee \overline{5} \vee \overline{6}}_{F,C}$$

$M = 1^d\, 2 \qquad l = 3 \qquad N = 4^d\, \overline{5} \qquad C = \overline{4} \vee 5$

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

## Backjump: Idea

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

## Backjump to Definition

- backjump $\qquad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$

  if $M\ l^d\ N \vDash \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \vDash C' \vee l'$ backjump clause
  - $M \vDash \neg C'$ and $l'$ is undefined in $M$, and $l'$ or $l'^c$ occurs in $F$ or in $M\ l^d\ N$

## Example

$$\underbrace{1^d\ 2}_{M}\ \underbrace{3^d}_{l}\ \underbrace{4^d\ \overline{5}}_{N} \parallel \underbrace{\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5,\ \overline{1} \vee \overline{5} \vee 6,\ \overline{2} \vee \overline{5} \vee \overline{6}}_{F, C}$$

$$M = 1^d\ 2 \qquad l = 3 \qquad N = 4^d\ \overline{5} \qquad C = \overline{4} \vee 5 \qquad C' = \overline{1} \qquad l' = \overline{5}$$

- $1^d\ 2\ 3^d\ 4^d\ \overline{5} \ \vDash\ \neg(\overline{4} \vee 5)$
- backjump clause $C' \vee l' = \overline{1} \vee \overline{5}$ satisfies $F, C \vDash C' \vee l'$

## Backjump: Idea

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

## Backjump to Definition

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

## Example

$$\underbrace{1^d\ 2}_{M}\ \underbrace{3^d}_{l}\ \underbrace{4^d\ \overline{5}}_{N} \parallel \underbrace{\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5,\ \overline{1} \vee \overline{5} \vee 6,\ \overline{2} \vee \overline{5} \vee \overline{6}}_{F, C}$$

$$M = 1^d\ 2 \qquad l = 3 \qquad N = 4^d\ \overline{5} \qquad C = \overline{4} \vee 5 \qquad C' = \overline{1} \qquad l' = \overline{5}$$

- $1^d\ 2\ 3^d\ 4^d\ \overline{5} \vDash \neg(\overline{4} \vee 5)$
- backjump clause $C' \vee l' = \overline{1} \vee \overline{5}$ satisfies $F, C \vDash C' \vee l'$
- $1^d\ 2 \vDash 1$

## Backjump: Idea

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

## Backjump to Definition

- backjump $\qquad M\, l^d\, N \parallel F, C \quad \Longrightarrow \quad M\, l' \parallel F, C$

  if $M\, l^d\, N \vDash \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \vDash C' \vee l'$ backjump clause
  - $M \vDash \neg C'$ and $l'$ is undefined in $M$, and $l'$ or $l'^c$ occurs in $F$ or in $M\, l^d\, N$

## Example

$$\underbrace{1^d\, 2}_{M}\; \underbrace{3^d}_{l}\; \underbrace{4^d\, \overline{5}}_{N} \parallel \underbrace{\overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5,\ \overline{1} \vee \overline{5} \vee 6,\ \overline{2} \vee \overline{5} \vee \overline{6}}_{F,C}$$

$$M = 1^d\, 2 \qquad l = 3 \qquad N = 4^d\, \overline{5} \qquad C = \overline{4} \vee 5 \qquad C' = \overline{1} \qquad l' = \overline{5}$$

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

## Backjump: Idea

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

## Backjump to Definition

- backjump $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$

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

## Example

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

$$\Longrightarrow 1^d\ 2\ \overline{5} \parallel \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4 \vee 5,\ \overline{2} \vee \overline{4} \vee \overline{5},\ 4 \vee \overline{5},\ \overline{4} \vee 5,\ \overline{1} \vee \overline{5} \vee 6,\ \overline{2} \vee \overline{5} \vee \overline{6}$$

$$M = 1^d\ 2 \qquad l = 3 \qquad N = 4^d\ \overline{5} \qquad C = \overline{4} \vee 5 \qquad C' = \overline{1} \qquad l' = \overline{5}$$

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

# Outline

- Summary of Last Week
- Conflict Analysis
- Conflict Driven Clause Learning
- Application: Test Case Generation

**Desirable Properties of Backjump Clauses**

- small
- should trigger progress

**How to Determine Backjump Clauses?**

- implication graph
- resolution

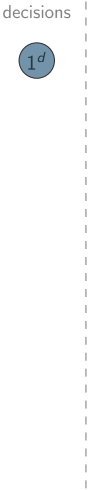
# Example: Implication Graph

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

decisions

# Example: Implication Graph

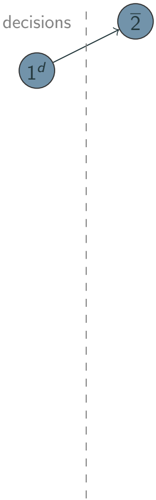
$$\varphi = (\overline{1} \lor \overline{2}) \land (\overline{1} \lor 2 \lor \overline{3}) \land (\overline{1} \lor 3 \lor 4) \land (\overline{4} \lor \overline{5} \lor \overline{6}) \land (\overline{5} \lor 6 \lor 7) \land$$
$$(\overline{7} \lor 8 \lor \overline{9} \lor 10) \land (\overline{10} \lor \overline{11}) \land (\overline{10} \lor 12) \land (\overline{12} \lor \overline{13}) \land (6 \lor 11 \lor 13)$$




| level | literal | reason |
| --- | --- | --- |
| 1 | 1 | decision |

# Example: Implication Graph

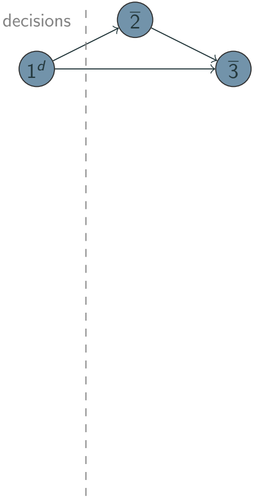
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




| level | literal | reason |
|---|---|---|
| 1 | 1 | decision |
| | $\overline{2}$ | $\overline{1} \vee \overline{2}$ |

# Example: Implication Graph

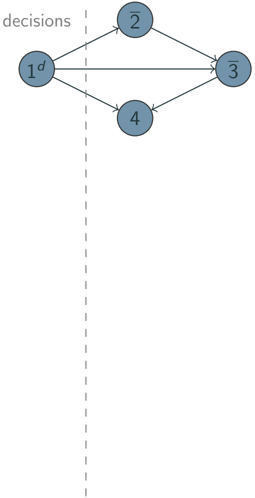
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




| level | literal | reason |
|---|---|---|
| 1 | $1$ | decision |
| | $\overline{2}$ | $\overline{1} \vee \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \vee 2 \vee \overline{3}$ |

# Example: Implication Graph

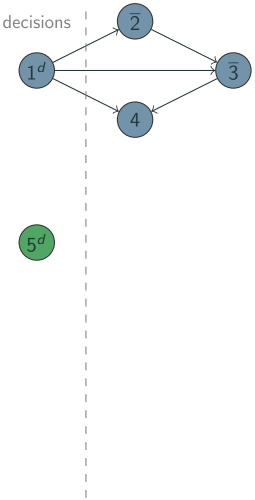
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




| level | literal | reason |
|---|---|---|
| 1 | $1$ | decision |
| | $\overline{2}$ | $\overline{1} \vee \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \vee 2 \vee \overline{3}$ |
| | $4$ | $\overline{1} \vee 3 \vee 4$ |

# Example: Implication Graph

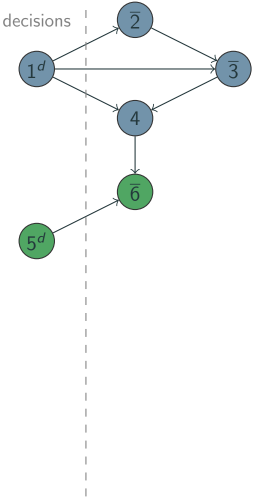
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge (\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10} \vee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




| level | literal | reason |
|---|---|---|
| 1 | 1 | decision |
| | $\overline{2}$ | $\overline{1} \vee \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \vee 2 \vee \overline{3}$ |
| | 4 | $\overline{1} \vee 3 \vee 4$ |
| 2 | 5 | decision |

# Example: Implication Graph

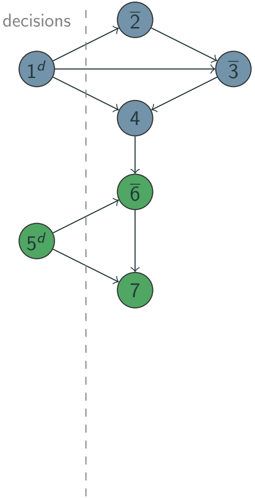
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




| level | literal | reason |
|---|---|---|
| 1 | 1 | decision |
| | $\overline{2}$ | $\overline{1} \vee \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \vee 2 \vee \overline{3}$ |
| | 4 | $\overline{1} \vee 3 \vee 4$ |
| 2 | 5 | decision |
| | $\overline{6}$ | $\overline{4} \vee \overline{5} \vee \overline{6}$ |

# Example: Implication Graph

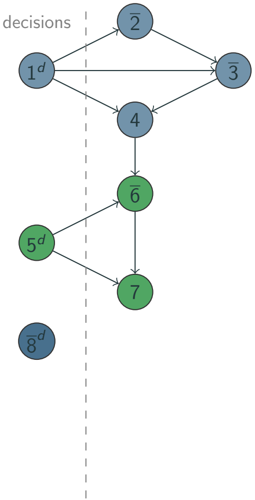
$$\varphi = (\overline{1} \lor \overline{2}) \land (\overline{1} \lor 2 \lor \overline{3}) \land (\overline{1} \lor 3 \lor 4) \land (\overline{4} \lor \overline{5} \lor \overline{6}) \land (\overline{5} \lor 6 \lor 7) \land$$
$$(\overline{7} \lor 8 \lor \overline{9} \lor 10) \land (\overline{10} \lor \overline{11}) \land (\overline{10} \lor 12) \land (\overline{12} \lor \overline{13}) \land (6 \lor 11 \lor 13)$$




| level | literal | reason |
|---|---|---|
| 1 | 1 | decision |
| | $\overline{2}$ | $\overline{1} \lor \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \lor 2 \lor \overline{3}$ |
| | 4 | $\overline{1} \lor 3 \lor 4$ |
| 2 | 5 | decision |
| | $\overline{6}$ | $\overline{4} \lor \overline{5} \lor \overline{6}$ |
| | 7 | $\overline{5} \lor 6 \lor 7$ |

# Example: Implication Graph

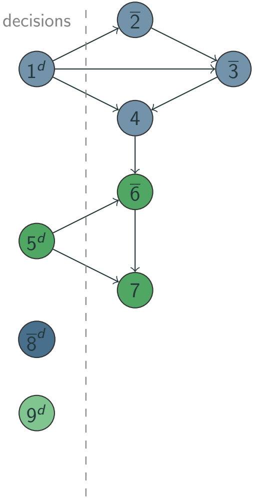
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




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

# Example: Implication Graph

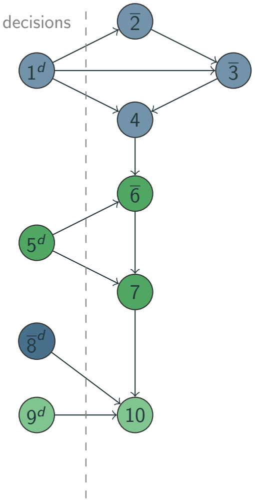
$$\varphi = (\overline{1} \lor \overline{2}) \land (\overline{1} \lor 2 \lor \overline{3}) \land (\overline{1} \lor 3 \lor 4) \land (\overline{4} \lor \overline{5} \lor \overline{6}) \land (\overline{5} \lor 6 \lor 7) \land$$
$$(\overline{7} \lor 8 \lor \overline{9} \lor 10) \land (\overline{10} \lor \overline{11}) \land (\overline{10} \lor 12) \land (\overline{12} \lor \overline{13}) \land (6 \lor 11 \lor 13)$$




| level | literal | reason |
|---|---|---|
| 1 | $1$ | decision |
| | $\overline{2}$ | $\overline{1} \lor \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \lor 2 \lor \overline{3}$ |
| | $4$ | $\overline{1} \lor 3 \lor 4$ |
| 2 | $5$ | decision |
| | $\overline{6}$ | $\overline{4} \lor \overline{5} \lor \overline{6}$ |
| | $7$ | $\overline{5} \lor 6 \lor 7$ |
| 3 | $\overline{8}$ | decision |
| 4 | $9$ | decision |

# Example: Implication Graph

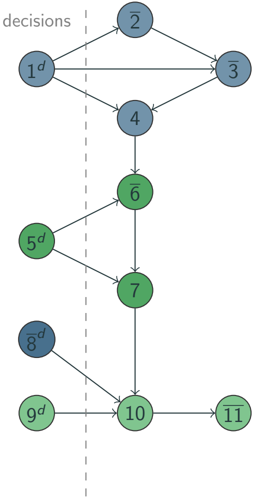
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




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

# Example: Implication Graph

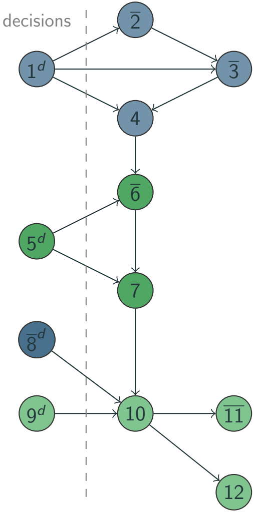
$$\varphi = (\overline{1} \lor \overline{2}) \land (\overline{1} \lor 2 \lor \overline{3}) \land (\overline{1} \lor 3 \lor 4) \land (\overline{4} \lor \overline{5} \lor \overline{6}) \land (\overline{5} \lor 6 \lor 7) \land$$
$$(\overline{7} \lor 8 \lor \overline{9} \lor 10) \land (\overline{10} \lor \overline{11}) \land (\overline{10} \lor 12) \land (\overline{12} \lor \overline{13}) \land (6 \lor 11 \lor 13)$$




| level | literal | reason |
|---|---|---|
| 1 | $1$ | decision |
| | $\overline{2}$ | $\overline{1} \lor \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \lor 2 \lor \overline{3}$ |
| | $4$ | $\overline{1} \lor 3 \lor 4$ |
| 2 | $5$ | decision |
| | $\overline{6}$ | $\overline{4} \lor \overline{5} \lor \overline{6}$ |
| | $7$ | $\overline{5} \lor 6 \lor 7$ |
| 3 | $\overline{8}$ | decision |
| 4 | $9$ | decision |
| | $10$ | $\overline{7} \lor 8 \lor \overline{9} \lor 10$ |
| | $\overline{11}$ | $\overline{10} \lor \overline{11}$ |

# Example: Implication Graph

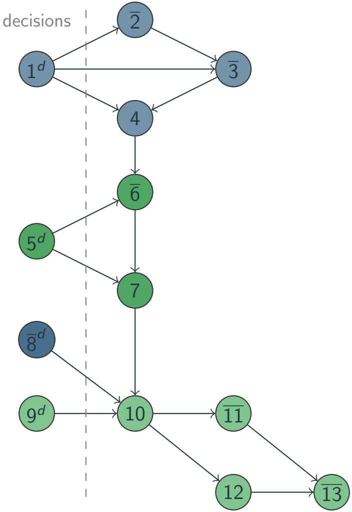
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




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

# Example: Implication Graph

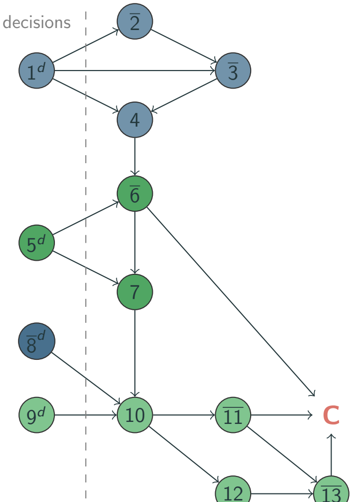
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




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

# Example: Implication Graph

$$\varphi = (\overline{1} \lor \overline{2}) \land (\overline{1} \lor 2 \lor \overline{3}) \land (\overline{1} \lor 3 \lor 4) \land (\overline{4} \lor \overline{5} \lor \overline{6}) \land (\overline{5} \lor 6 \lor 7) \land$$
$$(\overline{7} \lor 8 \lor \overline{9} \lor 10) \land (\overline{10} \lor \overline{11}) \land (\overline{10} \lor 12) \land (\overline{12} \lor \overline{13}) \land (6 \lor 11 \lor 13)$$



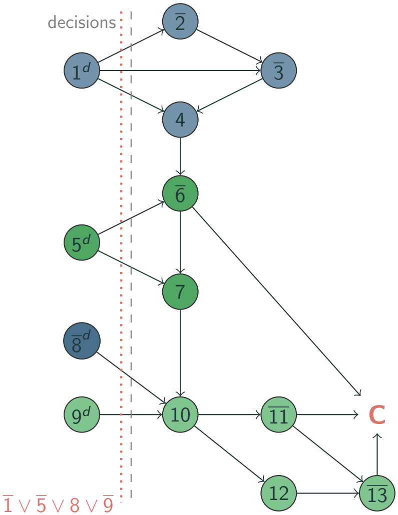

| level | literal | reason |
|---|---|---|
| 1 | 1 | decision |
| | $\overline{2}$ | $\overline{1} \lor \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \lor 2 \lor \overline{3}$ |
| | 4 | $\overline{1} \lor 3 \lor 4$ |
| 2 | 5 | decision |
| | $\overline{6}$ | $\overline{4} \lor \overline{5} \lor \overline{6}$ |
| | 7 | $\overline{5} \lor 6 \lor 7$ |
| 3 | $\overline{8}$ | decision |
| 4 | 9 | decision |
| | 10 | $\overline{7} \lor 8 \lor \overline{9} \lor 10$ |
| | $\overline{11}$ | $\overline{10} \lor \overline{11}$ |
| | 12 | $\overline{10} \lor 12$ |
| | $\overline{13}$ | $\overline{12} \lor \overline{13}$ |



next

# Example: Implication Graph

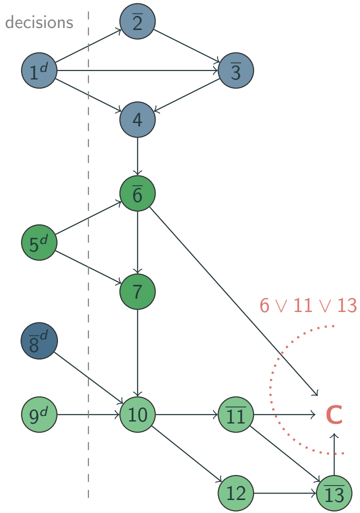
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge (\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10} \vee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




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

# Example: Implication Graph

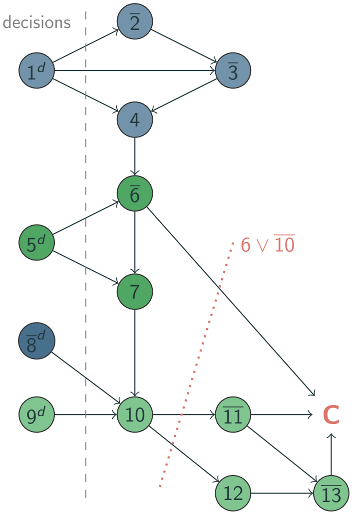
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge (\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10} \vee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




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

# Example: Implication Graph

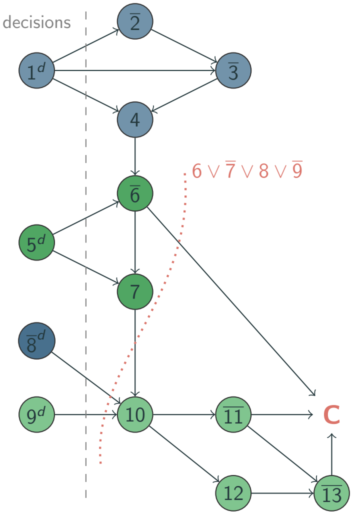
$$\varphi = (\overline{1} \vee \overline{2}) \wedge (\overline{1} \vee 2 \vee \overline{3}) \wedge (\overline{1} \vee 3 \vee 4) \wedge (\overline{4} \vee \overline{5} \vee \overline{6}) \wedge (\overline{5} \vee 6 \vee 7) \wedge$$
$$(\overline{7} \vee 8 \vee \overline{9} \vee 10) \wedge (\overline{10}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




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

# Example: Implication Graph

$$\varphi = (\overline{1} \lor \overline{2}) \land (\overline{1} \lor 2 \lor \overline{3}) \land (\overline{1} \lor 3 \lor 4) \land (\overline{4} \lor \overline{5} \lor \overline{6}) \land (\overline{5} \lor 6 \lor 7) \land$$
$$(\overline{7} \lor 8 \lor \overline{9} \lor 10) \land (\overline{10} \lor \overline{11}) \land (\overline{10} \lor 12) \land (\overline{12} \lor \overline{13}) \land (6 \lor 11 \lor 13)$$




| level | literal | reason |
|---|---|---|
| 1 | 1 | decision |
| | $\overline{2}$ | $\overline{1} \lor \overline{2}$ |
| | $\overline{3}$ | $\overline{1} \lor 2 \lor \overline{3}$ |
| | 4 | $\overline{1} \lor 3 \lor 4$ |
| 2 | 5 | decision |
| | $\overline{6}$ | $\overline{4} \lor \overline{5} \lor \overline{6}$ |
| | 7 | $\overline{5} \lor 6 \lor 7$ |
| 3 | $\overline{8}$ | decision |
| 4 | 9 | decision |
| | 10 | $\overline{7} \lor 8 \lor \overline{9} \lor 10$ |
| | $\overline{11}$ | $\overline{10} \lor \overline{11}$ |
| | 12 | $\overline{10} \lor 12$ |
| | $\overline{13}$ | $\overline{12} \lor \overline{13}$ |

# 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 $l_1 \to l'_1, \ldots, l_k \to l'_k$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ 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 $l_1 \to l_1', \dots, l_k \to l_k'$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ is entailed clause

## Example

- $\overline{1} \vee \overline{5} \vee 8 \vee \overline{9}$

# 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 $l_1 \to l'_1, \ldots, l_k \to l'_k$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ is entailed clause

## Example

- $\overline{1} \vee \overline{5} \vee 8 \vee \overline{9}$
- $6 \vee 11 \vee 13$

# 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 $l_1 \rightarrow l_1', \ldots, l_k \rightarrow l_k'$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ is entailed clause

## Example

- $\overline{1} \vee \overline{5} \vee 8 \vee \overline{9}$
- $6 \vee 11 \vee 13$
- $6 \vee \overline{10}$

# 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 $l_1 \to l_1', \ldots, l_k \to l_k'$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ is entailed clause

## Example

- $\overline{1} \vee \overline{5} \vee 8 \vee \overline{9}$
- $6 \vee 11 \vee 13$
- $6 \vee \overline{10}$
- $6 \vee \overline{7} \vee 8 \vee \overline{9}$

# 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 $l_1 \to l_1', \ldots, l_k \to l_k'$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ 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 $l_1 \to l'_1, \ldots, l_k \to l'_k$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ 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 $l_1 \to l'_1, \ldots, l_k \to l'_k$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ 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 $l_1 \to l_1', \ldots, l_k \to l_k'$ are cut edges then $l_1^c \vee \cdots \vee l_k^c$ 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 $l_1 \to l'_1, \ldots, l_k \to l'_k$ are cut edges then $l^c_1 \vee \cdots \vee l^c_k$ is entailed clause
- last decision literal is UIP

## Example

- UIPs are 9 and 10
- first UIP is 10

## Definition (Implication Graph)

Consider DPLL derivation $\| F \quad \Longrightarrow^*_{\mathcal{B}} M \| F$.

## Definition (Implication Graph)

Consider DPLL derivation to $\| F \quad \Longrightarrow_{\mathcal{B}}^{*} M \| F$.

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 $\| F \quad \Longrightarrow^*_{\mathcal{B}} M \| F$.

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 $l_1 \vee \ldots l_m \vee l'$ in $F$ such that there are already nodes $l_1^c, \ldots, l_m^c$

## Definition (Implication Graph)

Consider DPLL derivation to $\| F \quad \Longrightarrow^*_{\mathcal{B}} M \| F$.

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 $l_1 \vee \dots l_m \vee l'$ 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 $\parallel F \quad \Longrightarrow^*_{\mathcal{B}} M \parallel F$.

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 $l_1 \vee \ldots l_m \vee l'$ in $F$ such that there are already nodes $l_1^c, \ldots, l_m^c$
  - add node $l'$ if not yet present
  - add edges $l_i^c \rightarrow l'$ for all $1 \leqslant i \leqslant m$ if not yet present

## Definition (Implication Graph)

Consider DPLL derivation to $\| F \quad \Longrightarrow_{\mathcal{B}}^{*} M \| F$.

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 $l_1 \vee \dots l_m \vee l'$ in $F$ such that there are already nodes $l_1^c, \dots, l_m^c$
  - add node $l'$ if not yet present
  - add edges $l_i^c \to l'$ for all $1 \leqslant i \leqslant m$ if not yet present
- if $\exists$ clause $l'_1 \vee \cdots \vee l'_k$ in $F$ such that there are nodes $l_1'^c, \dots, l_k'^c$

## Definition (Implication Graph)

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

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 $l_1 \vee \dots l_m \vee l'$ in $F$ such that there are already nodes $l_1^c, \dots, l_m^c$
  - add node $l'$ if not yet present
  - add edges $l_i^c \to l'$ for all $1 \leqslant i \leqslant m$ if not yet present
- if $\exists$ clause $l_1' \vee \cdots \vee l_k'$ 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 $\| F \quad \Longrightarrow^*_{\mathcal{B}} M \| F$.

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 $l_1 \vee \ldots l_m \vee l'$ in $F$ such that there are already nodes $l_1^c, \ldots, l_m^c$
  - add node $l'$ if not yet present
  - add edges $l_i^c \to l'$ for all $1 \leqslant i \leqslant m$ if not yet present
- if $\exists$ clause $l_1' \vee \cdots \vee l_k'$ in $F$ such that there are nodes $l_1'^c, \ldots, l_k'^c$
  - add conflict node labeled $C$
  - add edges $l_i'^c \to C$

**Definition (Implication Graph)**

Consider DPLL derivation to $\| F \quad \Longrightarrow^*_{\mathcal{B}} M \| F$.

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 $l_1 \vee \dots l_m \vee l'$ in $F$ such that there are already nodes $l_1^c, \dots, l_m^c$
  - add node $l'$ if not yet present
  - add edges $l_i^c \to l'$ for all $1 \leqslant i \leqslant m$ 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 $l_i'^c \to C$

**Lemma**

*if edges intersected by cut are $l_1 \to l_1', \dots, l_k \to l_k'$ then $F \vDash l_1^c \vee \dots \vee l_k^c$*

## Definition (Implication Graph)

Consider DPLL derivation to $\| F \Longrightarrow^*_{\mathcal{B}} M \| F$.

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 $l_1 \vee \dots l_m \vee l'$ in $F$ such that there are already nodes $l_1^c, \dots, l_m^c$
  - add node $l'$ if not yet present
  - add edges $l_i^c \to l'$ for all $1 \leqslant i \leqslant m$ if not yet present
- if $\exists$ clause $l'_1 \vee \cdots \vee l'_k$ in $F$ such that there are nodes $l_1'^c, \dots, l_k'^c$
  - add conflict node labeled $C$
  - add edges $l_i'^c \to C$

potential backjump clause

## Lemma

*if edges intersected by cut are* $l_1 \to l'_1, \dots, l_k \to l'_k$ *then* $F \vDash l_1^c \vee \cdots \vee l_k^c$

# Resolution

## 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)

$$\frac{C \vee l \qquad C' \vee \neg l}{C \vee C'}$$

(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)

$$\frac{C \vee l \qquad C' \vee \neg l}{C \vee C'}$$

(assuming literals in clauses can be reordered)

## Example

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

## 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

- $C_0 = 6 \vee 11 \vee 13$

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

## 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

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

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

## 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

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

$$\dfrac{\dfrac{6 \vee 11 \vee 13 \qquad \overline{12} \vee \overline{13}}{6 \vee 11 \vee \overline{12}} \qquad \overline{10} \vee 12}{6 \vee 11 \vee \overline{10}}$$

## 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

- $C_0 = 6 \vee 11 \vee 13$
- $C_1 = 6 \vee 11 \vee \overline{12}$
- $C_2 = 6 \vee 11 \vee \overline{10}$

$$\dfrac{\dfrac{6 \vee 11 \vee 13 \qquad \overline{12} \vee \overline{13}}{6 \vee 11 \vee \overline{12}} \qquad \overline{10} \vee 12}{6 \vee 11 \vee \overline{10}}$$

## 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

- $C_0 = 6 \vee 11 \vee 13$
- $C_1 = 6 \vee 11 \vee \overline{12}$
- $C_2 = 6 \vee 11 \vee \overline{10}$

$$\dfrac{\dfrac{\dfrac{6 \vee 11 \vee 13 \qquad \overline{12} \vee \overline{13}}{6 \vee 11 \vee \overline{12}} \qquad \overline{10} \vee 12}{6 \vee 11 \vee \overline{10}} \qquad \overline{10} \vee \overline{11}}{6 \vee \overline{10}}$$

## 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

- $C_0 = 6 \vee 11 \vee 13$
- $C_1 = 6 \vee 11 \vee \overline{12}$
- $C_2 = 6 \vee 11 \vee \overline{10}$
- $C_3 = 6 \vee \overline{10}$

$$\dfrac{\dfrac{\dfrac{6 \vee 11 \vee 13 \qquad \overline{12} \vee \overline{13}}{6 \vee 11 \vee \overline{12}} \qquad \overline{10} \vee 12}{6 \vee 11 \vee \overline{10}} \qquad \overline{10} \vee \overline{11}}{6 \vee \overline{10}}$$

## 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

- $C_0 = 6 \vee 11 \vee 13$
- $C_1 = 6 \vee 11 \vee \overline{12}$
- $C_2 = 6 \vee 11 \vee \overline{10}$
- $C_3 = 6 \vee \overline{10}$

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

$$\frac{6 \vee 11 \vee \overline{12} \qquad \overline{10} \vee 12}{6 \vee 11 \vee \overline{10}}$$

$$\frac{6 \vee 11 \vee \overline{10} \qquad \overline{10} \vee \overline{11}}{6 \vee \overline{10}}$$

$$\frac{6 \vee \overline{10} \qquad \overline{7} \vee 8 \vee \overline{9} \vee 10}{6 \vee \overline{7} \vee 8 \vee \overline{9}}$$

## 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

- $C_0 = 6 \vee 11 \vee 13$
- $C_1 = 6 \vee 11 \vee \overline{12}$
- $C_2 = 6 \vee 11 \vee \overline{10}$
- $C_3 = 6 \vee \overline{10}$
- $C_4 = 6 \vee \overline{7} \vee 8 \vee \overline{9}$

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

$$\frac{6 \vee 11 \vee \overline{12} \qquad \overline{10} \vee 12}{6 \vee 11 \vee \overline{10}}$$

$$\frac{6 \vee 11 \vee \overline{10} \qquad \overline{10} \vee \overline{11}}{6 \vee \overline{10}}$$

$$\frac{6 \vee \overline{10} \qquad \overline{7} \vee 8 \vee \overline{9} \vee 10}{6 \vee \overline{7} \vee 8 \vee \overline{9}}$$

## Observations

- adding backjump clause to formula helps to prune search space (learning)
- if progress is too slow according to some measure, restart procedure: due to learned clauses heuristics will lead to different proof search

## Observations

- adding backjump clause to formula helps to prune search space (learning)
- if progress is too slow according to some measure, restart procedure:
  due to learned clauses heuristics will lead to different proof search

## Definition (DPLL with Learning and Restarts)

DPLL with learning and restarts $\mathcal{R}$ extends system $\mathcal{B}$ by following three rules:

## Observations

- adding backjump clause to formula helps to prune search space (learning)
- if progress is too slow according to some measure, restart procedure:
  due to learned clauses heuristics will lead to different proof search

## Definition (DPLL with Learning and Restarts)

DPLL with learning and restarts $\mathcal{R}$ extends system $\mathcal{B}$ by following three rules:

- learn $\qquad M \parallel F \quad \Longrightarrow \quad M \parallel F,\ C$
  if $F \vDash C$ and all atoms of $C$ occur in $M$ or $F$

## Observations

- adding backjump clause to formula helps to prune search space (learning)
- if progress is too slow according to some measure, restart procedure:
  due to learned clauses heuristics will lead to different proof search

## Definition (DPLL with Learning and Restarts)

DPLL with learning and restarts $\mathcal{R}$ extends system $\mathcal{B}$ by following three rules:

- learn $\quad M \parallel F \quad \Longrightarrow \quad M \parallel F,\ C$
  if $F \vDash C$ and all atoms of $C$ occur in $M$ or $F$

- forget $\quad M \parallel F,\ C \quad \Longrightarrow \quad M \parallel F$
  if $F \vDash C$

## Observations

- adding backjump clause to formula helps to prune search space (learning)
- if progress is too slow according to some measure, restart procedure:
  due to learned clauses heuristics will lead to different proof search

## Definition (DPLL with Learning and Restarts)

DPLL with learning and restarts $\mathcal{R}$ extends system $\mathcal{B}$ by following three rules:

- learn $\qquad M \parallel F \quad \Longrightarrow \quad M \parallel F,\ C$
  if $F \vDash C$ and all atoms of $C$ occur in $M$ or $F$
- forget $\qquad M \parallel F,\ C \quad \Longrightarrow \quad M \parallel F$
  if $F \vDash C$
- restart $\qquad M \parallel F \quad \Longrightarrow \quad \parallel F$

**Theorem (Termination)**

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

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

**Theorem (Termination)**

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

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

**Theorem (Termination)**

*any derivation* $\| F \Longrightarrow_{\mathcal{R}} S_1 \Longrightarrow_{\mathcal{R}} S_2 \Longrightarrow_{\mathcal{R}} \dots$ *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* $S_n$*:*

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

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

**Theorem (Termination)**

*any derivation* $\| F \Longrightarrow_{\mathcal{R}} S_1 \Longrightarrow_{\mathcal{R}} S_2 \Longrightarrow_{\mathcal{R}} \dots$ *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* $S_n$*:*

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

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

# Outline

- Summary of Last Week
- Conflict Analysis
- 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** | 32GB, 64GB, 128GB |
| **cores** | 2, 4, 8 |
| **camera** | 8MP, 12MP, 16MP |
| **SIM** | single, dual |
| **OS** | Android, iOS |

(a) testing model for mobile phones

| | storage | cores | camera | SIM | OS |
|---|---|---|---|---|---|
| 1 | 128GB | 4 | 12MP | single | Android |
| 2 | 32GB | 2 | 8MP | single | Android |
| 3 | 64GB | 2 | 12MP | dual | iOS |
| 4 | 32GB | 4 | 16MP | dual | 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 |

(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)

some combinations may be infeasible

| 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

| | storage | cores | camera | SIM | OS |
|---|---|---|---|---|---|
| 1 | 128GB | 4 | 12MP | single | Android |
| 2 | 32GB | 2 | 8MP | single | Android |
| 3 | 64GB | 2 | 12MP | dual | iOS |
| 4 | 32GB | 4 | 16MP | dual | 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 |

(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

$$\text{one\_value}(x_{j1}, \ldots, x_{jC_j}) = \bigvee_{1 \leqslant k \leqslant C_j} x_{jk} \wedge \bigwedge_{1 \leqslant k < k' \leqslant C_j} \neg x_{jk} \vee \neg x_{jk'}$$

# 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

$$\text{one\_value}(x_{j1}, \ldots, x_{jC_j}) = \bigvee_{1 \leqslant k \leqslant C_j} x_{jk} \wedge \bigwedge_{1 \leqslant k < k' \leqslant C_j} \neg x_{jk} \vee \neg x_{jk'}$$

- in test case every parameter has one value

$$\text{test\_case}(x_{11}, \ldots, x_{nC_n}) = \bigwedge_{1 \leqslant j \leqslant n} \text{one\_value}(x_{j1}, \ldots, x_{jC_j})$$

## 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

$$\text{one\_value}(x_{j1}, \ldots, x_{jC_j}) = \bigvee_{1 \leqslant k \leqslant C_j} x_{jk} \land \bigwedge_{1 \leqslant k < k' \leqslant C_j} \neg x_{jk} \lor \neg x_{jk'}$$

- in test case every parameter has one value

$$\text{test\_case}(x_{11}, \ldots, x_{nC_n}) = \bigwedge_{1 \leqslant j \leqslant n} \text{one\_value}(x_{j1}, \ldots, x_{jC_j})$$

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

## 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

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

- in test case every parameter has one value

$$\text{test\_case}(x_{11}, \dots, x_{nC_n}) = \bigwedge_{1 \leqslant j \leqslant n} \text{one\_value}(x_{j1}, \dots, x_{jC_j})$$

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

$$\bigwedge_{1 \leqslant i \leqslant m} \text{test\_case}(\overline{x^i}) \wedge \text{constraints}(\overline{x^i})$$

## 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

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

- in test case every parameter has one value

$$\text{test\_case}(x_{11}, \dots, x_{nC_n}) = \bigwedge_{1 \leqslant j \leqslant n} \text{one\_value}(x_{j1}, \dots, x_{jC_j})$$

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

$$\bigwedge_{1 \leqslant i \leqslant m} \text{test\_case}(\overline{x^i}) \wedge \text{constraints}(\overline{x^i}) \wedge \bigwedge_{(j,k),(j',k') \in P} \bigvee_{1 \leqslant i \leqslant m} x^i_{jk} \wedge x^i_{j'k'}$$

## 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

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

- in test case every parameter has one value

$$\text{test\_case}(x_{11},\dots,x_{nC_n}) = \bigwedge_{1\leqslant j\leqslant n} \text{one\_value}(x_{j1},\dots,x_{jC_j})$$

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

$$\bigwedge_{1\leqslant i\leqslant m} \text{test\_case}(\overline{x^i}) \wedge \text{constraints}(\overline{x^i}) \wedge \bigwedge_{(j,k),(j',k')\in P} \bigvee_{1\leqslant i\leqslant m} x^i_{jk} \wedge x^i_{j'k'}$$

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