

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

$$M \parallel F \implies M' \parallel F' \text{ or } \text{FailState}$$

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

$$a_0: \neg(p \vee q) \vee (p \wedge (p \vee q)) \quad a_1: \neg(p \vee q)$$

$$a_2: p \vee q \quad a_3: p \wedge (p \vee q)$$

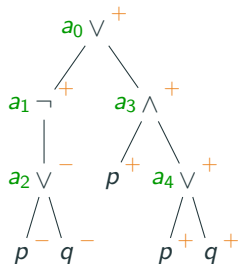
- ▶ Tseitin: add clause a_0 plus $(a_i \leftrightarrow \dots)$ for every subformula

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

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

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

- ▶ 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 $C' \vee I'$ 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

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

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

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

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

Outline

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

Desirable Properties of Backjump Clauses

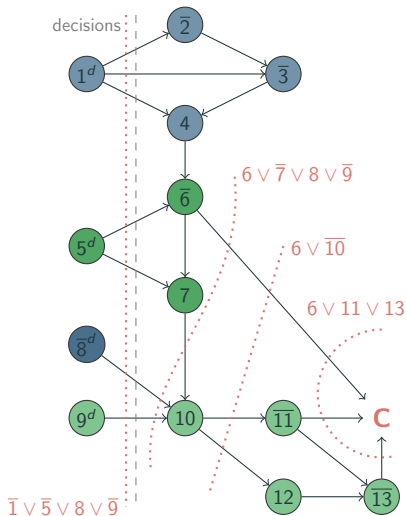
- ▶ small
- ▶ should trigger progress

How to Determine Backjump Clauses?

- ▶ implication graph
- ▶ resolution

Example: Implication Graph

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



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
- ▶ 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

- ▶ $\bar{1} \vee \bar{5} \vee 8 \vee \bar{9}$
- ▶ $6 \vee 11 \vee 13$
- ▶ $6 \vee \bar{10}$
- ▶ $6 \vee \bar{7} \vee 8 \vee \bar{9}$
- ▶ UIPs are 9 and 10

Definition (Implication Graph)

Consider DPLL derivation to $\| F \implies_B^* M \| F$.

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 $l_1 \vee \dots \vee l_m \vee l'$ 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 $F \models l_1^c \vee \dots \vee l_k^c$

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
- ▶ 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

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

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** $M \parallel F \implies M \parallel F, C$
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
- ▶ *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 $M \models F$

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

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	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

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

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

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