universität innsbruck

SAT and SMT Solving

Sarah Winkler

KRDB
Department of Computer Science
Free University of Bozen-Bolzano

lecture 4
WS 2022

Outline

- Summary of Last Week
- Unsatisfiable Cores
- Application: FPGA Routing
- Algorithm by Fu and Malik
- Unsatisfiable Cores in Practice

Maximum Satisfiability

Consider CNF formulas χ and φ as sets of clauses such that χ is satisfiable.

Definitions

- $\text{maxSAT}(\varphi)$ is maximal $\sum_{C\in\varphi} \overline{v}(C)$ for valuation v
- $\text{pmaxSAT}(\varphi, \chi)$ is maximal $\sum_{C\in\varphi} \overline{v}(C)$ for valuation v with $v(\chi) = \mathrm{T}$

Definitions

given weights $w_C \in \mathbb{Z}$ for all $C \in \varphi$,

- $\text{maxSAT}_w(\varphi)$ is maximal $\sum_{C\in\varphi} w_C \cdot \overline{v}(C)$ for valuation v?
- $\text{pmaxSAT}_w(\varphi, \chi)$ is maximal $\sum_{C\in\varphi} w_C \cdot \overline{v}(C)$ for valuation v with $v(\chi) = \mathrm{T}$

Definition

$\text{minUNSAT}(\varphi)$ is minimal $\sum_{C\in\varphi} \overline{v}(\neg C)$ for valuation v

Lemma

$$|\varphi| = |\text{minUNSAT}(\varphi)| + |\text{maxSAT}(\varphi)|$$

Branch & Bound

Idea

- gets list of clauses φ as input return minUNSAT(φ)
- explores assignments in depth-first search

```
function BnB(φ, UB)
  φ = simp(φ)
  if φ contains only empty clauses then
      return #empty(φ)
  if #empty(φ) ⩾ UB then
      return UB
  x = selectVariable(φ)
  UB′ = min(UB, BnB(φ_x, UB))
  return min(UB′, BnB(φ_x̄, UB′))
```

Theorem

$\mathtt{BnB}(\varphi, |\varphi|) = \mathsf{minUNSAT}(\varphi)$

Binary Search

Idea

- gets list of clauses φ as input and returns $\mathrm{minUNSAT}(\varphi)$
- repeatedly call SAT solver in binary search fashion

Definitions

- cardinality constraint is
$$\sum_{x \in X} x \bowtie N$$
where $\bowtie$ is $=$, $<$, $>$, $\leqslant$, or $\geqslant$, X is set of propositional variables, and $N \in \mathbb{N}$
- valuation v satisfies $\sum_{x \in X} x \bowtie N$ iff $k \bowtie N$
where k is number of variables $x \in X$ such that $v(x) = \mathrm{T}$

Remark

cardinality constraints are expressible in CNF

Algorithm (Binary Search)

function BinarySearch($\{C_1, \ldots, C_m\}$)

  $\varphi := \{C_1 \vee b_1, \ldots, C_m \vee b_m\}$

  return search(φ,0,m)

$b_1, \ldots, b_m$ are fresh variables

function search(φ, L, U)

  if L $\geqslant$ U then

    return U

  mid $:= \lfloor \frac{\mathtt{U+L}}{2} \rfloor$

  if SAT($\varphi \wedge$ CNF($\sum_{i=1}^{m} b_i \leqslant$ mid)) then

    return search(φ, L, mid)

  else

    return search(φ, mid + 1, U)

Theorem

BinarySearch(ψ) = minUNSAT(ψ)

Outline

- Summary of Last Week
- Unsatisfiable Cores
- Application: FPGA Routing
- Algorithm by Fu and Malik
- Unsatisfiable Cores in Practice

Definitions

for unsatisfiable CNF formula φ given as set of clauses

- unsatisfiable core (UC) of φ is $\psi \subseteq \varphi$ such that $\bigwedge_{C \in \psi} C$ is unsatisfiable
- UC ψ is minimal if every strict subset of ψ is satisfiable
- SUC (smallest unsatisfiable core) is UC such that $|\psi|$ is minimal

Example

$$\varphi = \{\neg x, \quad x \vee z, \quad \neg y \vee \neg z, \quad x, \quad y \vee \neg z\}$$

unsatisfiable cores are

- φ
- $\{\, \neg x,\ x \vee z,\ \neg y \vee \neg z,\ y \vee \neg z \,\}$ minimal
- $\{\, \neg x,\ x \,\}$ minimal and SUC

Remark

SUC is always minimal unsatisfiable core

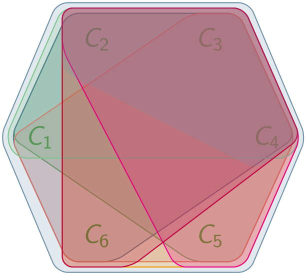
Example

$\varphi = \{C_1, \ldots, C_6\}$ is unsatisfiable

$C_1 : x_1 \vee \neg x_3$ $\quad$ $C_2 : x_2$ $\quad$ $C_3 : \neg x_2 \vee x_3$

$C_4 : \neg x_2 \vee \neg x_3$ $\quad$ $C_5 : x_2 \vee x_3$ $\quad$ $C_6 : \neg x_1 \vee x_2 \vee \neg x_3$

φ has 9 unsatisfiable cores:



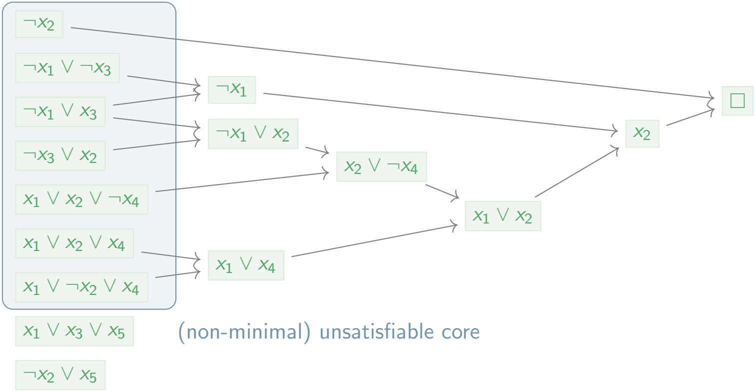
$UC_1 = \{C_1, C_2, C_3, C_4, C_5, C_6\}$
$UC_2 = \{C_1, C_2, C_3, C_4, C_5\}$
$UC_3 = \{C_1, C_2, C_3, C_4, C_6\}$
$UC_4 = \{C_1, C_3, C_4, C_5, C_6\}$
$UC_5 = \{C_2, C_3, C_4, C_5, C_6\}$
$UC_6 = \{C_1, C_2, C_3, C_4\}$
$UC_7 = \{C_2, C_3, C_4, C_5\}$
$UC_8 = \{C_2, C_3, C_4, C_6\}$
$UC_9 = \{C_2, C_3, C_4\}$ minimal and SUC

Finding Minimal Unsatisfiable Cores by Resolution

Idea

- repeatedly pick clause C from φ and check satisfiability: if $\varphi \setminus \{C\}$ is satisfiable, keep C for UC, otherwise drop C
- SAT solvers can give resolution proof if conflict detected: use resolution graphs for more efficient implementation of this idea

Example (Resolution Graph)



Assume φ is unsatisfiable.

Definition (Resolution Graph)

directed acyclic graph $G = (V, E)$ is resolution graph for set of clauses φ if

1. $V = V_i \uplus V_c$ is set of clauses and $V_i = \varphi$,
2. V_i nodes have no incoming edges, initial nodes
3. there is exactly one node $\square$ without outgoing edges,
4. $\forall C \in V_c \; \exists$ edges $D \to C$, $D' \to C$ such that C is resolvent of D and D', and
5. there are no other edges.

Remark

- if φ is unsatisfiable then sequence of resolution steps can derive $\square$ because resolution is complete proof method
- so resolution graph exists

Notation

- $Reach_G(C)$ is set of nodes reachable from C in G
- $Reach_G^E(C)$ is set of edges reachable from C in G
- $\overline{N}$ is $V \setminus N$ for any set of nodes N

Algorithm minUnsatCore(φ)

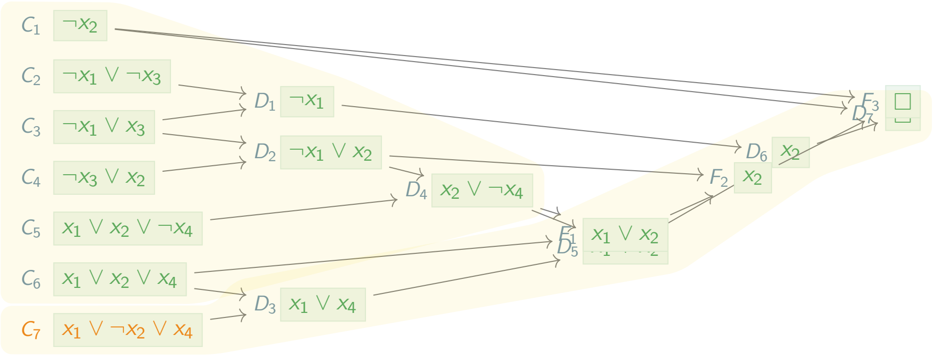
Input: unsatisfiable formula φ
Output: minimal unsatisfiable core of φ

build resolution graph $G = (V_i \uplus V_c, E)$ for φ
while $\exists$ unmarked clause in V_i **do**
  $C \leftarrow$ unmarked clause in V_i
  if SAT($\overline{Reach_G(C)}$) **then** ▷ subgraph without C satisfiable?
    mark C ▷ C is UC member
  else
    build resolution graph $G' = (V_i' \uplus V_c', E')$ for $\overline{Reach_G(C)}$
    $V_i \leftarrow V_i \setminus \{C\}$ and $V_c \leftarrow V_c' \cup (V_c \setminus Reach_G(C))$
    $E \leftarrow E' \cup (E \setminus Reach_G^E(C))$
    $G \leftarrow (V_i \cup V_c, E)$
    $G \leftarrow G|_\Box$ ▷ restrict to nodes with path to $\Box$
return V_i

Theorem

if φ unsatisfiable then minUnsatCore(φ) is minimal unsatisfiable core of φ

Example



minUnsatCore(φ)

re-use relevant resolvents: fewer steps to $\square$

- pick C_7
- $Reach_G(C_7) = \{C_7, D_3, D_5, D_6, D_7\}$ $\overline{Reach_G(C_7)} = \{C_1, \ldots, C_6, D_1, D_2, D_4\}$
- check $\mathrm{SAT}(\overline{Reach_G(C_7)})$
- unsatisfiable: get new resolution graph G_7 for $\varphi \cup \{D_1, D_2, D_4\}$
- construct resolution graph G' for φ by adding edges from G to G_7
- set G to G' restricted to nodes with path to $\square$
- after 5 more loop iterations: return $\{C_1, C_3, \ldots, C_6\}$

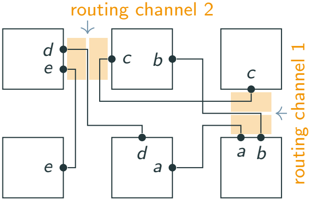
Application: FPGA Routing

Field Programmable Gate Arrays (FPGAs)

- can simulate microprocessors but faster for special tasks (from complex combinatorics to mere logic)
- logic blocks connected by "routing channels"
- "routing": determine which channels are used for what

Example (Encoding Routing Requirements)

- consider connections a, b, c, d, e of 2 bits each
- liveness: want to route $\geqslant 1$ bit of a, b, c, d, e
- 2 routing channels of 2 tracks each
- exclusivity: each channel has only 2 tracks
- unsatisfiable: UCs indicate problems



$a_0 \vee a_1$	$\neg a_0 \vee \neg b_0$	$\neg c_0 \vee \neg d_0$
$b_0 \vee b_1$	$\neg a_0 \vee \neg c_0$	$\neg c_0 \vee \neg e_0$
$c_0 \vee c_1$	$\neg b_0 \vee \neg c_0$	$\neg d_0 \vee \neg e_0$
$d_0 \vee d_1$	$\neg a_1 \vee \neg b_1$	$\neg c_1 \vee \neg d_1$
$e_0 \vee e_1$	$\neg a_1 \vee \neg c_1$	$\neg c_1 \vee \neg e_1$
	$\neg b_1 \vee \neg c_1$	$\neg d_1 \vee \neg e_1$

UC_1: channel 1 capacity exceeded

UC_2: channel 2 capacity exceeded

UC_3: c is overconstrained

UC_4: c is overconstrained

Outline

- Summary of Last Week
- Unsatisfiable Cores
- Application: FPGA Routing
- Algorithm by Fu and Malik
- Unsatisfiable Cores in Practice

Bounds for Maximum Satisfiability

consider CNF formula $\varphi = C_1 \wedge \cdots \wedge C_m$

Definition

blocked formula is $\varphi_B = (C_1 \vee b_1) \wedge \cdots \wedge (C_m \vee b_m)$ for fresh variables $b_1, \dots, b_m$

Lemma (Lower Bound)

if v satisfies φ_B and $B_{\mathsf{T}} = \{b_i \mid v(b_i) = \mathsf{T}\}$ then $\mathsf{maxSAT}(\varphi) \geqslant m - |B_{\mathsf{T}}|$

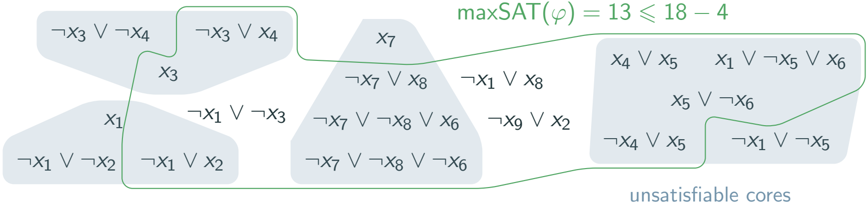
Lemma (Upper Bound)

if φ contains k disjoint unsatisfiable cores then $\mathsf{maxSAT}(\varphi) \leqslant m - k$

must miss at least one clause from every core!

Example (Upper Bound)

$\mathsf{maxSAT}(\varphi) = 13 \leqslant 18 - 4$



Algorithm by Fu and Malik

Idea

- maxsat valuation must make at least one clause in unsatisfiable core false
- while there exists (minimal) unsatisfiable core:

  relax formula such that one clause from core need not be satisfied
- until formula becomes satisfiable

Definition (Partial minUNSAT)

pminUNSAT(χ, φ) is minimal $\sum_{C \in \varphi} \overline{v}(\neg C)$ for valuation v with $v(\chi) = \mathsf{T}$

Lemma

$$|\varphi| = \mathrm{pminUNSAT}(\chi, \varphi) + \mathrm{pmaxSAT}(\chi, \varphi)$$

Example

χ:	$\neg x_1 \vee x_3$	$\neg x_7 \vee x_2$	$x_7 \vee x_2$	$x_1 \vee \neg x_2$
φ:	$\neg x_1 \vee \neg x_2 \vee b_1$	$\neg x_1 \vee x_2 \vee b_2$	$\neg x_1 \vee x_7$	$x_1 \vee b_3$
	$\neg x_3 \vee x_4 \vee c_1$	$x_3 \vee c_2$	$\neg x_3 \vee \neg x_4 \vee c_3$	$x_4 \vee x_5$
	$\neg x_4 \vee x_5$	$x_1 \vee \neg x_5 \vee x_6$	$x_5 \vee \neg x_6$	$x_7 \vee d_1$
	$\neg x_7 \vee x_8 \vee d_2$	$\neg x_7 \vee \neg x_8 \vee x_6 \vee d_3$	$\neg x_7 \vee \neg x_8 \vee \neg x_6 \vee d_4$	$\neg x_1 \vee \neg x_3 \vee e_1$

- unsatisfiable core: $\neg x_1 \vee \neg x_2,\ \neg x_1 \vee x_2,\ x_1$
  $\chi = \chi \cup \mathrm{CNF}(b_1 + b_2 + b_3 = 1)$
  $cost = 1$
- unsatisfiable core: $\neg x_3 \vee x_4,\ x_3, \neg x_3 \vee \neg x_4$
  $\chi = \chi \cup \mathrm{CNF}(c_1 + c_2 + c_3 = 1)$
  $cost = 2$
- unsatisfiable core: $x_7,\ \neg x_7 \vee x_8,\ \neg x_7 \vee \neg x_8 \vee x_6,\ \neg x_7 \vee \neg x_8 \vee \neg x_6$
  $\chi = \chi \cup \mathrm{CNF}(d_1 + d_2 + d_3 + d_4 = 1)$
  $cost = 3$
- unsatisfiable core: $\neg x_1 \vee x_3,\ \neg x_7 \vee x_2,\ x_7 \vee x_2,\ x_1 \vee \neg x_2,\ \neg x_1 \vee \neg x_3$
  $\chi = \chi \cup \mathrm{CNF}(e_1 = 1)$
  $cost = 4$
- satisfiable: $v(x_1) = v(x_2) = v(x_3) = v(x_5) = v(x_7) = \mathrm{T}$ and $v(x_i) = \mathrm{F}$ otherwise
- $\mathrm{pminUNSAT}(\chi, \varphi) = 4$ and $\mathrm{pmaxSAT}(\chi, \varphi) = 12$

Algorithm FuMalik(χ, φ)

Input: soft clauses φ and satisfiable hard clauses χ
Output: pminUNSAT(χ, φ)

```
cost ← 0
while ¬SAT(χ ∪ φ) do
    UC ← unsatCore(χ ∪ φ)                     ▷ UC must be minimal
    B ← ∅
    for C ∈ UC ∩ φ do                          ▷ loop over soft clauses in core
        φ ← φ \ {C} ∪ {C ∨ b}                 ▷ b is fresh "blocking" variable
        B ← B ∪ {b}
    χ ← χ ∪ CNF(∑_{b∈B} b = 1)                ▷ cardinality constraint is hard
    cost ← cost + 1
return cost
```

Theorem

FuMalik$(\chi, \varphi) =$ pminUNSAT(χ, φ)

Unsatisfiable Cores in z3

```
from z3 import *

x1,x2,x3 = Bool("x1"), Bool("x2"), Bool("x3")
phi = [ Or(Not(x1), Not(x2)), Or(Not(x1), x2),\
  Or(Not(x1), x3), x1, Or(Not(x3), x2)]

solver = Solver()
solver.set(unsat_core=True)

# assert clauses in phi with names phi0 ... phi4
for i,c in enumerate(phi):
  solver.assert_and_track(c, "phi" + str(i))

if solver.check() == z3.unsat:
  uc = solver.unsat_core()
  print(uc) # [phi0, phi1, phi3]
```

Literature

Nachum Dershowitz, Ziyad Hanna, and Alexander Nadel.
A Scalable Algorithm for Minimal Unsatisfiable Core Extraction.
Proc. 9th Conference on Theory and Applications of Satisfiability, pp. 36–41, 2006.

Yoonna Oh, Maher Mneimneh, Zaher Andraus, Karem Sakallah, and Igor Markov
AMUSE: A Minimally-Unsatisfiable Subformula Extractor.
Proc. 41st Design Automation Conference, pp. 518–523, 2004.

Zhaohui Fu and Sharad Malik.
On solving the partial MAX-SAT problem.
In Proc. 9th Conference on Theory and Applications of Satisfiability, pp. 252–265, 2006