# Satisfiability

- SAT problem
- Standard representation
- DPLL algorithm
- Automata and SAT
- SMT

Def. Given a well-formed formula in propositional logic, determine whether there exists a satisfying solution (or valuation).

Boolean Satisfiability Problem SAT

Example :

Set of variables

type

Boolean operators

Well-founded formula (informal) is a formula involving the variables and operators "properly"

Recall, a valuation $\mathbf{x}$ maps each $x_i \in X$ to $\{0,1\}$

A valuation $\mathbf{x}$ satisfies $\alpha$ if each $x_i$ in $\alpha$ replaced by $\mathbf{x}\lceil x_i$ evaluates to true.

We write this as $\mathbf{x} \models \alpha$

otherwise $\mathbf{x}$ does not satisfy $\alpha$, $\mathbf{x} \not\models \alpha$

Example :

**Def.** Given a well-formed formula in propositional logic, determine whether there exists a satisfying solution (or valuation).

$$\exists x \in val(X) : x \models \alpha ?$$

if yes $\alpha$ is satisfiable
else $\alpha$ is unsatisfiable

if $\forall x \in val(x)$ $x \models \alpha$ then $\alpha$ is valid or a tautology.

if $\alpha$ is valid then $\neg\alpha$ is unsatisfiable.

$\alpha$ and $\alpha'$ are tautologically equivalent if they have the same truth tables

$$\forall x \in val(X) : x \models \alpha \Leftrightarrow x \models \alpha'$$

$\alpha$ and $\alpha'$ are equisatisfiable if
$\alpha$ is satisfiable iff $\alpha'$ is also satisfiable

# How to solve SAT?

- A solver for SAT can be used to solve any other problem in the NP-Class with only polynomial slow down
- Make sense to build SAT Solvers
- Modern SAT solvers can solve problems with 10K+ variables and million + Clauses

# How do SAT solvers work?

We assume $\alpha$ to be in Conjunctive normal form (CNF)

literals : variable or negation $x_3$ $\neg x_3$
Clause : disjunction (OR) of literals
$x_1 \vee x_2 \vee \neg x_3$
CNF : a formula Conjunction of Clauses
$\alpha$

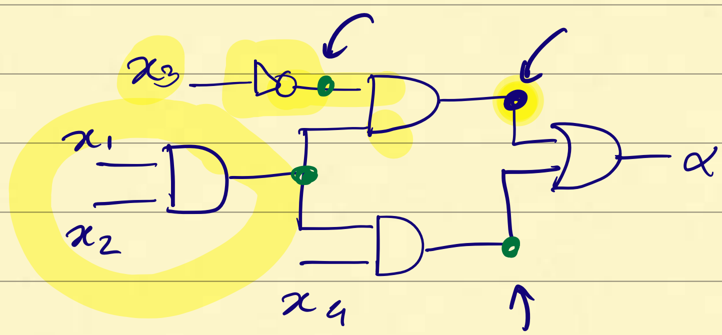
$x_2$ appears positively in the first clause
negatively in the second

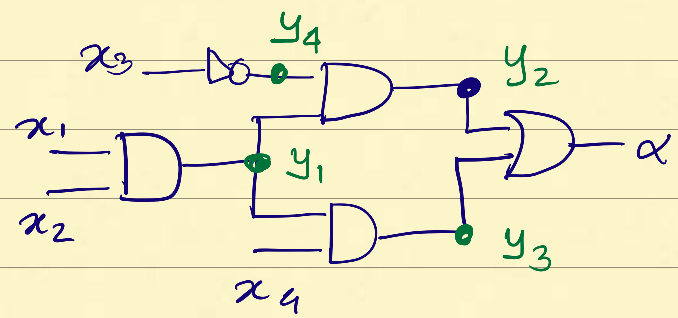
How to construct CNF?

Logic and Circuits

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

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






$y_4 \Leftrightarrow \neg x_3$

$x_1 \wedge x_2 \Leftrightarrow y_1$

$y_1 \wedge x_3 \Leftrightarrow y_3$

— 3

$\alpha \Leftrightarrow y_2 \wedge y_3$

— 4

—

$\alpha' \equiv 1 \wedge 2 \wedge 3 \wedge 4$

# Standard representations of CNF

- $(\neg x_1 \vee \neg x_2 \vee x_5) \wedge (\neg x_5 \vee x_1)$
  $\wedge (\neg x_5 \vee x_2)$

- $(x_1' + x_2' + x_5)(x_5' + x_1)$
  $(x_5' + x_2)$

- (-1 -2 5)(-5 1)(-5 2)
  DIMACS

- SMTLib

# GSAT

input : Clauses $C$ over $X$

parameters $\underline{\text{max flips}}$, $\underline{\text{max-tries}}$

output : $X$ satisfying $\alpha$ or $\emptyset$

for $i = 1$ to max_tries

    $\underline{v}$ := random choice from val($X$)

    for $j = 1$ to max flips

        if $v \models C$ return $v$

        $p$:= variable in $C$ s.t. flipping $p$ gives the largest increase in # satisfied clauses

        $v = v$ with assignment to $p$ flipped

return $\phi$

# Davis Putnam Logemann Loveland (DPLL) Algorithm 1962

- Transform the given formula $\alpha$ by applying a sequence of satisfiability preserving rules

- if final result has no literals $\rightarrow$ unsatisfiable
- if final result has no clauses $\rightarrow$ satisfiable

# Davis Putnam Algorithm (DP) 1960

Rule 1. Unit propagation
Rule 2. Pure literal
Rule 3. Resolution

Rule 1. Unit Prop

if a clause has a single literal

$$\alpha \equiv \ldots (x_1 \vee \neg p \vee x_2) \wedge \boxed{p} \wedge \cdots \wedge (\neg x_3 \vee \neg p \vee x_1)$$

$$\alpha' \equiv \ldots (x_1 \vee x_2) \wedge \cdots \wedge (\neg x_3 \vee x_1)$$

$\alpha$ and $\alpha'$ are equisatisfiable.

## Rule 2. Pure literal

A literal $p$ appears only positively or negatively
set $p = 1$ (or 0) and remove all the
$\neg p$ occurrences

$$\alpha \equiv \cdots \wedge (x_1 \vee \neg p \vee x_2) \wedge (x_4 \vee \neg p) \wedge \cdots \wedge (\neg x_3 \vee x_1) \cdots$$

$p$ does not appear anywhere

Makes sense to set $p = 0$

$$\alpha' \equiv \cdots \wedge \cdots \wedge \cdots (\neg x_3 \vee x_1) \cdots$$

## Rule 3. Resolution

Choose a literal $p$ that appears both +vely and
−vely : $(l_1 \vee l_2 \vee \cdots \vee p)$
$(k_1 \vee k_2 \vee \cdots \neg p)$

resolved clause $(l_1 \vee l_2 \vee \cdots \vee k_1 \vee k_2 \vee \cdots)$

Pairwise resolve each such pair

Take conjunction of all the resolve clauses.

- Why is the result equisatisfiable?

What is the size of $\alpha$ after resolution?

$O(N^2)$

DPLL modifies resolution in DP
with DFS

Rule 3'. Let $\Delta$ be the curent set of clauses
Choose a literal $p$ in $\Delta$
Check satisfiability of $\Delta \cup \{p\}$
(guess $p=1$)
if SAT then return True
else return result of Satisfiability
of $\Delta \cup \{\neg p\}$

| Problem | tautology | dptaut | dplltaut |
|---|---|---|---|
| prime 3 | 0.00 | 0.00 | 0.00 |
| prime 4 | 0.02 | 0.06 | 0.04 |
| prime 9 | 18.94 | 2.98 | 0.51 |
| prime 10 | 11.40 | 3.03 | 0.96 |
| prime 11 | 28.11 | 2.98 | 0.51 |
| prime 16 | >1 hour | out of memory | 9.15 |
| prime 17 | >1 hour | out of memory | 3.87 |
| ramsey 3 3 5 | 0.03 | 0.06 | 0.02 |
| ramsey 3 3 6 | 5.13 | 8.28 | 0.31 |
| mk_adder_test 3 2 | >>1 hour | 6.50 | 7.34 |
| mk_adder_test 4 2 | >>1 hour | 22.95 | 46.86 |
| mk_adder_test 5 2 | >>1 hour | 44.83 | 170.98 |
| mk_adder_test 5 3 | >>1 hour | 38.27 | 250.16 |
| mk_adder_test 6 3 | >>1 hour | out of memory | 1186.4 |
| mk_adder_test 7 3 | >>1 hour | out of memory | 3759.9 |

From Slides of Clark Barrett's lecture.
Summer School on Verification Technology, Systems & Applications, September 17, 2008 – p. 42/98

## Abstract DPLL as an automaton

States and transitions

$M \parallel F$

M: sequence of literals denoting partial assignment to variables

F: CNF formula being checked represented as list of clauses

initial state $\emptyset \parallel F$

final state : Fail ($F$ is UNSAT)

$M \| G$   $G$ equisatisfiable with $F$ and $M \models G$

Transitions

Unit prop   $M \| F, C \vee \ell \rightarrow M\ell \| F, C \vee \ell$

if $\ell$ is unassigned in $M$

and $M \models \neg C$

Pure literal   $M \| F \longrightarrow M\ell \| F$

if $\ell$ is unassigned in $M$

$\ell$ occurs in some clause in $F$

$\neg \ell$ does not occur in $F$

Decide   $M \| F \longrightarrow M\ell^d \| F$

if $\ell$ is unassigned in $M$

both $\ell$ and $\neg \ell$ appear in some clauses in $F$

Backtrack   $M\ell^d N \| F, C \rightarrow M \neg \ell \| F, C$

if $M\ell^d N \models \neg C$

and $N$ contains no decision literals

Fail $\quad M \parallel F, C \longrightarrow \text{Fail}$

if $M \models \neg C$

and $M$ contains no decision literals

# Example DPLL

$\phi \parallel 1 \vee \bar{2} \quad \bar{1} \vee \bar{2} \quad 2 \vee 3 \quad \bar{3} \vee 2 \quad 1 \vee 4$

# Automata, Reachability, and SAT.

Given an automaton $\mathcal{A} = \langle V, \Theta, A, \mathcal{D} \rangle$

and a candidate invariant $I \subseteq val(V)$

how to write the invariance check

(Thm 7.1) as a satisfiability

problem?

Example: Automaton

Variables

$x_1 : \mathbb{B}$ } To be filled

$x_2 : \mathbb{B}$

transitions

pre $x_1 \vee x_2$ Pre $\neg(x_1 \oplus x_2)$

eff $x_2 := 1$ eff $x_1 := \neg x_1$

pre $x_2 \wedge x_1$ $x_2 := \neg x_2$

eff $x_2 := 0$

Sample executions

$\langle 0,1\rangle \rightarrow \langle 1,1\rangle \rightarrow \langle 0,1\rangle \rightarrow \langle 0,1\rangle$

$\rightarrow \langle 1,1\rangle$

The transition relation of this automaton can be written a relation

$I$ $\qquad\qquad\qquad$ $\mathcal{D}$

$\Theta =$

Suppose $I$ is an invaint

$$I(x_1, x_2) \equiv x_1 \oplus x_2$$

Checking the invariant can be stated as a SAT question.

(start condition)

(transition cond)

To check validity of this statement we check the satisfiability of its negation