Lecture 8

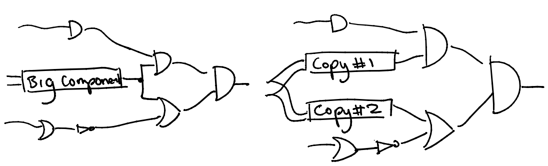
Wednesday, February 12, 2020 1:35 PM

Announcements:

1. No class on Monday.
2. Project proposals due on Feb 19
3. Typo in hw1 has been fixed.

	Circuits	Formulas	CNF	DNF	ROBDD
sat	NPC	NPC	NPC	trivial	trivial
validity	coNPC	coNPC	trivial	coNPC	trivial
model counting	#PC	#PC	#PC	#PC	trivial

Circuits $\Longrightarrow$ Formulas

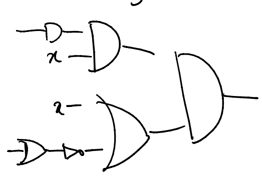


Possibly exponential blowup.

Challenge: Getting rid of fanout
aka. sharing
aka. let bindings

let x = Big Component

in



Formulas $\Longrightarrow$ CNF, DNF

Distributivity: $A \cdot B + C = (A+C)\cdot(B+C)$

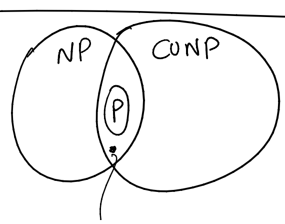
$(A+B)\cdot C = A\cdot C + B\cdot C$

Double negations: $\overline{\overline{A}} = A$

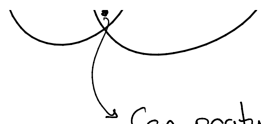
De Morgan: $\overline{A\cdot B} = \overline{A} + \overline{B}$

$\overline{A+B} = \overline{A}\cdot\overline{B}$

Possibly exponential blowup.



Open problem:
Is $NP \cap Co\text{-}NP = P$?



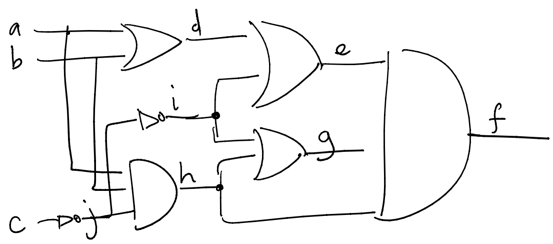
Can parity games be solved in poly time?

Circuit $\Longrightarrow$ Equivalent formula

Formula $\Longrightarrow$ Equivalent CNF

Circuit $\Longrightarrow$ Equisatisfiable CNF

Tseitin's transform



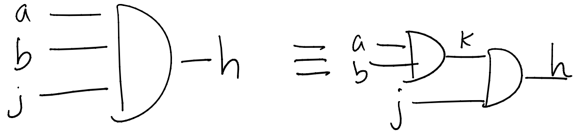
$$\varphi(a,b,c) \overset{\text{equisatisfiable}}{\rightleftharpoons} \psi(a,b,c,d,e,f,g,h,i,j)$$

$$\boxed{d = a+b} \equiv d \Rightarrow a+b \text{ and } a+b \Rightarrow d$$

$$\underbrace{(\bar{d}+a+b)} \text{ and } \underbrace{\overline{a+b}+d}$$

$$\bar{a}\cdot\bar{b}+d$$

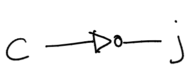
$$\boxed{(\bar{d}+a+b) \cdot (\bar{a}+d)\cdot(\bar{b}+d)}$$



$$k = a\cdot b \equiv k \Rightarrow a\cdot b \text{ and } a\cdot b \Rightarrow k$$

$$\bar{k} + a\cdot b \quad \text{and} \quad \overline{a\cdot b} + k$$

$$(\bar{k}+a)\cdot(\bar{k}+b) \text{ and } (\bar{a}+\bar{b}+k)$$



$$j = \bar{c} \equiv j \Rightarrow \bar{c} \text{ and } \bar{c} \Rightarrow j$$

$$(\bar{j}+\bar{c}) \text{ and } (\bar{\bar{c}}+j)$$

$(\bar{j} + \bar{c})$ and $(\bar{\bar{c}} + j)$

$(\bar{j} + \bar{c})$ and $(c + j)$

<u>Theorem</u>: $\varphi(a,b,c)$ is satisfiable iff $\psi(a,b,c,\ldots)$ is satisfiable.

<u>Proof</u>

<u>Case 1</u>: If φ is satisfiable then ψ is satisfiable.

Just use same a, b, c.

Use values computed by the circuit for everything else.

<u>Case 2</u>: If ψ is satisfiable, then φ is satisficble.

Just project out the values of a, b, c.

MiniSAT + $\underbrace{\text{Satelite}}$ + Glucose + ...

Tools to convert formulas into CNF.

<u>How to determine satisfiability of CNF formulas?</u>

<u>Alg 1 (φ)</u>

1. For each assignment $S: V \to \{\text{true}, \text{false}\}$ check if S satisfies φ. If so return sat
2. Return unsat

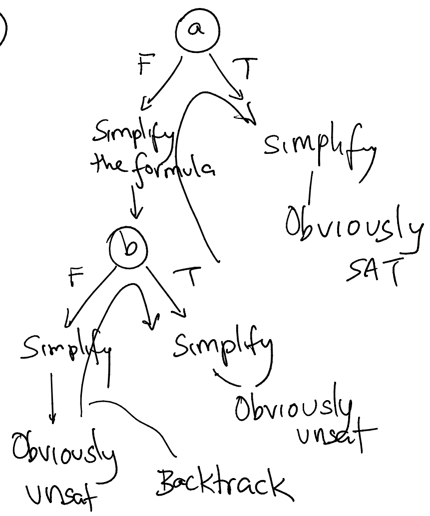
<u>Theorem</u>: Alg 1 returns sat iff φ is satisfiable.

is satisfiable.

But! Might require $2^{|V|}$ time!

Incrementally coming up with satisfying assignments

$\varphi(a, b, c, d)$



$(\bar{a} + b + c)$ ✓

and $(a + c + d)$ ✓

and $(a + c + \bar{d})$ ✓

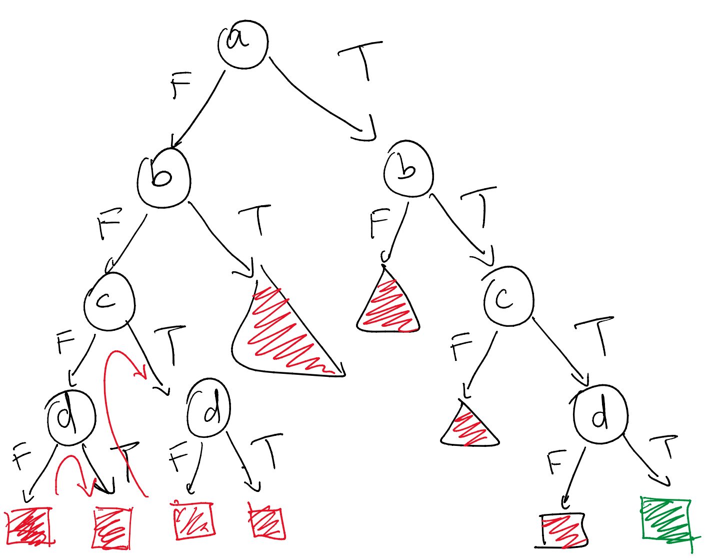
and $(a + \bar{c} + d)$ ✓

and $(a + \bar{c} + \bar{d})$ ✓

and $(\bar{b} + \bar{c} + d)$

and $(\bar{a} + b + \bar{c})$ ✓

and $(\bar{a} + \bar{b} + c)$ ✓



$\text{Alg}_2(\varphi,$ partial assignment $S)$

If $\exists$ clause $c \in \varphi$ already unsat,

return unsat

If $\forall$ clauses C, c already sat,
then return sat.

Pick a variable $v \notin S$

If $Alg_2(\varphi, S \cup \{v\})$ return sat

If $Alg_2(\varphi, S \cup \{\bar{v}\})$ return sat

Return unsat.

<u>Theorem</u>: $Alg_2(\varphi, \{\})$ returns sat

Initial partial assignment

iff φ is satisfiable.

<u>Unit propagation</u>

$(a + b + \bar{c})$

Partial assignment: $\{\bar{a}, \bar{b}\}$.

Then c should be false in any <u>satisfying completion</u>

1. Formalize unit propagation
2. DPLL (for free)
3. Polytime algorithms for

    Horn-SAT

    2-SAT.

Unit-Propagate (φ S partial assignment):

$S' := S$.

For each clause $c = (l_1 \text{ or } l_2 \text{ or } \cdots \text{ or } l_k)$ in φ:

if $\exists\, l_i$ which is unassigned and $\forall\, l_j \neq l_i$, l_j is false then $S' := S \cup \{l_i\}$.

If for some v, both v & $\bar{v} \in S'$ then report unsat

If $S' \neq S$: return $UP(\varphi, S')$ else ~~return~~ S.

Example 1 : $S = \{\}$

$(a \text{ or } b)$ and $(\bar{a} \text{ or } c)$ and $(\bar{c} \text{ or } d)$ and a.

$\equiv \underbrace{(a \text{ or } b)}_{1}$ and $\underbrace{(a \Rightarrow c)}_{2}$ and $\underbrace{(c \Rightarrow d)}_{3}$ and $\underbrace{a}_{4}$.

a

$\downarrow T$ (No need to worry about 4 or 1)

c

$\downarrow T$

d

$\downarrow T$

Formula satisfied

Example 2 : $(\bar{a} \text{ or } b)$ and $(\bar{a} \text{ or } \bar{b})$ and a

$\equiv \underbrace{(a \Rightarrow b)}_{1}$ and $\underbrace{(a \Rightarrow \bar{b})}_{2}$ and $\underbrace{a}_{3}$

a

$\downarrow T$ B.C. of clause 3.

b

$\downarrow T$ B.c. of clause 1

b

$| F$ B.C of clause 2 .

(b)
$\downarrow$ F B.C of clause ②.

$\therefore$ Unsat

DPLL (φ partial assignment S)

$S' :=$ unit-propagate (φ S)

If S' is unsat, then return unsat.

Pick a variable v not assigned in S'.

Check $r_1 =$ DPLL(φ, $S' \cup \{v\}$)

If r_1 is sat, return sat

Check $r_2 =$ DPLL(φ, $S' \cup \{\bar{v}\}$)

If r_2 is sat, return sat.

Return unsat.

<u>Theorem</u>: DPLL (φ $\{\}$) returns sat iff φ is satisfiable.

Vijay Ganesh / Waterloo

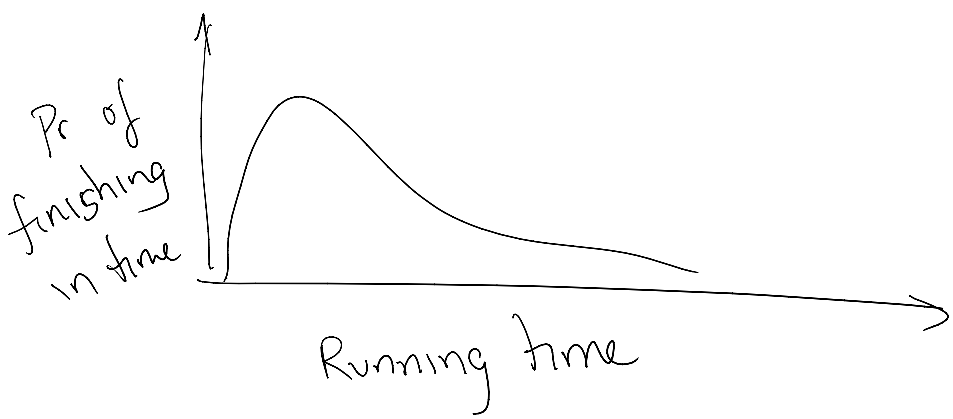
Maple SAT.

... decide branching

– Maple SAT 1.
Machine learning to decide branching variables.

DPLL $\Longrightarrow$ Conflict Driven Clause Learning (CDCL)

Clause learning ~ ① Why did a partial assignment fail?
② Non-chronological backtracking
③ Restarts



φ

---

① Tseitin's transform

② Simple algorithms for SAT : Alg_1 & Alg_2

③ Unit propagation.

④ DPLL

5 Horn - SAT
2 - SAT

6 CDCL (next class)

Horn-SAT : If φ is a formula made of only Horn clauses then is φ is satisfiable?

Horn-clause : Clause $c = l_1 \vee l_2 \vee \dots \vee l_k$

c is a Horn clause if it has at most one positive literal.

$\bar{a} \vee \bar{b} \vee \bar{c} \equiv \overline{a \wedge b \wedge c}$

$\bar{a} \vee \bar{b} \vee c \equiv \overline{a \cdot b} \vee c \equiv \underbrace{a \cdot b \Rightarrow c}$

c

1 Unit propagation will start by setting this to true

2 Force you to set other things to true.

(3) Say we end up with a partial assignment S.

__Claim__: S is a part of every satisfying assignment of φ.

(4) If we have discovered that the formula is unsat, we are done.

(5) Otherwise: set all unassigned vars to false. Formula is SAT.

__Example__: $\underbrace{c}_{(1)}$ and $\underbrace{c \Rightarrow a}_{(2)}$ and $\underbrace{a \cdot c \Rightarrow d}_{(3)}$ and $\underbrace{a \cdot b \Rightarrow f}_{(4)}$ and $\underbrace{\bar{f} + \bar{b}}_{(5)}$

Step 1: c must be true.

Step 2: a must be true

Step 3: d must be true

Unit prop is done.

Clauses (1) (2) (3) are satisfied

Variables b, f are unassigned.

Set them to false.

I claim 4 & 5 are also satisfied.

Example: $\underbrace{c \Rightarrow a}_{1}$ and $\underbrace{a \cdot c \Rightarrow d}_{2}$ and $\underbrace{a \cdot b \Rightarrow f}_{3}$ and $\underbrace{\bar{f} + \bar{b}}_{4}$.

Unit propagation is a no-op.

Set every variable to false.

In every $b \Rightarrow h$, b is false. So $b \Rightarrow h$ is true.

In every $\bar{x} + \bar{y} + \dots + \bar{z}$, every var is false.

So each literal is true.

So the clause is also true.

Every clause has been satisfied.