PH 5840 Lectures- 14

Note Title 9/9/2015

Classical Computational Models & Complexity classes

(1) Circuit Model:- Classical computing device evaluates,

$$f : \{0,1\}^n \longrightarrow \{0,1\}^m$$

Input: n-bit binary string

output: m-bit binary string

→ m functions of the form

$$g : \{0,1\}^n \longrightarrow \{0,1\}$$

output is a bit

$g \in$ "Boolean" functions : Map n-bit inputs to a single bit of output.

* n-bit Boolean function $f_n(x)$ ⟷ (one-to-one correspondence) Binary strings of length 2^n

⇒ # of n-bit Boolean fns $= 2^{2^n}$!!

- Universality : Arbitrary Boolean fn. can be reduced to a sequence of basic logical operations :-

Given $f_n(x)$: $S \equiv \{$ set of all input strings 'x' satisfying $f(x) = 1 \}$

$\overline{S} \equiv \{$ set of all 'x' satisfying $f(x) = 0 \}$

Let $\$ = (x^{(1)}, x^{(2)}, \ldots x^{(i)}, \ldots)$

Define functions: $f^{(i)}(x) = \begin{cases} 1 \text{ if } x = x^{(i)} \\ 0 \text{ otherwise} \end{cases}$

(a) Then, $f_n(x) = f^{(1)}(x) \vee f^{(2)}(x) \vee \ldots f^{(i)}(x) \vee \ldots$

↑ logical OR operation

For a pair of bits (x,y): $x \vee y = x + y - x \cdot y \pmod 2$

Let $x = x_{n-1} \cdots x_1 x_0$

(b) If $x^{(i)} = 111..111$, $f^{(i)}(x) = x_{n-1} \wedge x_{n-2} \wedge \cdots x_1 \wedge x_0$

↑ logical AND operation

$x \wedge y = x \cdot y$

For any other sting, just use NOT gate at appropriate locations.

∴ We've reconstructed $f_n(x)$ out of AND, OR, NOT as well as, INPUT(x_i)

---

* Alternate "proof":- By induction.

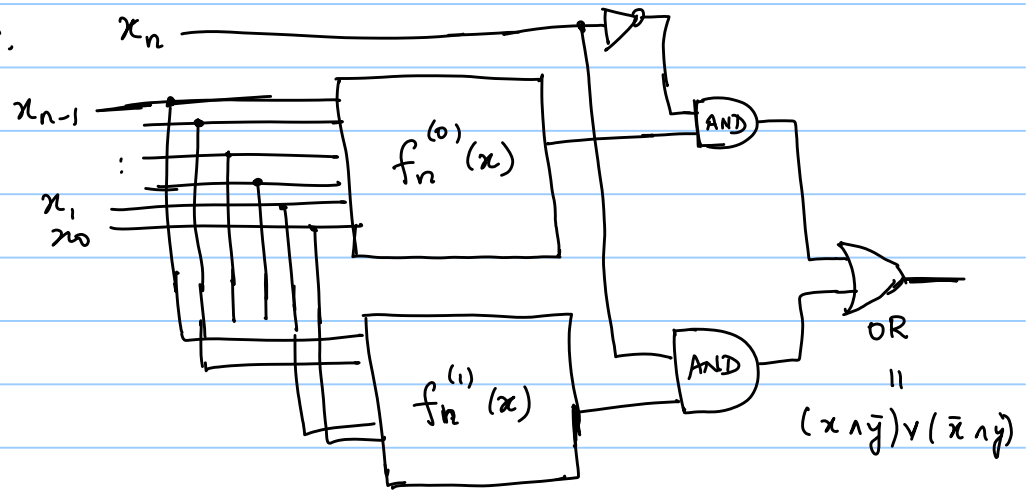
- For $n=1$, $f_1(x)$ — flip (NOT operation)
  - identity (just a wire!)
  - Replace 'x' by '0' ($x \wedge 0$) ↳ Ancilla
  - Replace 'x' by '1' ($x \vee 1$) ↳ Ancilla bit

- If ∃ a "circuit" to implement $f_n(x)$, can implement $f_{n+1}(x)$ using AND, OR, NOT :-

$$f_n^{(0)}(x) = f_{n+1}(0\, x_{n-1}\, x_{n-2} \cdots x_0)$$
$$f_n^{(1)}(x) = f_{n+1}(1\, x_{n-1}\, x_{n-2} \cdots x_0)$$

} n-bit Boolean functions

$\therefore$



$$f_{n+1}(x) = \bar{x}_n \wedge f_n^{(0)}(x) \oplus x_n \wedge f_n^{(1)}(x)$$

* Boolean circuits: Sequence of gates implementing a Boolean function.

* {AND, OR, NOT, INPUT} are universal.

(2) Complexity in the circuit model :-

- For an arbitrary Boolean fn, need,

$\leq 2^n$ OR gates + $n2^n$ AND gates + $n2^n$ NOT gates + $n2^n$ INPUT gates

$\leq (3n+1)2^n$ gates!

- Lower bound?!

Can show via a counting argument that we need atleast $(1-\varepsilon)\frac{2^n}{2n}$ gates to be able to implement most Boolean functions.

* Any Boolean fn. f can be thought of as the solution to a "Decision problem".

f : I/p $\longrightarrow$ {YES, NO}.

(Eg. PRIMES : $f(x) = \begin{cases} 1 & \text{if } 'x' \text{ is prime} \\ 0 & \text{otherwise} \end{cases}$)

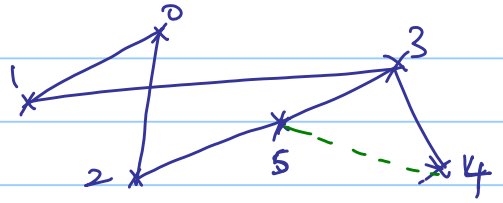
Non-trivial examples:-

(i) FACTORING :- $f(x,y) = \begin{cases} 1, & \text{if } 'x' \text{ has a divisor } z,\ 1 < z < y \\ 0, & \text{otherwise} \end{cases}$

Knowing $f(x,y)$ $\forall$ $y < x$ $\Leftrightarrow$ knowing the least non-trivial factor of $'x'$.

(ii) HAMILTONIAN CYCLE PROBLEM:-

- Graph: collection of vertices, connected by edges
  $(V_1, V_2, \ldots V_n)$ $\qquad$ $E = (V_i, V_j)$



Adjacency matrix A

A : nxn matrix

$a_{ij} = \begin{cases} 1 & \text{iff there's an edge } (V_i, V_j) \\ 0 & \text{otherwise} \end{cases}$

- Cycle : Sequence of vertices $(V_1, V_2, \ldots V_m)$ such that every pair (V_i, V_{i+1}) is an edge

- Hamiltonian cycle: A cycle which visits every vertex of the graph exactly once.

HAMILTONIAN CYCLE: $f(x) = \begin{cases} 1 & \text{if graph } x \text{ has a Hamiltonian path} \\ 0 & \text{otherwise} \end{cases}$

Note: These Boolean fns. are not restricted to I/Ps of a specific size:

- Decision problems represent a family of Boolean fns.

Eg. FACTORING: x, y are integers.

HAM. PATH: x is some $n \times n$ adjacency matrix.

$$f : \{0,1\}^* \longrightarrow \{0,1\}$$

Identifies a "language" L :-

$$L \equiv \left\{ \text{set of all } x \in \{0,1\}^* \text{ such that } f(x) = 1 \right\}$$

(All strings that satisfy f belong to L)

* Computational Problem $\Leftrightarrow$ Decision problem

$\Updownarrow$

Family $\{f_n\}$ of Boolean fns (and a language 'L')

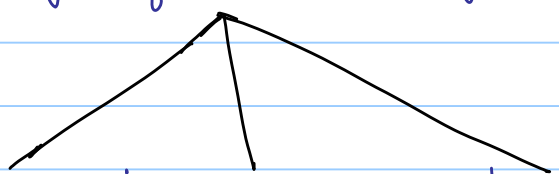
$\Updownarrow$

Circuit family $\{C_n\}$ that compute them.

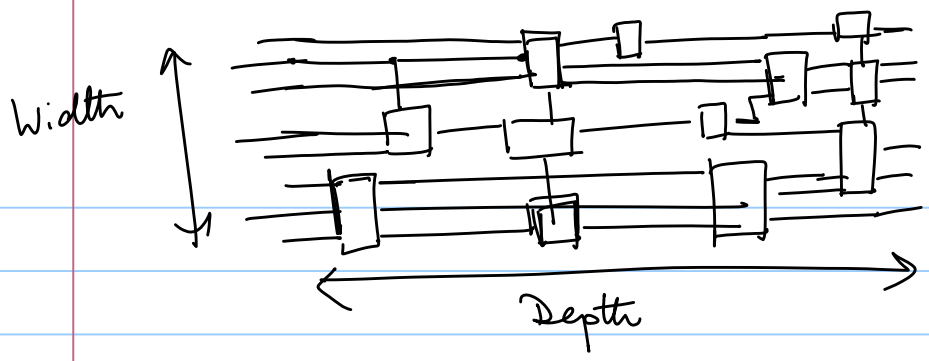
$\therefore$ "Hardness" of a problem

$\Updownarrow$

"Complexity" of Circuit family $\{C_n\}$



Circuit size	Time	Space
$N(n)$	"Depth"	"Width"
* How does # of gates scales with 'n'?	# of time-steps required to do the computation	Max # of gates in a given time step.



* Complexity classes :-

("Uniform" circuit family)

(a) $\{C_n\}$ is "polynomial size" if $|C_n|$ (size) grows no faster than any polynomial fn. of 'n'.

$\text{Size}(C_n) = |C_n| \leq \text{poly}(n)$

$$P \equiv \left\{ \begin{array}{l} \text{Decision problems solved by} \\ \text{polynomial size circuit families} \end{array} \right\}$$

Problems in P are "easy";
Problems not in P are "hard"!

* Extra reading: "PRIMES is in P"

Manindra Agrawal, Neeraj Kayal, Nitin Saxena } IIT-kanpur

Annals of Mathematics (2002)

* classification is independent of the circuit components (so long as the gate set is universal and finite)

* Most function families are not in P.