CS121 Lecture 5 September 17, 2019

Syntactic Sugar and Computing All Functions

Brief Review of O notation

$f(n) = O(g(n))$: $f \leq g$ ignoring constants and small n

$f(n) = o(g(n))$ if $f \ll g$ even if don't care about constant factors

Circuits

Theorem 1: Every function $f:\{0,1\}^n \to \{0,1\}$ can be computed by circuit size $O(2^n/n)$

Theorem 2: Some functions $f:\{0,1\}^n \to \{0,1\}$ cannot be computed by circuits of size $o(2^n/n)$

Theorem 3: If you can compute $f:\{0,1\}^n \to \{0,1\}$ in s cycles/secs, f can be computed by circuit of $\approx s$ gates

* if f outputs m bits then add factor m to Thm 1, 2

Today

Theorem 4.12 :
Theorem 4.14 ? $\forall f:\{0,1\}^n \to \{0,1\}^m$ there is a boolean circuit C computing f.

$|C| := size(C) \leq$ ~~$O(n \cdot 2^n \cdot m)$ $O(2^n \cdot m)$~~ $O(2^n \cdot m/n)$

Toolkit: "syntactic sugar" transformations to construct circuits/progs

For every n there is a circuit $O(n^{1.6})$ gates to compute the map $a, b \mapsto a \cdot b$ where a, b are n-bit numbers

EX1: Let $\delta_{101}: \{0,1\}^3 \to \{0,1\}$ defined as $\delta_{101}(x) = \begin{cases} 1, & x=101 \\ 0, & o/w \end{cases}$

Give boolean circuit to compute δ_{101}

X_0 AND NOT(X_1) AND X_2 = $X_0 \wedge \overline{X_1} \wedge X_2$

EX 2: compute boolean circuit to compute f

x	f(x)
000	0
001	1
010	0
011	0
100	1
101	0
110	0
111	1

$f(x) =$

$\delta_{001}(x) \vee \delta_{100}(x) \vee \delta_{111}(x)$

Theorem 4.12:

proof. Let $f_i: \{0,1\}^n \to \{0,1\}$ be the i^{th} bit of f

$f(f_i(x) = f(x)_i)$

computing $f_0 \ldots f_{m-1} \to$ computing f

$f(x) = \underbrace{\delta_{0^n}(x) \vee \delta_{0^{n-2}10}(x) \vee \ldots \delta_{1^n}(x)}$

at most 2^n copies of δ_{x_i}, each

computation by circuit of $n-1$ ANDs

and $\leq n$ NOTs

$\to$ size $\leq O(n \cdot 2^n)$ ■

Syntactic Sugar

Take programming language $P \to P{+}{+}$ by:

- adding extra features
- write "transpiler" that: P++ program and maps to P program w/ equivalent functionality

- for is syntactic sugar for while
- Define NAND-CIRC++ to include: if statements,

user-defined procedures, non-boolean value variables, arrays, etc.

ex: What does this compute

$NAND(NAND(NAND(x_0, x_0), x_2), NAND(x_1, x_0))$

$if(x_0): x_1$, $else: x_2$

```
If (cond) {
  temp = NAND(bar, blah)
  foo = IF(cond, temp, foo)
}
```

ex: define $LOOKUP_\ell(X, i) = X_i$ where $X \in \{0,1\}^{2^\ell}$ and $i \in \{0,1\}^\ell$

$$LOOKUP_1(x_0, x_1, i_0) = x_{i_0}$$

$$LOOKUP_2(x_0, x_1, x_2, x_3, i_0, i_1) = x_{i_0 i_1} = \begin{cases} x_0 & i_0=0, i_1=0 \\ x_1 & i_0=0, i_1=1 \\ x_2 & i_0=1, i_1=0 \\ x_3 & i_0=1, i_1=1 \end{cases}$$

$LOOKUP_1 = IF(i_0, x_0, x_1)$

$LOOKUP_2 = IF(i_0, IF(i_1, x_3, x_2), IF(i_1, x_1, x_0))$

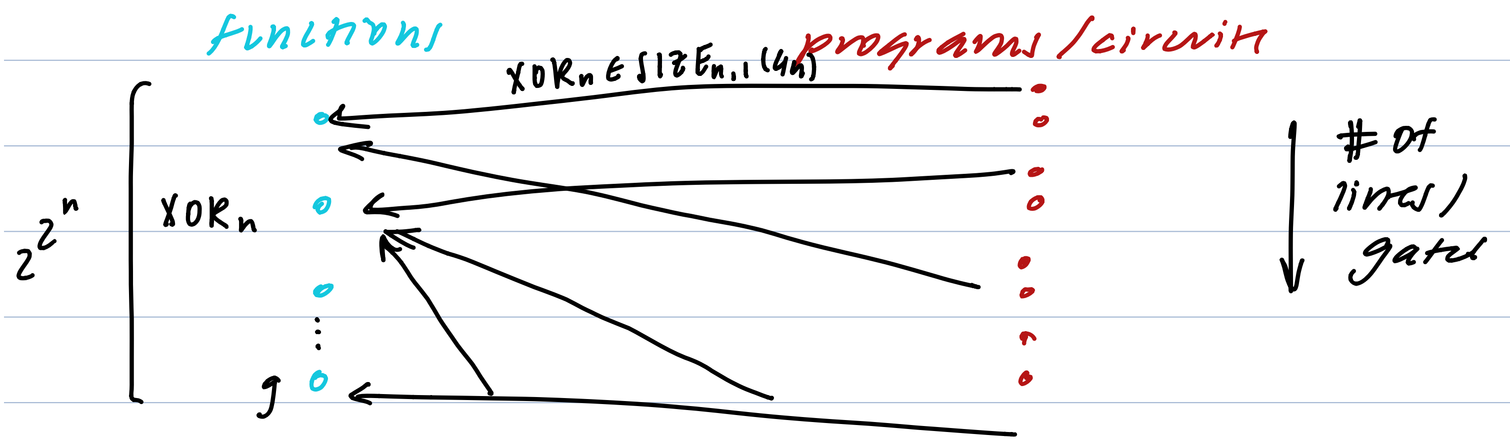
ex: recursive LOOKUP

$$LOOKUP_{\ell+1}(x, i) = \begin{cases} LOOKUP_\ell(x_0 \dots x_{2^{\ell}-1}, i_1, \dots, i_{\ell-1}) & i_0 = 0 \\ LOOKUP_\ell(x_{2^{\ell-1}}, \dots, x_{2^{\ell}-1}, i_1 \dots i_{\ell-1}) & i_0 = 1 \end{cases}$$

$size(LOOKUP_{\ell+1}) \le 2 \cdot size(LOOKUP_\ell) + 4$

$SIZE_{n,m}(s) = \{ f : \{0,1\}^n \to \{0,1\}^m \mid \exists C \; |C| \le s \text{ computes } f \}$

$SIZE(s) \subseteq SIZE(10 \cdot s)$



ex: let $one : \{0,1\} \to \{0,1\}$ be the function $one(a) = 1$ for $a \in \{0,1\}$ and let $zero : \{0,1\} \to \{0,1\}$ where $zero(a) = 0$. Give NAND circuits to compute one and zero

one: $NAND(a, NAND(a,a))$

zero: $NAND(b, b)$

Can add the function LOOKUP and the constants 0,1 to NAND-CIRC and get equivalent power.