CS3102 Theory of Computation

diagonalization

$$\underbrace{\#\text{ functions}}_{\text{uncountable}} > \underbrace{\#\text{ of implementations}}_{\text{countable}}$$

Warm up:

We showed that NAND gates are $\boxed{\text{“Universal”}}$. We also showed that not all things are “Computable”. How can both things be true?

functions

$\boxed{\text{finite}}$ infinite next

Logistics

- Midterm on Thursday in class
  - Review session tomorrow evening
    - 6:30pm – 8:00pm
    - Thornton E316

Last Time

- Complexity
  - SIZE
  - Complexity Classes
  - Big-Oh

How many gates are required?

- TCS Theorem 5.3: There is a constant $\delta > 0$, such that for every sufficiently large n there is a function $f: \{0,1\}^n \to \{0,1\}$ such that $f \notin SIZE\left(\frac{\delta 2^n}{n}\right)$. That is, the shortest NAND program to compute f requires at least $\delta \cdot \frac{2^n}{n}$ gates.

none

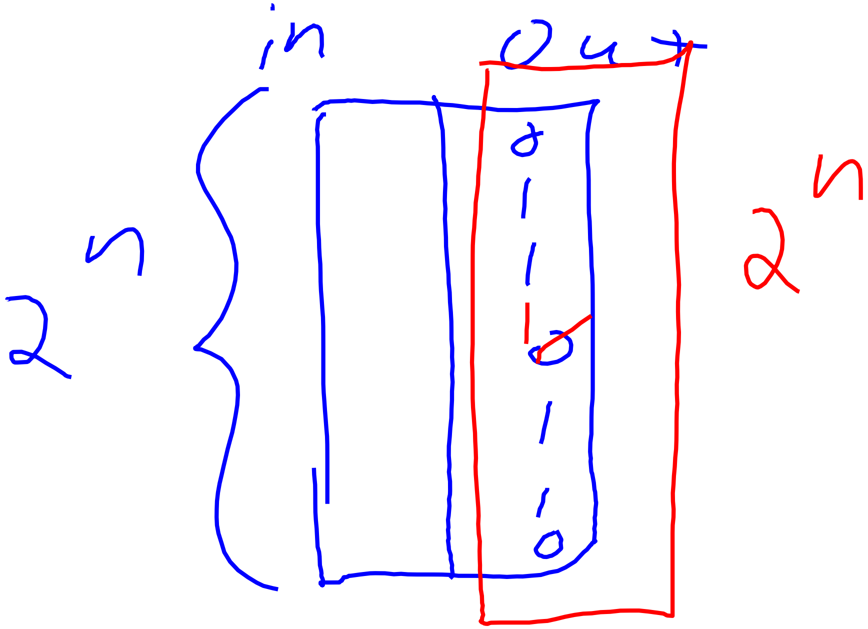
something

How to show this

1. Count the number of n- input functions
2. Count the number of programs of size $\delta \cdot \frac{2^n}{n}$
3. Show there are more functions than programs

How many functions? $= 2^{2^n}$

- How many functions are there of form $|\underline{\{0,1\}^n}| \to \{0,1\}$?
- How can we count this?



\# functions $= |\{0,1\}^{2^n}|$

straightline

How many programs?

- Bits required for an s-line program:
  - At most $3s$ variables (3 variables mentioned for each of the s lines)
    - $\log_2 3s$ bits per variable
  - 3 variables per line
    - $3 \cdot \log_2 3s$ bits per line
  - s lines total
    - $3s \log_2 3s$ bits total
- Upper bound on the number of s-line programs:
  - $\boxed{2^{3s\log_2 3s}}$
  - $\boxed{2^{O(s \log s)}}$ $\approx 2^{s \log s}$

s line program has

$\boxed{3s \log_2 3s \text{ bits}}$

Fixing the Length

$2^{O(s \log s)}$

- If we fix the length of the programs to be $\boxed{\delta \cdot \frac{2^n}{n}}$ lines, how many programs are there?
- $2^{c \cdot s \log s}$ programs of length s
- $2^{\frac{c\delta 2^n}{n} \log s}$ programs
- Let $\delta = \frac{1}{c}$
- $2^{\frac{2^n}{n} \log s} < 2^{2^n}$
- Some functions require more than $\delta \cdot \frac{2^n}{n}$ lines

$2^{c \cdot s \log s}$

$\boxed{2^{\frac{2^n}{n} \log s}} < 2^{2^n}$

64 bit machine

- I want to make $EVAL$ to evaluate any program for a function $f: \{0,1\}^{64} \to \{0,1\}$. How many gates do I need?
- Some functions will require at least $\delta \cdot \frac{2^n}{n}$ gates.
  - Assume $\delta = \frac{1}{10}$
- We must evaluate programs longer than: $\frac{2^{64}}{640}$ lines
- We need at least $\left(\frac{2^{64}}{640}\right)^2 \log_2\left(\frac{2^{64}}{640}\right)$ gates $s^2 \log s$
  - $\boxed{4.5 \times 10^{34} \text{ gates}}$ 1nm
  - Your computer would need to be the area of the solar system

Conclusion

- A domain of 2^{64} is large enough that perhaps it's not useful to think of the function as finite
- Let's think of that as an infinite function instead
- We need a model of computing for $\boxed{\text{infinite}}$ functions

do a finite function in a loop

After the exam

- A model of computing for infinite functions
- How to do simple operations over and over again to compute
  - Real computers update memory by computing “simple” functions in hardware over and over again

Major topics

- Representing things as strings
- Computing requires finite representations
  - There are more functions than finite representations of things, so some functions aren't computable!
- Boolean gates/programs as a model of computing
  - Computing with logic!
- Simple components can build complicated behavior
  - With just NAND we can do complex functions
    - ANY finite function, actually
    - Including evaluating programs
- With a model of computing we can measure efficiency of computing
  - Allows us to categoriz functions by difficulty

Is there a baby in the picture?

Connor

500,000 · 24 = K

Computing the "Baby" function

First: Represent pictures as strings (in binary)

– Assumption: all our pictures will be scaled to be the same size

1000 x 500 pixels

(0-255, 0-255, 0-255)

R G B

8 bits/color, 24 bits/pixel

Computing the "Baby" function

Second: Define the function

$$BABY: \{0,1\}^k \rightarrow \{0,1\}$$

$$BABY(p) = \begin{cases} 1 & \text{if } p \text{ has a baby in it} \\ 0 & \text{otherwise} \end{cases}$$

Computing the “Baby” function

Third: Build a NAND-circuit/program for the function

How can we tell if that’s possible?

it's finite

Questions on the Exam

1. Representing things in binary
    1. Naturals, integers, coordinates
2. Is the set countable
    1. Finite binary strings, onto mapping from naturals
3. Is the set uncountable
    1. Diagonalization, bijection (1-1) from a known uncountable (infinite binary strings)
4. Are these models of computation equivalent
    1. Take an implementation from each, and convert to an implementation of the same function in the other (AON = NAND, CIRC = Straightline)
5. Is this set of gates universal
    1. We can convert AON or NAND into these gates (AON = NAND, NOR=NAND)
6. How many gates are in this circuit
    1. Here's an implementation of some function, how many gates did it use? (LOOKUP, CMP,EQUIV)
7. Big-oh, omega, theta
    1. Is f in O(g)?