CS 121: Introduction to Theoretical Computer Science Fall 2019

Section 1

Prof. Boaz Barak

1 Representation

1. Can we come up with a representation of the complex numbers? Why or why not? What if we had an exact representation of the reals?

   No, because we can surject complex numbers onto the reals. If we had an exact representation of the reals, we could certainly do it with the same trick as using the rationals to represent the real and complex part of the number, or the polar coordinates (you can do whatever parametrization you want)!

2. Write down an encoding $E : \{0,1,2\}^* \to \{0,1\}^*$. Can you come up with a one-to-one function $S \to \{0,1,2\}^*$, where S is the set of n-tuples of natural numbers?

   Encoding $\{0,1,2\}^* \to \{0,1\}^*$ can simply map 0 to 00, 1 to 11, and 2 to 01.
   One-to-one function from S to $\{0,1,2\}^*$ is just writing out the natural numbers in binary, then appending a 2 between them.

3. For each of the following sets S, determine if there exists an encoding $S \to \{0,1\}^*$:
   (a) S is the set of infinite integer sequences that are uniformly zero after some point
   (b) S is the set of infinite integer sequences that are uniformly zero before some point

   (a) Yes (this is equivalent to an encoding for the set of finite integer sequences)
   (b) No (this is equivalent to an encoding for the set of infinite integer sequences)

2 Computability

1. Define CMP that on input (a,b,c,d), CMP function outputs 1 if the natural number represented by (a,b) is greater than the natural number represented by (d,e). Describe an AON-CIRC program computing the CMP function.

   We want to check if $2a+b > 2c+d$. Note that this is equivalent to $(a > b) \vee ((a = b) \wedge (b > d))$. For two bits a, b, we can compute $a > b$ with $\text{AND}(\text{XOR}(a,b), a)$ and we can compute $a = b$ with $\text{NOT}(\text{XOR}(a,b))$. Hence, we can write

   $$CMP(a,b,c,d) = \text{OR}(\text{AND}(\text{XOR}(a,c),a), \text{AND}(\text{NOT}(\text{XOR}(a,c)), \text{AND}(\text{XOR}(b,d),b))$$

   Recall that we can compute $\text{XOR}(a,b)$ with $\text{AND}(\text{NOT}(\text{AND}(a,b), \text{OR}(a,b))$, so we have all the pieces to implement the relevant AON-CIRC program.

2. The NOR operation to, on input (a, b), output 1 if $a = b = 0$ and 0 otherwise. Let NOR-CIRC be the programming language where we have just the NOR operation. Compare the power of AON-CIRC programs and NOR-CIRC programs (i.e. either show that there is some function that only one type of program can compute or if a function is computable by one type of program iff it is computable by the other).

   AON-CIRC and NOR-CIRC are equally powerful.

   To show that AON-CIRC is as powerful as NOR-CIRC, we just need to show that we can compute NOR with AND, OR, and NOT. We can compute NOR(a, b) with

$$\begin{aligned} \text{temp\_1} &= \text{OR}(\text{X}[0], \text{X}[1]) \\ \text{Y}[0] &= \text{NOT}(\text{temp\_1}) \end{aligned}$$

   The idea is that once we have this, if function f is computable by NOR-CIRC program P, we can replace every NOR operation with lines that execute the above combination of NOT and OR operations.

   To show that NOR-CIRC is as powerful as AON-CIRC, we need to show that we can compute AND, OR, and NOT with NOR. We can compute NOT(a) with

$$\text{Y}[0] = \text{NOR}(\text{X}[0])$$

   OR(a, b) with

$$\begin{aligned} \text{temp\_1} &= \text{NOR}(\text{X}[0], \text{X}[1]) \\ \text{Y}[0] &= \text{NOR}(\text{temp\_1}, \text{temp\_1}) \end{aligned}$$

   and AND(a, b) with

$$\begin{aligned} \text{temp\_1} &= \text{NOR}(\text{X}[0], \text{X}[0]) \\ \text{temp\_1} &= \text{NOR}(\text{X}[1], \text{X}[1]) \\ \text{Y}[0] &= \text{NOR}(\text{temp\_1}, \text{temp\_2}) \end{aligned}$$

   The idea is that once we have this, if function f is computable by AON-CIRC program P, we can replace AND, OR, and NOT NOR operations with lines that execute the above combinations of NOR operations.