CS 181 Lecture Notes

Kyle Chui

2022-09-28

Contents

1 Lecture 2

One of the big ideas of data representation is *composition*, e.g. we can take pairs of integers to represent the rationals.

1.1 Prefix-Free Encoding

Definition. *Prefix-Free Encoding*
We say that a function $E\colon \Theta \to \{0,1\}^*$ is *prefix-free* if for all $x, y \in \Theta$ where $x \neq y$, we have $E(x)$ *is not* a prefix of $E(y)$.

Example. Consider the $NtoB$ function, that converts natural numbers to their binary counterparts. Then we know that $NtoB$ is *not* a prefix-free encoding, since $NtoB(1) = 1$ is a prefix for $NtoB(3) = 11$.

Example. Consider the modified $\overline{ZtoB}$ encoding function (which duplicates each bit and adds 01 to the end). This is a prefix-free function, since the 01 for x must match with the 01 in y; then for x to be a prefix for y we must have $x = y$.

Prefix-free encodings are quite useful, since the bit representation is unique and unambiguous (we don't need any special characters to demarcate objects).

Theorem. Suppose we have a prefix-free encoding

$$E\colon \Theta \to \{0,1\}^*.$$

We define $\overline{E}\colon \Theta^* \to \{0,1\}^*$ by

$$\overline{E}((x_1, x_2, \dots, x_n)) = E(x_1) \circ E(x_2) \cdots E(x_n).$$

Then $\overline{E}$ is a valid encoding of Θ^*.

Proof. Suppose we have the binary sequence

$$\overline{E}((x_1, x_2, \dots, x_n)) = E(x_1) \circ E(x_2) \cdots E(x_n).$$

Then we define the following decoding algorithm:

Algorithm 1:

1 **while** *we still have remaining bits* **do**
2 Keep reading in bits until the sequence matches an encoding.
3 Once we find it, that is the data representation for our next object; go to step 2.

The above algorithm is a valid decoder, since our encoding is prefix-free and the data representation is unambiguous. ☐

Theorem. Suppose $E\colon \Theta \to \{0,1\}^*$ is an encoding. We claim that there exists a prefix-free encoding $pfE\colon \Theta \to \{0,1\}^*$ defined by the result of:

- Computing $E(a)$
- Replacing each 0 with 00, and each 1 with 11

- Appending 01 at the end

Then we have that pfE is a prefix-free encoding.

Proof. Let x have the binary representation $E(x) = b_0b_1 \dots b_n$. Then $pfE(x) = b_0b_0b_1b_1 \dots b_nb_n01$. Since the 01 may only occur at the end, we have that pfE is a prefix-free encoding. $\square$

We now have ways to:

- Represent integers as binary strings in $\{0,1\}^*$.
- Represent integers using a prefix-free encoding.
- Represent lists of integers.
- Represent lists of integers using a prefix-free encoding.
- Represent lists of lists of integers.

1.1.1 Efficiency of Prefix-Free Encodings

The length of $pfE(x)$ is $2 \cdot |E(x)| + 2$. There exists a different transformation where the new encoding has length

$$|E(x)| + 2 \cdot \log_2 |E(x)| + 2.$$

Can we represent real numbers as $\{0,1\}^*$? In other words, is there an injective mapping $E\colon \mathbb{R} \to \{0,1\}^*$?

Note (Limitations of Encoding). No, since the number of binary strings is countable and the number of real numbers is uncountable.

1.2 Algorithms

Definition. *Boolean Circuit*
Some computations with AND/OR/NOT as basic operations.

Definition. *Directed Acyclic Graph*
A directed graph that has no cycles.