CS 121 Section 2: Finite Computation and Computing Every Function

TF: Ife Omidiran
Overview

1. Defining computation
2. Computing every function
   ○ Practice question
3. Section exercises
Computation with Circuits

- A Boolean circuit with \( n \) inputs, \( m \) outputs, and \( s \) gates, is a labeled directed acyclic graph (DAG) with \( s + n \) vertices. There are exactly \( n \) of the vertices with no in-neighbors, which we call the inputs and label them with \( X[0], \ldots, X[n - 1] \).
- The other \( s \) vertices are gates, and each is labeled with \( \land \), \( \lor \) or \( \neg \). Gates labeled with \( \land \) (AND) or \( \lor \) (OR) have two in-neighbors, and those labeled with \( \neg \) (NOT) have one in-neighbor.
- Finally, exactly \( m \) of the gates are outputs, and are labeled with \( Y[0], \ldots, Y[m - 1] \).
Straight-Line Programs

- An AON-CIRC program is a string of lines of the form:
  
  ```
  foo = AND(bar,blah)  
  foo = OR(bar,blah)   
  foo = NOT(bar)       
  ```

  (where “foo”, “bar” and “blah” are variable names)

- Straight line program = no loops or branching
Universality: Computing Every Function

- Syntactic sugar: We can use NAND to achieve more advanced features, such as user defined procedures

```python
def Proc(a,b):
    proc_code
    return c
some_code
f = Proc(d,e)
some_more_code
```

```python
some_code
proc_code'
some_more_code
```
For every $k$, the lookup function of order $k$, $LOOKUP_k : \{0, 1\}^{2^k+k} \rightarrow \{0, 1\}$ is defined as follows:

For every $x \in \{0, 1\}^{2^k}$ and $i \in \{0, 1\}^k$,

$$LOOKUP_k (x, i) = x_i$$

where $x_i$ denotes the $i$-th entry of $x$, using the binary representation to identify $i$ with a number in $\{0, \ldots, 2^k - 1\}$. 
Computing $LOOKUP_2$

def LOOKUP_2(X[0], X[1], X[2], X[3], i[0], i[1]):
    a = LOOKUP_1(X[2], X[3], i[1])
    b = LOOKUP_1(X[0], X[1], i[1])
    return LOOKUP_1(b, a, i[0])
Theorem 4.10. For every $k > 0$, there is a NAND-CIRC program that computes the function $LOOKUP_k : \{0, 1\}^{2^k+k} \rightarrow \{0, 1\}$. Moreover, the number of lines in this program is at most $4 \cdot 2^k$.

Proof Idea: Compute $LOOKUP_k$ using $LOOKUP_{k-1}$

- For $k > 1$, we use a generalization of the code on the last slide
- $LOOKUP_1$ — we've shown how to compute this
Theorem 4.12. There exists some constant $c > 0$ such that for every $n, m > 0$ and function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, there is a NAND-CIRC program with at most $c \cdot m2^n$ lines that computes the function $f$.

Proof Idea: We’ve shown that we can compute $LOOKUP_n$ using syntactic sugar. For any finite function, we can enumerate all possible inputs and outputs in a table, and look up the output corresponding to the given input!
Review: SIZE class of functions

- \( \text{SIZE}(s) \) is the set of functions that can be computed by a NAND-CIRC program of at most \( s \) lines.
- Examples:
Practice: Prove that $SIZE$ is closed under complement. That is, show that if $f$ is in the class $SIZE_n(s)$, then the complement of $f$, which is the function $g(x) = 1 - f(x)$, must be in the class $SIZE_n(s + 1)$. 
Exercise 1

a. Show that AND and NOT are a universal gate set. (Hint: Use the definition of universality.)

b. Show that AND and OR are not a universal set of operations. (Hint: Think about the properties of functions that can be computed using AND/OR. What if all of the inputs to the function are 1?)
Let \textsc{IF-CIRC} be the programming language where we have the following operations: 
\texttt{foo = 0}, \texttt{foo = 1}, \texttt{foo = IF(cond, yes, no)}; that is, we can use the constants 0 and 1, and the \textsc{IF}: \{0, 1\}^3 \rightarrow \{0, 1\} function such that \textsc{IF}(a, b, c) equals \texttt{b} if \texttt{a} = 1 and equals \texttt{c} if \texttt{a} = 0).

Show that \textsc{AON-CIRC} is as powerful as \textsc{IF-CIRC}, and vice versa.\(^1\)

\textbf{Exercise 2}

\begin{itemize}
  \item Let \textsc{IF-CIRC} be the programming language where we have the following operations: 
    \texttt{foo = 0}, \texttt{foo = 1}, \texttt{foo = IF(cond, yes, no)}; that is, we can use the constants 0 and 1, and the \textsc{IF}: \{0, 1\}^3 \rightarrow \{0, 1\} function such that \textsc{IF}(a, b, c) equals \texttt{b} if \texttt{a} = 1 and equals \texttt{c} if \texttt{a} = 0).
  \item Show that \textsc{AON-CIRC} is as powerful as \textsc{IF-CIRC}, and vice versa.\(^1\)
\end{itemize}

\textit{Hint:} The \textsc{LOOKUP} function is closely related to \textsc{IF}.
Exercise 3

In the proof presented for the universality of NAND, we mentioned, but didn’t prove, that the lookup function is in $\text{SIZE}(4 \cdot 2^k)$. Prove that $\text{LOOKUP}_k$ can indeed be computed using a circuit of at most $4 \cdot 2^k - 1$ gates.