Homework 14

Joe Baker, Brett Schreiber, Brian Knotten

February 14, 2018

21

a

Consider that every Boolean circuit can be represented by AND, OR, and NOT gates.
Consider a Boolean circuit C containing n gates. There exists a corresponding Boolean formula F of size n, because AND, OR, and NOT gates can be represented in a Boolean formula as logical operators $\wedge$'s, $\vee$'s, and $\neg$'s. Since there is a one-to-one mapping of gates to logical operators, there exists a Boolean formula F with an output equivalent to C of size n.

Any Boolean function F for an input size n can be encoded into a string x of size $|x| = S = 2^n$, where each bit represents an output for a given input. For example, if $n = 4$, the first bit of x, either a 0 or 1, represents the output for the input 0000. The second bit of x represents the output for input 0001... and the last bit of x represents the output for input 1111.

The formula for F can be naively implemented through the following algorithm:

List all binary numbers of length n from 0 to 2^n. Let B be this list. For each number b in B:
    for each bit b_i in b:
        if $b_i = 0$, write out $\neg x_i$
        if $b_i = 1$, write out x_i
    Join together all of these boolean expressions with $\wedge$'s
Join together all of these boolean expressions with $\vee$'s

Here is an example Boolean function f encoded as 1011. This means f has the following outputs:
$f(00) = 1$
$f(01) = 0$
$f(10) = 1$
$f(11) = 1$

The corresponding boolean formula using the naive implementation is: $f(x_1x_2) = (\neg x_1 \wedge \neg x_2) \vee$
$\neg(\neg x_1 \wedge x_2) \vee$
$(x_1 \wedge \neg x_2) \vee$
$(x_1 \wedge x_2)$

So every boolean function on n bits has a formula of size at most 2^n. Since $\wedge$, $\vee$, and $\neg$ exist as gates in circuits, this formula can be converted into a circuit with size at most 2^n gates.

b

First, show that f computed by circuit $|C| = S \Rightarrow f$ computed by S-line program:
For each Boolean gate (NOT, AND, OR) in C, there exists an equivalent straight-line program statement with a left-side assignment variable corresponding to the output of the gate and a right side consisting of either a boolean operation ($\wedge$, $\vee$) performing the gate's operation on input variables corresponding to the input of the gate, or the negation ($\neg$) of an input variable corresponding to the input of the gate.
Therefore, given a circuit $|C| = S$, we can construct an equivalent straight-line program with S statements by iterating over each level of C and converting each gate to an equivalent straight-line program statement. Note that C will have at most S levels as it has S inputs.

Next, show that f computed by S-line program $\Rightarrow$ f computed by circuit $|C| = S$:
For every straight-line program statement there exists an equivalent Boolean gate (NOT, AND, OR) with inputs corresponding to the right-side variable(s) of the statement that performs the corresponding Boolean operation ($\neg$, $\wedge$, $\vee$) of the right-side of the statement and an output corresponding to the left-side assignment variable of the statement.
Therefore, given a straight-line program with S statements, we can construct an equivalent circuit $|C| = S$ by iterating over each statement and converting it to an equivalent Boolean gate.

Thus, f can be computed by a circuit $|C| = S \Leftrightarrow f$ can be computed by a S-line straight-line program.

22

Problem:
Show that there are some Boolean functions $f : \{0,1\}^n \to \{0,1\}$ that can be computed with n^4 gates but not with n^2 gates.

Proof:
Let $f : \{0,1\}^x \to \{0,1\}$ be a Boolean function that cannot be computed with a circuit of $\frac{2^x}{x}$ or less gates. We know from Shannon's proof of Theorem 6.21 in the book that such a function exists. We also know from class that a circuit of size $x * 2^x$ gates or less can compute f.

Let $x = 3 * log(n)$ and define $g : \{0,1\}^n \to \{0,1\}$ as a Boolean function that runs f for the first x bits of input.

Then the size of g uses less than n^4 gates to compute f since a circuit of size $x * 2^x$ gates or less can compute f and $(3 * log(n)) * 2^{3*log(n)} = 3 * log(n) * n^3 \in O(n^4)$. Additionally, g uses more than n^2 gates to compute f since f cannot be computed with less than $\frac{2^x}{x}$ and $2^{3*log(n)}/(3 * log(n)) = n^3/(3 * log(n)) \notin O(n^2)$. Thus f is a function which can be computed with n^4 gates but not n^2 gates.