MATH 262A LECTURE 1: AN INTRODUCTION TO BOOLEAN CIRCUITS AND FORMULAE

SCRIBE: CHRISTIAN WOODS

1. Administrivia

MATH 262A - Circuit Complexity

WF 3:30 - 5:00, APM B412

Instructor: Dr. Sam Buss, `sbuss@math.ucsd.edu`

Course Website: `http://www.math.ucsd.edu/~sbuss/CourseWeb/Math262A_2013F/`

Temporary Office Hour: Monday, September 30, 2:30 - 3:30

No course meeting on Friday, November 8

2. Basic Structures and Examples

Boolean functions

- computed by Boolean circuits or Boolean formulae
- take on binary values of True/False, 0/1, etc.

Gates

- logical connectives, like
- $\wedge$
- $\vee$
- $\oplus$
- $\neg$
- $\leftrightarrow$

Date: September 27, 2013.

As an example, let us write the Boolean function $x \oplus y$ using only the connectives $\wedge$, $\vee$, and $\neg$:

$$x \oplus y = (x \vee y) \wedge \neg(x \wedge y).$$

We can write this as a circuit in the following way:

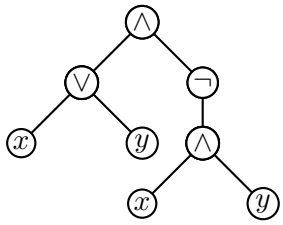

FIGURE 1. Circuit (1).

Notice that both the circuit and the written formula use the same number of logic gates. This need not always be the case, because a circuit (unlike a formula) may reuse a subformula. An example where this can happen is the function

$$Parity_3(x, y, z) := x \oplus y \oplus z.$$

We may use the following circuit to compute the function:

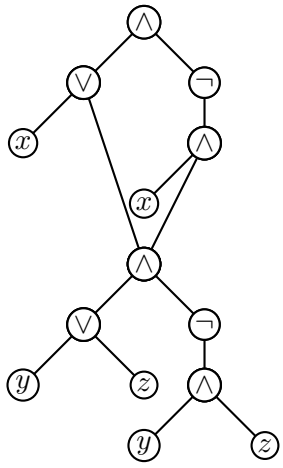

FIGURE 2. Circuit (2).

This depiction is a circuit, but *not* a formula.

We observe the following

Fact. *$Parity_n(x_1, x_2, \ldots, x_n) := x_1 \oplus x_2 \oplus \ldots \oplus x_n$ has circuits of size $O(n)$.*

Proof. By a simple induction argument, we need only add on a copy of circuit (1) to the output of the circuit for $Parity_{n-1}(x_1, x_2, \ldots, x_{n-1})$ and x_n to achieve the desired circuit. □

Our naïve formula for $Parity_n(x_1, \ldots, x_n)$ has size $O(2^n)$. We can actually improve this to get a formula of size $O(n^2)$ by noticing that

$$Parity_n(x_1, \ldots, x_n) = Parity_{\frac{n}{2}}(x_1, \ldots, x_{\frac{n}{2}}) \oplus Parity_{\frac{n}{2}}(x_{\frac{n}{2}+1}, \ldots, x_n).$$

3. Introductory Definitions and Theorems

Definition.
- *A (family of) Boolean function(s) is a mapping* $f : \{0,1\}^* \to \{0,1\}$.
- *An n-ary Boolean function is a mapping* $f : \{0,1\}^n \to \{0,1\}$.

Notice that we can decompose a Boolean function into a class $\{f_0, f_1, f_2, \ldots\}$ of n-ary Boolean functions, where f_k is k-ary.

Definiton. *A basis Ω is a set of Boolean functions.*

Definition. *A Boolean circuit over a basis Ω is a directed acyclic graph (DAG) such that for each vertex v in the graph, either*

(i) v has indegree 0 and is labeled with an input "x_i" for some $i \in \mathbb{N}$

or (ii) v has indegree $n > 0$ and is labeled with $g \in \Omega$, where g is an n-ary Boolean function and with an ordering on the incoming edges to v.

Usually a Boolean circuit has a single ouput, represented with a gate with outdegree 0.

Note: If a circuit has inputs among $x_1, \ldots, x_n$ it computes an n-ary function $f : \{0,1\}^n \to \{0,1\}$ in the "obvious" way.

Definition. *A gate is a non-input vertex in a Boolean circuit.*

Definition. *A Boolean circuit is a Boolean formula if each gate has outdegree less than or equal to 1.*

We may think of a Boolean formula as 'a directed tree, except perhaps at the leaves (since an input x_i can occur at multiple leaves).

Definition. *The size of a Boolean circuit is the number of gates in the circuit.*

To discuss Boolean circuits, it is necessary to choose a basis of Boolean functions. Some common choices that are studied are

- $\Omega = \{\wedge, \vee, \neg\}$, the DeMorgan basis

- $\Omega = B_2$, the set of all functions $f : \{0,1\}^2 \to \{0,1\}$
- $\Omega = \{\neg,$ unbounded fan-in $\vee,$ unbounded fan-in $\wedge\}$, where an *or* or *and* gate may take in any finite number of inputs.

A useful observation is

Theorem. *There are 2^{2^n} n-ary Boolean functions (i.e., $|B_n| = 2^{2^n}$).*

Proof. The size of the domain of an n-ary Boolean function is 2^n. For each input we must specify one of 2 outputs, so there are 2^{2^n} possible functions in total. □

When we do not wish to count $\neg$ gates in the size of a given circuit over a basis containing $\neg$, we will call the resulting count the $*$-size of the circuit.

Theorem. *Every n-ary Boolean function has a circuit (and formula) of $*$-size at most $n2^n$ over the basis $\Omega = \{\wedge, \vee, \neg\}$.*

Proof. Let f be an n-ary Boolean function. The truth table for f has 2^n rows. For each row of the truth table let us define a vector $\vec{a}$ in $\{0,1\}^n$ where $a_i = 1$ if and only if $x_i = 1$ in that row.

Define the symbol

$$x_i^{a_i} = \begin{cases} x_i & \text{if } a_i = 1 \\ \neg x_i & \text{if } a_i = 0. \end{cases}$$

Then

$$f(x) = \bigvee_{\vec{a} s.t. f(\vec{a})=1} (x_1^{a_1} \wedge x_2^{a_2} \wedge \ldots \wedge x_n^{a_n}).$$

Let us count how many $\wedge$ gates and $\vee$ gates it takes to create the circuit (which is actually a formula) corresponding to this symbolic representation of f. We see that there are at most 2^n $\vee$'s required, because this is the number of rows of the truth table. For each of these 2^n rows, we need $n-1$ $\wedge$'s to connect the individual inputs. So at most $2^n(n-1)$ gates are required. In total, the $*$-size of such a formula is

$$2^n + 2^n(n-1) = n2^n.$$

□

Note: This is not an optimal size. In fact, we can easily reduce the upper bound by a factor of 2 with a simple observation: if there are at most half of the rows of f with ouput 1, then we need only use 2^{n-1} rows (instead of 2^n) in the calculations above. Otherwise, there are at most half false rows. In this case, we create the formula for the negation of f in the style of the proof. Then we add a final $\neg$ gate, which does not contribute to the $*$-size.

Definition.
- *The C_Ω-size of a Boolean function f is the size of the smallest circuit over basis Ω that computes f.*
- *The L_Ω-size of a Boolean function f is the size of the smallest formula over basis Ω that computes f.*

Theorem (Shannon, 1949). *Most n-ary Boolean functions have C_{B_2}-size at least $\frac{2^n}{n}$ for n sufficiently large.*

Proof. Let $F(s,n)$ be the number of n-ary Boolean functions that can be computed by a B_2 circuit of size s.

To completely describe a circuit of size s we require a specification of s gates, each of which can be one of $|B_2| = 16$ types, each having two inputs. These inputs could come from the $s-1$ gates that have positive outdegree, or from the original n inputs to the function f. Thus there are at most $s+n-1$ total choices for an input to a gate.

Therefore there are at most $16(s+n-1)^2$ descriptions for a single gate. We also note that the "order" of the gates does not matter, so long as the inputs are well-defined. So the number of different possible circuits of size s is bounded by

$$\begin{aligned} F(s,n) &\leq \frac{\left(16(s+n-1)^2\right)^s}{s!} \\ &= \frac{16^s(s+n-1)^{2s}}{s!} \\ &\lesssim \frac{16^s(s+n-1)^{2s}}{s^s/e^s} \\ &= \frac{(16e)^s s^{2s}\left(1+\frac{n-1}{s}\right)^{2s}}{s^s} \\ &\leq (64e)^s s^s \\ &= c^s s^s \end{aligned}$$

for some constant c.

Now let $s = \frac{2^n}{n}$. Then the above gives

$$\begin{aligned} F\left(\frac{2^n}{n}, n\right) &\leq c^{\frac{2^n}{n}} \left(\frac{2^n}{n}\right)^{\frac{2^n}{n}} \\ &= \left(\frac{c}{n}\right)^{\frac{2^n}{n}} (2^n)^{\frac{2^n}{n}} \\ &= \left(\frac{c}{n}\right)^{\frac{2^n}{n}} \left(2^{2^n}\right) \\ &= o\left(2^{2^n}\right). \end{aligned}$$

But 2^{2^n} is the number of n-ary Boolean functions. This proves Shannon's Theorem. $\square$

A similar theorem for formulae was proved earlier by Riordan and Shannon, which we will discuss next lecture:

Theorem (Riordan-Shannon, 1942). *Let n be sufficiently large and fix $\delta < 1$. Then most n-ary Boolean functions have L_{B_2}-size at least $\frac{\delta 2^n}{\log n}$.*

Exercise: Prove that the Halting Problem does not have polynomial-size circuits. (Here the Halting Problem is defined as the set of Turing machines that halt on blank input, using an efficient encoding of Turing machines by Gödel numbers.)

Future Topics:

- All n-ary functions have $\{\wedge, \vee, \neg\}$-circuits of $*$-size $\frac{2^n}{n}(1 + o(1))$
- All n-ary functions have $\{\wedge, \vee, \neg\}$-formulae of $*$-size $\frac{2^{n+1}}{\log n}(1 + o(1))$