Barrington's Theorem

Arefin Huq

October 25, 2010

1 Overview

Recall that a *branching program* is a directed acyclic graph with the following properties:

1. Every edge is labeled 0 or 1.
2. Every node that has no outgoing edges is labeled with an output value (e.g. accept or reject).
3. Every node that has outgoing edges is labeled with a binary variable x_i and has at least one outgoing edge labeled 0 and at least one outgoing edge labeled 1.
4. One node is designated the start node.

Note that a given input variable x_i can appear many times as a node label. If we require that non-output nodes have out-degree 2 then the branching program is deterministic, and a given setting of the $\{x_i\}$ determines a single path through the program.

A *constant-width* branching program is a branching program where all the nodes are organized into layers such that the following additional properties hold:

1. The start node is in the first layer.
2. All the output nodes are in the last layer.
3. Each layer (except the first) only has incoming edges from the previous layer.
4. Each layer (except the last) only has outgoing edges to the next layer.
5. The number of nodes in any layer is at most a constant.

The maximum number of nodes in any layer is the *width* of the branching program. It was conjectured that functions such as majority require superpolynomial size constant-width branching programs. This conjecture was disproved by the following theorem:

Theorem 1 (Barrington)**.** *A language L has a polynomial size, width 5 branching program iff $L \in NC^1$.*

We will prove that any $L \in \mathrm{NC}^1$ has a polynomial size width 5 branching program. The other direction is much easier and is a homework problem. To prove this direction we will establish properties of a related class of programs called *permuting branching programs* and show how to turn a permuting branching program into a branching program. First we review some properties of permutations.

2 Properties of Symmetric Groups

2.1 Definition of a Group

Recall that a *group* is a set A combined with an operation $\cdot$ with the following properties:

1. (Closure) For all $x, y \in A$, $x \cdot y$ is in A.
2. (Associativity) For all $x, y, z \in A$, $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.
3. (Identity) There is an $e \in A$ such that for all $x \in A$, $e \cdot x = x \cdot e = x$.
4. (Inverses) For each $x \in A$ there exists a $y \in A$ such that $x \cdot y = y \cdot x = e$.

For convenience we will usually drop the operator symbol $(\cdot)$ and denote $x \cdot y$ by xy.

2.2 The Symmetric Group S_n

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. A *permutation* of $[n]$ is a function $\sigma : [n] \to [n]$ that maps each element of $[n]$ to a unique element of $[n]$. For example if $\sigma(1) = 3, \sigma(2) = 2, \sigma(3) = 1$ then σ is a permutation of $[3]$. It is easy to see that every permutation on $[n]$ is a bijection from $[n]$ to itself, and vice versa.

S_n is called the *symmetric group on n elements* and is defined to be the set of all permutations of $[n]$, combined with function composition as the group operation. Clearly the composition of two functions from $[n]$ to itself is also a function from $[n]$ to itself. It can be seen that S_n satisfies the group properties:

1. Closure: The composition of two bijections is a bijection.
2. Associativity: Function composition is associative.
3. Identity: The permutation $\sigma(i) = i$ is the identity.
4. Inverses: Every bijection can be inverted.

2.3 Cycles

Consider the following permutation on 6 elements:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 2 & 6 & 1 & 4 & 3 \end{pmatrix}$$

The notation indicates that each element in the top row is mapped to the element directly beneath it in the bottom row. Consider what happens to the element 1 as this permutation is applied repeatedly: $\sigma(1) = 5, \sigma(5) = 4, \sigma(4) = 1$, or put another way: $1 \to 5 \to 4 \to 1$. Likewise, $5 \to 4 \to 1 \to 5$, etc. We say that the elements $1, 5, 4$ form a 3-cycle and we denote this cycle as (154). Note that (541) and (415) denote the same cycle. In general $(x_i \dots x_n)$ denotes the cycle $x_1 \to x_2 \to \dots x_n \to x_1$.

Now consider the first element of [6] not contained in this cycle. This is 2 and we discover that 2 is in a cycle by itself: (2). Proceeding in this fashion we discover one more cycle: (36). Since any single element cycle has no effect, we can ignore the cycle (2) and write σ as the *product* (i.e. composition) of cycles as:

$$\sigma = (154)(36)$$

Notice that the two cycles are disjoint, that is they contain no elements in common. In fact, every permutation can be written as the product of disjoint cycles, by using the procedure outlined above.

2.4 Conjugates

If A is a group and $x \in A$, we say for every $y \in A$ that the element yxy^{-1} is a *conjugate* of x. Observe that if a is a conjugate of b then b is a conjugate of a.

Lemma 1. *If $\sigma, \tau \in S_n$ are both k-cycles then they are conjugates of each other. In other words, there exists $\gamma \in S_n$ such that $\tau = \gamma\sigma\gamma^{-1}$.*

Proof. Let $\sigma = (s_1 s_2 \dots s_k)$ and $\tau = (t_1 t_2 \dots t_k)$. Consider the following permutation:

$$\gamma = \begin{pmatrix} s_1 & s_2 & \dots & s_k \\ t_1 & t_2 & \dots & t_k \end{pmatrix}$$

This gives rise to the following equations:

$$\begin{aligned} \sigma(s_i) &= s_{i+1} \\ \tau(t_i) &= t_{i+1} \\ \gamma(s_i) &= t_i \\ \gamma^{-1}(t_i) &= s_i \end{aligned}$$

where in the first two equations $(i+1)$ is computed mod n.

Now consider the action of $\gamma\sigma\gamma^{-1}$:

$$\gamma(\sigma(\gamma^{-1}(t_i))) = \gamma(\sigma(s_i)) = \gamma(s_{i+1}) = t_{i+1} = \tau(t_i)$$

Therefore $\tau = \gamma\sigma\gamma^{-1}$. Note that γ is not unique since the expression of σ and τ as cycles is not unique. $\square$

2.5 The Commutator

If A is a group and for some $a, b \in A$ it is the case that $ab \neq ba$ then we say that A is not *commutative*. If we take the expression $ab = ba$ and multiply both sides on the right by $a^{-1}b^{-1}$ we get $aba^{-1}b^{-1} = e$. The term $aba^{-1}b^{-1}$ is called the *commutator* of a and b, and as we just saw, $ab \neq ba \iff aba^{-1}b^{-1} \neq e$.

3 Permuting Branching Programs

A *k-permuting branching program* (k-PBP) is a sequence of instructions such as the following:

$$\begin{array}{l} (x_{i_0}, \alpha_{i_0}, \beta_{i_0}) \\ \vdots \\ (x_{i_m}, \alpha_{i_m}, \beta_{i_m}) \end{array}$$

where x_{i_j} is one of the input variables, and $\alpha_{i_j}, \beta_{i_j}$ are cyclic permutations in S_k. The jth instruction evaluates to α_{i_j} if x_{i_j} is 0 and β_{i_j} if x_{i_j} is 1. That is, each instruction outputs one of two different permutations based on the value of x_{i_j}. The output of the program is the product of the permutations produced by each instruction. Let p be a k-PBP and let $p(x)$ denote the output of p on input $x = (x_1, \ldots, x_n)$.

Definition 1. *We say that p σ-accepts a language B if $p(x) = \sigma$ whenever $x \in B$ and $p(x) = e$ (the identity permutation) whenever $x \notin B$.*

The following theorem states that if σ is a k-cycle we can replace σ with another k-cycle.

Lemma 2. *If p σ-accepts B, σ is a k-cycle, and τ is another k-cycle, then there is another k-PBP of the same size that τ-accepts B.*

Proof. Use Lemma 1 to find γ such that $\tau = \gamma\sigma\gamma^{-1}$. Multiply both permutations in the first instruction of p by γ and multiply both permutations in the last instruction of p by γ^{-1}. $\square$

Lemma 3. *If p σ-accepts B there is another k-PBP of the same size that σ-accepts $\overline{B}$.*

Proof. Use Lemma 2 to create a k-PBP q of the same size that σ^{-1}-accepts B. Then multiply both permutations in the last instruction of q by σ. □

Lemma 4. *If p σ-accepts B and q σ-accepts C then there is a permuting branching program of size $2(size(p) + size(q))$ that $\sigma\tau\sigma^-1\tau^{-1}$-accepts $B \bigcap C$.*

Proof. Use Lemma 2 to get a σ^{-1}-acceptor for B and a τ^{-1}-acceptor for C. Combine them in the order $\sigma\tau\sigma^{-1}\tau^{-1}$. □

It may happen that $\sigma\tau\sigma^{-1}\tau^{-1}$ is the identity permutation. The following lemma resolves this issue:

Lemma 5. *There are cyclic permutations in S_5 that do not commute.*

Proof. (12345) and (13254) have this property. Their commutator is a cyclic permutation. □

4 Conclusion of Proof

Suppose $L \in \mathrm{NC}^1$ is decided by a depth d circuit C. Transform the circuit to C' so that it only contains AND and NOT gates. Use the preceding section to construct a 5-PBP that computes C' with length at most 4^d. Since $d = O(\log n)$ this gives a polynomial size PBP.

Turn the 5-PBP p into a width-5 branching program r as follows: each instruction in $(x_{i_j}, \alpha_{i_j}, \beta_{i_j})$ in p is a layer in r. Connect the outgoing edges labeled 0 based on α_{i_j} and the outgoing edges labeled 1 based on β_{i_j}. Create a new start node and connect it to any of the nodes in the first layer. Mark the corresponding node in the last layer as rejecting and mark all the other nodes in the last layer as accepting.

5 Acknowledgments

This material is adapted from [1], [2], [3].

References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 1 edition, April 2009.

[2] Ravi B. Boppana and Michael Sipser. The complexity of finite functions. pages 757–804, 1990.

[3] Charles Pinter. *A Book of Abstract Algebra.* McGraw-Hill Publishing Company, 1982.