Monotone Separation of Logspace from $NC^1$

Michelangelo Grigni* 
Michael Sipser*

Department of Mathematics
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

We show that the monotone analogue of logspace computation is more powerful than monotone log-depth circuits: monotone circuits for a certain function in monotone logspace require depth $\Omega(\log^2 n)$.

This classification scheme for monotone functions inherits much of the naturalness and robustness of the more familiar nonmonotone scheme. Most of the familiar containments still go through, because the simulations used to prove those containments are monotonicity preserving. Differences in the structure of the two schemes highlight simulations which do not preserve monotonicity. For example the inductive counting technique used to prove that $NL = co-NL$ [I, S] cannot be replaced with a monotone simulation since $mNL \neq co-mNL$ (below). We see the further elucidation of the structure of the monotone classification scheme to be an important goal of complexity theory.

In this paper we prove that $mNC^1 \neq mL$. This result shows that the process of pointer jumping, i.e., following a chain of pointers to the end, cannot be simulated by a monotone $NC^1$ circuit. Our proof is based upon the communication game method of Karchmer and Wigderson, although since we must work with a function in $mL$ rather than $mNL$, the argument differs in a number of essential ways.

Let $mSAC^1$ be the class of polynomial size $O(\log n)$ depth monotone circuits with bounded fanin AND gates and unbounded fanin OR gates. For a given class of monotone boolean functions $mC$, let $co-mC$ denote the class of dual functions $co-f(x) = \neg f(\neg x)$ where $f \in mC$. Then a careful review of Karchmer and Wigderson's proof reveals a stronger result: $mNL \subsetneq co-mSAC^1$, i.e. $co-mSAC$ circuits for $ustconn$ require depth $\Omega(\log^2 n)$. Since $ustconn \in mNL \subset mSAC^1$, it follows that $mNL \neq co-mNL$, and that $mL$ is strictly contained in $mNL$. Hence the present separation is strictly stronger than theorem 4 above.

1 Introduction

In recent years there have been several strong results in monotone complexity theory. Razborov's theorem [R1] showing that the clique function requires superpolynomial size monotone circuits solved a long-standing open question. More recently the result of Karchmer and Wigderson [KW] showed that the $st$-connectivity function requires superlogarithmic depth. These results may be viewed as proving monotone analogues of separations believed to be true in general nonmonotone complexity.

This idea was made precise in [GS]. They propose monotone analogues of standard complexity classes. Such a monotone class, for example denoted $mC$, is in general different from those functions in the general class that happen to be monotone, denoted $C \cap \text{mono}$. In this framework many theorems about monotone complexity may be conveniently restated. For example:

Theorem 1 [R1] $mP \neq mNP$.

Theorem 2 [R2] $mP \neq P \cap \text{mono}$.

Theorem 3 [AG] $\text{mAC}^0 \neq AC^0 \cap \text{mono}$.

Theorem 4 [KW] $mNC^1 \neq mL$.

Theorem 5 [Y] $\text{mTC}^0 \neq mNC^1$.

Theorem 6 [RW2] $NC^1 \cap \text{mono} \not\subset mNC$.

2 Monotone Logspace

We do not know of a Turing machine model for $mL$ (unlike the case for $mNL$), so instead we rely on a circuit model. In the nonmonotone nonuniform case we...
know that \( L \) is equivalent to polynomial size log-width circuits. Since our lower bound ignores uniformity, we define \( mL \) to mean monotone polynomial size log-width circuits. Note that \( mNC^1 \subseteq mL \).

We consider a certain monotone boolean function \( \text{fork} \) (the name follows from the appearance of the minterms). There is a directed graph with a distinguished vertex \( s \). Given as input the adjacency matrix \( A \) of the graph, the function \( \text{fork}(A) \) is true if and only if there is a directed path from \( s \) to some node with outdegree at least two. As a special case, it may be that \( s \) itself has outdegree at least two.

A simple construction shows that \( \text{fork} \) is in \( mL \), and is in fact it is complete for \( L \) (although not known to be complete for \( mL \)). It will suffice for our lower bound to deal with the undirected version \( \text{ufork} \): given an undirected adjacency matrix, \( \text{ufork} \) is true if and only if there is a path connecting \( s \) to some node of degree at least three, or if \( s \) itself has degree at least two.

In the following we show that monotone circuits for the \( \text{ufork} \) function require depth \( \Omega(\lg^2 n) \).

### 3 Communication Complexity

We briefly review the communication complexity method developed in [KW] and applied in [RW1, RW2].

Given a monotone boolean function \( f \) on \( n \) variables, define a two player communication game. One player is given a minterm of the function (a set of variables that force the function to 1 if they are all set to 1) and the other player is given a maxterm of the function (a set of variables that force the function to 0 if they are all set to 0). The players follow some agreed-upon deterministic protocol of communicating bits back and forth until in the end they have agreed on a variable in both the minterm and the maxterm; such a variable always exists. Because of an isomorphism between protocols for this game and monotone fanin two AND/OR formulas for \( f \), the minimum depth of a such a formula (or circuit) for \( f \) is exactly the minimum over all protocols for \( f \) of the maximum number of bits used by some path in the protocol. Thus to prove a lower bound on monotone circuit depth, it suffices to prove a lower bound on the communication complexity of this game or any easier game.

### 4 The Fork Game

We apply the above method to \( \text{ufork} \), and derive a completely symmetric game on strings (below) which we call the fork game. We will aim to preserve this symmetry throughout our analysis, controlling the amount of information released by both players, not just one. This symmetry is apparently an artifact of the symmetry of \( mL \).

The minterms of \( \text{ufork} \) correspond to simple paths ending in a fork (figure 1). Similarly the maxterms of \( \text{ufork} \) correspond to simple paths not ending in a fork; more precisely, the maxterm consists of all edges adjacent to the path but not themselves part of the path. Given a forked minterm path and a simple maxterm path, the goal of the game is to find a \textit{goal edge}: some edge of the minterm that shares a vertex with the maxterm path but is not itself part of the maxterm path.

We restrict the problem by leveling the graph into \( l + 2 \) levels. The parameter \( l \) is at most some polynomial in \( n \); it suffices to take \( l = n \) for this section, although we will be decreasing \( l \) in the lower bound arguments. We level the graph with the start node \( s \) alone on level 0, \( n \) nodes each in levels 1 through \( l \), and two distinguished end nodes \( t \) and \( t_b \) on level \( l + 1 \); also there are two additional nodes attached to \( t \), so that any path reaching \( t \) is sure to fork. We restrict the minterm paths to those which start at \( s \), proceed from level to level and end at \( t \) (where we have set up the fork). We restrict the maxterm paths to those which start at \( s \), proceed from level to level and end at \( t_b \).

Each such minterm or maxterm path is determined by the sequence of nodes it chooses to visit in levels 1 through \( l \); representing these choices by strings, we arrive at the following completely symmetric game on strings.
5 The Lower Bound Strategy

Consider some intermediate point in the protocol. Let $A$ denote the set of all strings $a$ consistent with what the first player has said so far, and let $B$ be the set of all strings $b$ consistent with what the second player has said so far. Then $S = A \cap B$ satisfies the following condition.

**Definition 7** A two-player protocol is an $(\alpha, l)$-protocol if there is some $S \subseteq \{1, \ldots, n\}^l$ such that $|S| \geq \alpha n^l$, and the protocol correctly solves the game on all input pairs $(a, b)$ where $a, b \in S$.

In particular, a correct protocol for the original game is a $(1, n)$-protocol.

We use the following rule to traverse a protocol: on each step, whichever player is to speak gives the answer which keeps $S$ as large as possible, so the size of $S$ decreases by at most one half. After $k$ bits are communicated by this strategy, $S$ still has size at least $2^{-k}$ fraction of its previous size.

Suppose that (after following this strategy for a while) $S$ still has size at least $\alpha n^l$. If $\alpha > 1/n$, we claim the players cannot yet know the answer $i$. As long as $S$ is nonempty, it is possible that $a = b$, thus the only possible correct answer is $i = l$. For the correct answer to be $i = l$, we would have to know that all $a \in A$ and $b \in B$ pass through the same point in the $l$th level; thus $|S| \leq |A \cap B| \leq n^{l-1}$, proving the claim.

Of course this traversal strategy alone will only show an $\Omega(\lg n)$ lower bound on the protocol depth. We now turn to an amplification step similar to that used in [KW] for the ustconn function: given an $(\alpha, l)$-protocol (with $l \geq 2$ and $\alpha$ not too small), we convert it to a $(\sqrt{\alpha}/2, l/2)$-protocol. Thus $\alpha$ may be amplified greatly while $l$ is cut in half. By amplifying $\alpha$ after every $\Theta(\lg n)$ steps of the traversal strategy, we may keep $\alpha > 1/n$ until $l$ reaches 1, showing the protocol has a path of depth at least $\Omega(\lg^2 n)$.

6 The Amplification Step

Let $U$ and $V$ be finite sets, and let $G$ be a bipartite graph of edges between $U$ and $V$. Say that $G$ has edge-density $\beta$ if $|G| = \beta |U| \cdot |V|$, where $|G|$ is the number of edges in $G$. Say that a vertex $u$ has degree-density $\beta$ if the degree of $u$ is $\beta|V|$ (and similarly for a vertex $v \in V$). Let $U_\beta$ denote the set of all $u \in U$ with degree-density at least $\beta$. We use the following simple lemma.
Lemma 8 If the edge-density of G is at least \( \alpha \), then either (a) there exists some \( u \in U \) with degree-density at least \( \sqrt{\alpha /2} \), or (b) \( |U_{a/2}| \geq \sqrt{\alpha /2} |V| \).

Proof: At most half of the edges of G can be adjacent to \( u \in U \setminus U_{a/2} \), or in other words at least half the edges of G come out of \( U_{a/2} \). Now if \( U_{a/2} \) is not large enough to satisfy (b), then by averaging some one vertex \( u \in U_{a/2} \) has enough adjacent edges to satisfy (a). \( \square \)

Note: The argument of [KW] (as presented in [BS]) depends on a similar lemma, which says that either \( U_{a/4} \) or \( V_{a/4} \) has size at least \( \sqrt{\alpha /2} \).

We are given an \((\alpha, l)\)-protocol, with set \( S \) as defined above. We apply the lemma where \( U \) consists of all \( n^1/2 \) possible strings on the first \( l/2 \) levels, and \( V \) consists of all \( n^1/2 \) possible strings on the last \( l/2 \) levels. We connect \( u \) and \( v \) in G if their concatenation \( uv \) is a string in \( S \); we say that \( v \) is an extension of \( u \). In both cases (a) and (b), we show how to construct a \((\sqrt{\alpha /2}, l/2)\)-protocol (see figure 3).

In case (a), we have one string \( u \) on the left that has many extensions \( v \) on the right such that \( uv \in S \). Thus we can recover a \((\sqrt{\alpha /2}, l/2)\)-protocol as follows: let \( S' \) be the set of extensions of \( u \). Given two strings \( a, b \in S' \), the two players can play the \( l/2 \)-game on these inputs by following the \( l \)-protocol for the strings \( ua, ub \). Since these paths are identical on the first \( l/2 \) levels, the answer \( i \) must correspond to a point where (the paths corresponding to) \( a \) and \( b \) diverge.

In case (b), we take a random partition of the \( nl/2 \) nodes in the right \( l/2 \) levels. More precisely, take \( n/2 \) nodes at random from each of the right \( l/2 \) levels, and call their union \( X \); call the set of remaining \( n/4 \) right nodes \( Y \).

Lemma 9 Given case (b) holds and we take a string \( u \in U_{a/2} \), then with probability at least \( 1 - 2e^{-\alpha n/4} \) there is an extension \( v_X(u) \) of \( u \) entirely in set \( X \) and another extension \( v_Y(u) \) of \( u \) entirely in set \( Y \).

Proof: We prove that such an extension \( v_X(u) \) exists with probability \( 1 - e^{-\alpha n/4} \), and then add failure probabilities.

We may construct \( X \) as follows. Take \( n/2 \) independent uniformly distributed paths \( v_1, \ldots, v_{n/2} \) on the right \( l/2 \) levels, and take the union of their vertices. Add more vertices if necessary in each column until each column has exactly \( n/2 \) vertices. Then this construction yields a random \( X \) according to our chosen distribution. Now for \( u \) to fail to have an extension in \( X \), it must be true in particular that each of \( v_1, \ldots, v_{a/2} \) failed to be an extension for \( u \). But since \( u \) has \( \alpha/2 \) degree-density in the bipartite graph, each \( v_i \) has probability at least \( \alpha/2 \) of being an extension of \( u \), so the probability that they all fail is at most \( (1 - \alpha/2)^{n/2} < e^{-\alpha n/4} \).

Now if we have \( \alpha \gg 1/n \), then \( 1 - o(1) \) of strings \( u \in U_{a/2} \) will have extensions in both \( X \) and \( Y \). In particular if we always keep \( \alpha \geq n^{-1/2} \), then (for \( n \) large enough) there exists some choice of \( X \) and \( Y \) such that \( \sqrt{\alpha /2} \) of all possible left-half strings have extensions in both \( X \) and \( Y \). This set of extendable strings is our new \( S' \).

This yields a \((\sqrt{\alpha /2}, l/2)\)-protocol as follows. Given strings \( a, b \in S' \), the players follow the \( l \)-protocol on the inputs \( av_X(a), bv_Y(b) \). Since the \( l \)-protocol is correct on these strings, and since they share no vertices in the right \( l/2 \) levels, the protocol must return an answer \( i \) in the first \( l/2 \) levels, hence the answer is in fact valid for \( a \) and \( b \).

7 Remarks

An obvious problem at this point is separating \( mL \) from \( mNL \cap co-mNL \). A candidate function for this separation is planar \( st \)-connectivity (say on a grid), where \( s \) and \( t \) are on the outer face. In principle it is possible to construct space-bounded communication games which capture circuit width, so perhaps this problem is tractable.

References


Figure 3: Two cases for constructing a protocol for strings $a$ and $b$ of length $l/2$.


