Monotone Separation of Logarithmic Space from Logarithmic Depth*

Michelangelo Grigni

Department of Computer Science, University of British Columbia, 6356 Agricultural Road, Vancouver, British Columbia, V6T 1Z2, Canada

and

Michael Sipser

Department of Mathematics, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts, 02139, USA

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

1. Introduction

In recent years there have been several strong results in monotone complexity theory. Razborov's theorem [10] showing that the clique function requires superpolynomial size monotone circuits solved a long-standing open question. More recently the result of Karchmer and Wigderson [7] 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.

For a complexity class C, we define its monotone counterpart, denoted by mC, which is in general different from the class $C \cap \mathbf{mono}$, functions in C that happen to be monotone. For example, P is the class of functions

*Supported by NSF grant 8912586-CCR, and Air Force contract AFOSR 89-0271. Research completed while the first author was at MIT.

computed by polynomial size AND/OR/NOT circuits, and mP is the class of functions computed by polynomial size AND/OR circuits with no negations. Other monotone circuit classes may be defined similarly, with the caveat that nondeterministic bits are allowed to be negated. These definitions are made precise in [4].

In this framework many theorems about monotone complexity may be conveniently restated. For example:

THEOREM 1.1 [10] $\mathrm{mP} \neq \mathrm{mNP}$.

THEOREM 1.2 [11] $\mathrm{mP} \neq \mathrm{P} \cap \mathbf{mono}$.

THEOREM 1.3 [1] $\mathrm{mAC}^0 \neq \mathrm{AC}^0 \cap \mathbf{mono}$.

THEOREM 1.4 [7] $\mathrm{mNC}^1 \neq \mathrm{mNL}$.

THEOREM 1.5 [13] $\mathrm{mTC}^0 \neq \mathrm{mNC}^1$.

THEOREM 1.6 [9] $\mathrm{NC}^1 \cap \mathbf{mono} \not\subset \mathrm{mNC}$.

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 $NL = co\text{-}NL$ [6, 12] cannot be replaced with a monotone simulation since $mNL \neq co\text{-}mNL$ (below). We see the further elucidation of the monotone classification scheme as an important step in the development of circuit complexity.

In this paper we show that monotone logarithmic depth circuits are strictly weaker than monotone polynomial size logarithmic width circuits, i.e. 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: $mNL \not\subset$ *co-*$mSAC^1$, i.e., *co-*$mSAC$ circuits for ustconn require depth $\Omega(\lg^2 n)$. Since ustconn $\in mNL \subseteq mSAC^1$, it follows that $mNL \neq$ *co-*mNL, and that mL is strictly contained in mNL. Hence the present separation is strictly stronger than Theorem 1.4 above.

2. Monotone Logspace

In the nonmonotone nonuniform case we know that L is the class of functions computable by polynomial size log-width circuits. Since our lower bound applies even in the non-uniform case, we give a circuit model for mL. We define mL to mean those functions computable by monotone polynomial size log-width circuits. Note that $mNC^1 \subseteq mL$, and we want to show this containment is strict.

We consider a certain monotone boolean function fork; the name follows from the appearance of its minterms. For a given size n, we consider a directed n-node graph with a distinguished vertex s (say s is vertex number 1). The input to fork is the adjacency matrix of the graph. Given such a matrix A, the function 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.

The construction below shows that fork is in mL. In fact fork is complete for L, even under very weak (but nonmonotone) reductions such as first-order translation reductions [5]. Here we show fork $\in mL$ by a simple circuit construction.

- The circuit will maintain a $\lceil \lg n \rceil$-bit binary address a of the current node. Since the circuit is monotone, it will also need to maintain $b = \neg a$ separately. Initially a and b are constants since the start node s is fixed; say $a = \vec{0}$ and $b = \vec{1}$. Given we have computed a and b and maintained $b = \neg a$, it is easy to check whether $a = v$ for any constant vertex address v (here "constant" means that the bits of v do not depend on the input).

Besides the addresses a and b, the circuit also maintains an "accept" bit f, initially 0.

- The circuit then repeats the following steps $n-1$ times.
  - Set up new address registers a' and b', initially *both* $\vec{0}$, so they do not yet correspond to any real address.
  - For each vertex u and for each vertex v, if $a = u$ and there is an edge from u to v (this is where we read an input), then set $a' \leftarrow a' \vee v$ and $b' \leftarrow b' \vee \neg v$; these $\vee$ operations are bitwise. The value $\neg v$ is available since v does not depend on the inputs; it is just a constant built into the circuit.
  - Check whether the addresses a' and b' have a 1 bit in common in some position; if so then set $f \leftarrow 1$. Assuming that $a = \neg b$, this happens iff there was outdegree greater than 1 at the node addressed by a and b.
  - Let $a \leftarrow a'$, $b \leftarrow b'$.
- Finally the circuit outputs the bit f.

Note if the path has not forked after $n-1$ steps, then we are in a cycle, hence the path will never fork. Also if we ever reach a node with 0 outedges, we will then set a and b to $\vec{0}$, and thereafter the $v = a$ test will always fail. Thus the above circuit correctly computes fork.

It will suffice for our lower bound to deal with the undirected version ufork: given the adjacency matrix of an undirected graph, 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 bounded fanin circuits for the ufork function require depth $\Omega(\lg^2 n)$.

3. Communication Complexity

We briefly review the communication complexity method developed in [7] and applied in [8, 9].

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

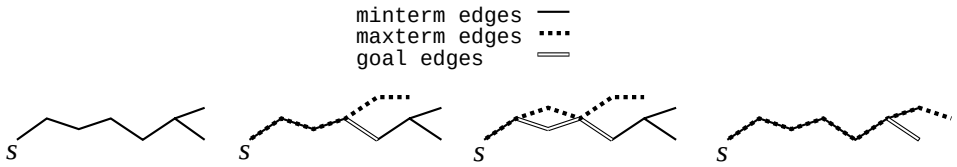

FIG. 1. A forked minterm path of ufork (left) and its interaction with three possible maxterm paths. In each instance, there is at least one *goal edge*: a minterm edge adjacent to but not part of the maxterm.

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 any 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 ufork, and derive a completely symmetric game on strings 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 preserving argument exploits the symmetry of $mL = co\text{-}mL$.

The minterms of ufork correspond to simple paths ending in a fork (see Figure 1). Similarly the maxterms of 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 *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 dividing the graph into $l + 2$ levels (see Figure 2). The parameter l is at most some polynomial in n; it suffices to take

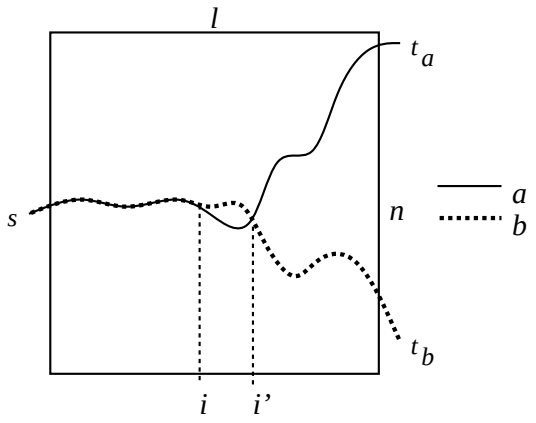

FIG. 2. The symmetric l-level communication game, where the goal is to find a level i where the paths share a vertex but differ on an adjacent level. There may be more than one possible answer, such as levels i and i' in this example. The fork at t_a has been suppressed in this figure to emphasize the symmetry.

$l = n$ 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_a and t_b on level $l + 1$; also there are two additional nodes attached to t_a, so that any path reaching t_a is sure to fork. We restrict the minterm paths to those which start at s, proceed from level to level and end at t_a (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.

> The two players are each given a string in $\{1, \ldots, n\}^l$; call these strings $a = a_1 \cdots a_l$ and $b = b_1 \cdots b_l$. The protocol must decide on an $i \in \{0, \ldots, l\}$ such that
>
> 1. if $i > 0$, then $a_i = b_i$;
> 2. either $i < l$ and $a_{i+1} \neq b_{i+1}$, or $i > 1$ and $a_{i-1} \neq b_{i-1}$.

If $a_1 \neq b_1$, then $i = 0$ is a valid answer, since the paths diverge on their first step from s. If $a = b$, the only correct answer is $i = l$, since the paths only diverge on their last step.

In terms of the underlying graph, the players are to find some level i where the paths converge or diverge; i.e. the paths share a node at level i but differ at an adjacent level. If a protocol for ufork could find a goal edge using d bits, then we could find a level i for this game using $d + O(1)$ bits, so a lower bound for this game solves our original problem.

There is a straightforward $O(\lg^2 n)$-bit protocol for this game using binary search; in fact either one of the players may do most of the talking, with the other player communicating only $O(\lg n)$ bits overall. This corresponds to the containment $mL \subseteq mSAC^1 \cap co\text{-}mSAC^1$.

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. The following definition is satisfied if we take $S = A \cap B$ and $\alpha = |S|/n^l$.

Definition 5.1 *A two-player protocol is an (α, 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.

Given a protocol, it is up to us to choose how to traverse it in order to demonstrate that a long path exists. We choose 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.

The following "stopping" lemma tells us when an (α, l)-protocol cannot yet be finished.

Lemma 5.2 *Given an (α, l)-protocol where $\alpha > 1/n$ and $l \geq 1$, then neither player yet knows the answer i.*

Proof. Suppose that (say) the first player knows that level i is a valid answer. Since S is non-empty, the first player must allow the possibility that $b = a$; in this case the only possible answer is level l, so in fact $i = l$.

This would imply that all paths in B must pass through the same node in level l as does path a. Then $|S| \leq |B| \leq n^{l-1}$, and so $\alpha \leq 1/n$. ∎

The traversal strategy and stopping lemma together show only an $\Omega(\lg n)$ lower bound on the communication complexity; such a bound also follows from information-theoretic considerations.

In the next section we prove the following amplification lemma similar to that used by Karchmer and Wigderson [7] for the ustconn function.

LEMMA 5.3 *For large enough n, given an (α, l)-protocol with $l \geq 2$ and $\alpha \geq n^{-1/2}$, there is a $(\sqrt{\alpha}/2, \lfloor l/2 \rfloor)$-protocol with the same or lesser depth.*

Thus α is amplified greatly while l is cut in half. Given this lemma, we now prove our main result. Starting with the original $(1, n)$-protocol, we repeatedly apply this amplification lemma after every $\lfloor (\lg n)/4 \rfloor$ steps of the traversal strategy; this keeps $\alpha > n^{-1/2}$ until l reaches 1. Hence the original protocol has a path of depth at least $\Omega(\lg^2 n)$, and monotone circuits for ufork have depth $\Omega(\lg^2 n)$.

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 β if $|G| = \beta\ |U| \cdot |V|$, where $|G|$ is the number of edges in G. Say that a vertex u has degree-density β if the degree of u is $\beta|V|$ (and similarly for a vertex $v \in V$). Let U_β denote the set of all $u \in U$ with degree-density at least β. We use the following simple lemma.

LEMMA 6.1 *If the edge-density of G is at least α, then either (a) there exists some $u \in U$ with degree-density at least $\sqrt{\alpha/2}$, or (b) $|U_{\alpha/2}| \geq \sqrt{\alpha/2}\ |U|$.*

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

We remark that the argument of [7] (as presented in [2]) depends on a similar lemma, which says that either $U_{\alpha/4}$ or $V_{\alpha/4}$ has size at least $\sqrt{\alpha/2}$.

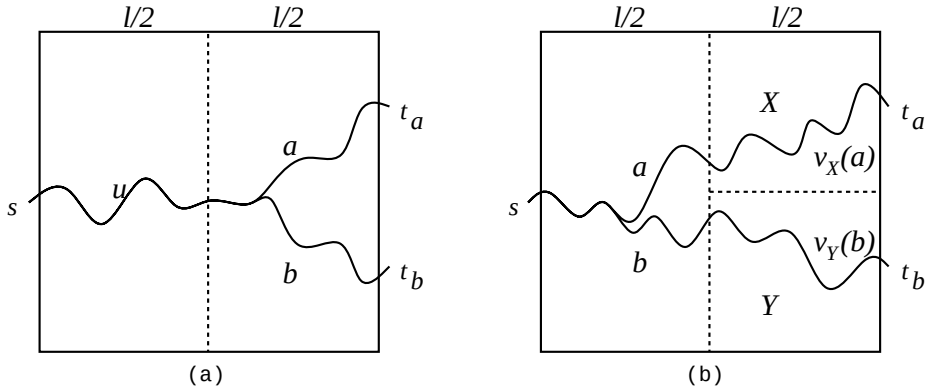

FIG. 3. Two cases for constructing a protocol for strings a and b of length $l/2$.

We now apply this to prove our amplification lemma. We are given an (α, l)-protocol, with set S as defined above. For simplicity, suppose l is even. We apply the lemma where U consists of all $n^{l/2}$ possible strings on the first $l/2$ levels, and V consists of all $n^{l/2}$ 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) given by Lemma 6.1, 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 $nl/4$ right nodes Y.

LEMMA 6.2 *Given case (b) holds and we take a string $u \in U_{\alpha/2}$, 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. In each column randomly take more vertices if necessary until we have 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 $v_1, \ldots, v_{n/2}$ 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 $u \in U_{\alpha/2}$ 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 $mNL \cap co\text{-}mNL$. A candidate function for this separation is planar st-connectivity (say on a grid), where s and t are on the outer face.

A further problem is finding a reasonable uniform non-circuit model for mL. This is unlike the case for mNL, where nondeterminacy may be exploited to give a monotone Turing machine model. A perhaps related result is that while fork is complete for L under log-time reductions, it is not complete for mL even under mTC^0 (monotone bounded depth polynomial size threshold circuit) reductions [3].

References

1. M. Ajtai and Y. Gurevich, Monotone versus positive, *Journal of the ACM* **34** (1987), 1004–1015.

2. R. B. Boppana and M. Sipser, The complexity of finite functions, *in* "Handbook of Theoretical Computer Science," Vol. A, Elsevier and MIT Press, 1990, 757–804.

3. M. Grigni, "Structure in Monotone Complexity," Ph.D. thesis, Massachusetts Institute of Technology, Technical Report MIT/LCS/TR-520, 1991.

4. M. Grigni and M. Sipser, Monotone Complexity, *to appear in* "Boolean function complexity: selected papers from the London Math. Soc. Durham Symp., 1990," LMS Lecture Notes, Cambridge U. Press, 1992.

5. N. Immerman, Languages that capture complexity classes, *SIAM J. on Computing*, **16** (1987), 760–778.

6. N. Immerman, Nondeterministic space is closed under complementation, *SIAM J. on Computing* **17** (1988), 935–938.

7. M. Karchmer and A. Wigderson, Monotone circuits for connectivity require super-logarithmic depth, *SIAM J. on Discrete Math.* **3** (1990), 255–265. *Also in* "Proc. of the 20th Annual ACM Symp. on Theory of Computing, 1988," 539–550.

8. R. Raz and A. Wigderson, Probabilistic communication complexity of boolean relations, *in* "Proc. of 30th Annual IEEE Symp. on Foundations of Computer Science, 1989," 562–573.

9. R. Raz and A. Wigderson, Monotone circuits for matching require linear depth, *in* "Proc. of the 22nd Annual ACM Symp. on Theory of Computing, 1990," 287–292.

10. A. A. Razborov, Lower bounds on the monotone complexity of some Boolean functions, *Doklady Akademii Nauk SSSR* **281** (1985), 798–801. English translation in *Soviet Mathematics Doklady* **31** (1985), 354–357.

11. A. A. Razborov, A lower bound on the monotone network complexity of the logical permanent, *Matematicheskie Zametki* **37** (1985), 887–900. English translation in *Mathematical Notes of the Academy of Sciences of the USSR* **37** (1985), 485–493.

12. R. Szelepcsényi, The method of forcing for nondeterministic automata, *Bull. Europ. Ass. Theoretical Computer Sci.* **33** (1987), 96–100.

13. A. Yao, Circuits and local computation, *in* "Proc. of the 21st Annual ACM Symp. on Theory of Computing, 1989," 186–196.