A Lower Bound for Noncommutative Monotone Arithmetic Circuits *
(Extended Abstract)

Rimli Sengupta
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
e-mail : rimli@cc.gatech.edu

GIT–ICS–94/05

November, 1993

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332–0280

Abstract

We consider arithmetic circuits over the semiring $(\Sigma^*, min, concat)$ and show that such circuits require super-polynomial size to compute the lexicographically minimum perfect matching of a bipartite graph. By defining monotone analogues of optimization classes such as OptP, OptL and OptSAC^1 using the monotone analogues of their arithmetic circuit characterizations [13, 1], our lower bound implies that this problem is not in monotone OptSAC^1. But we show that this problem is in monotone OptP, leading to a separation between these two classes.

*This work was supported by NSF grant CCR-9200878.

1 Introduction

We consider arithmetic circuits over the semiring $(\Sigma^*, min, concat)$ and show that such circuits require super-polynomial size to compute the lexicographically minimum perfect matching of a bipartite graph. By defining monotone analogues of optimization classes such as OptP, OptL and OptSAC^1 using the monotone analogues of their arithmetic circuit characterizations [13, 1], our lower bound implies that this problem is not in monotone OptSAC^1. But we show that this problem is in monotone OptP, leading to a separation between these two classes.

Krentel [5] introduced OptP to be the class of functions f, such that $f(x)$ is the lexicographically optimum amongst all the strings produced along computation paths of a non-deterministic polynomial time transducer, on input x. Subsequently, Alvarez and Jenner [2] studied OptL, the class defined analogously for a logspace bounded non-deterministic transducer. The corresponding class for logspace, polynomial time non-deterministic auxilliary pushdown automata was called OptSAC^1 and studied by Vinay in [13].

In [1], Allender and Jiao observe that using the techniques developed in [2, 12], OptL (OptP) can be characterized as all those functions computable within polynomial size (depth, resp.), by uniform families of arithmetic circuits, in which each *concat* gate has at most one non-leaf input. They also give a characterization of OptSAC^1 as the class of functions computable by uniform families of arithmetic circuits, within polynomial size and polynomial degree. The circuits in all these characterizations are equipped with certain special leaf nodes, that are capable of comparing a given input to any symbol in the alphabet Σ. These special leaf nodes are at least syntactically analogous to the negated inputs of a Boolean circuit that makes it non-monotone. In this paper, we define "monotone" arithmetic circuits to be those without the special leaf nodes and consider the "monotone" versions of Opt classes defined using such circuits. We define mOptSAC^1 to be the monotone analogue of OptSAC^1. mOptL and mOptP are defined similarly. Our main result is that a natural function, namely lexicographically minimum bipartite perfect matching, is not in mOptSAC^1, but is in mOptP. We show this by proving a super-polynomial size lower bound for monotone arithmetic circuits that compute this function. Our result uses Razborov's [9] lower bound for monotone Boolean circuits that decide whether a bipartite graph has a perfect matching.

The following are some of the interesting aspects of the results in this paper :

- To our knowledge, this is the first non-trivial *circuit* size lower bound for computation in a non-commutative semiring. In [8], Nisan proved an exponential *formula* size lower bound, or equivalently, a linear depth lower bound for computing the determinant in a non-commutative ring.

- There is considerable interest in studying monotone analogues of complexity classes due to the success in obtaining separations between these classes, such as, the separation of monotone P from monotone NP [9] and monotone L from monotone NC^1 [4]. Our separation of mOptSAC^1 from mOptP has a similar flavor. We note that many of these monotone classes may not have natural Turing machine analogues.

- We provide a natural circuit setting for studying lexicographically optimum versions of monotone Boolean functions. Given a monotone Boolean function $f : \{0,1\}^* \rightarrow \{0,1\}$, consider the function that on input $x \in \{0,1\}^*$, outputs a string that encodes the lexicographically minimum minterm of f that evaluates to 1 on x. Such a function is computable in a natural fashion by arithmetic circuits over $(\Sigma^*, min, concat)$. Similarly, lexicograph-

ically maximum versions of monotone Boolean functions are computable naturally with circuits over $(\Sigma^*, max, concat)$.

- Lexicographically optimum versions of natural problems have attracted significant attention in the past [5, 6, 3, 7, 11]. Our lower bound helps determine the circuit complexity of one such problem.

2 Preliminaries

A *semiring* is an algebra with two operations $+$ and $\times$, satisfying the usual ring axioms, but not necessarily having additive inverses. We consider the semiring $(\Sigma^* \cup \{\perp\}, +, \times)$, where Σ is any alphabet, $\times$ denotes concatenation ($\perp \times x = x \times \perp = \perp$, for all x) and $+$ denotes lexicographic minimum ($x + \perp = \perp + x = x$, for all x). We denote this semiring as MIN and define the semiring MAX analogously, with $+$ denoting lexicographic maximum.

An *arithmetic circuit* over MAX or MIN is a rooted directed acyclic graph with interior nodes labeled with $+$ or $\times$. The leaf nodes of the circuit are labeled either with an input variable x_i or with some element of $\Sigma \cup \{\perp\}$. The root node is the output of the circuit. A *circuit family* is a set of circuits $\{C_n |\ n \geq 1\}$, where C_n has n input variables. An arithmetic circuit is said to be *skew* if each $\times$ gate has at most one non-leaf input.

An arithmetic circuit is said to be *non-monotone*, if in addition to the leaf nodes described above, it has special leaf nodes that return λ if $x_i = a$, for $a \in \Sigma \cup \{\perp\}$, and return $\perp$ otherwise. If an arithmetic circuit is not non-monotone, we shall say it is *monotone*.

The *size* of an arithmetic circuit is the number of gates (non-leaf nodes) in it, and its *depth* is the length of the longest path from the output to any input. The *degree* of an arithmetic circuit is defined inductively : a leaf node has degree 1; the degree of a $+$ node is the maximum of the degree of its inputs; the degree of a $\times$ node is the sum of the degree of its inputs; the degree of the circuit is the degree of the output node.

The circuit families we consider are uniform with respect to U_D-uniformity, defined by Ruzzo [10]. That is, the direct connection language of an arithmetic circuit is recognizable by a deterministic Turing machine within time logarithmic in the size of the circuit.

As was observed in [1], the techniques in [12] can be used to prove the following characterizations:

Proposition 2.1 *OptL (OptP) is the class of function families $\{f_n |\ n \geq 1\}$, computable by uniform families of skew non-monotone arithmetic circuits $\{C_n |\ n \geq 1\}$ over MAX or MIN, such that for all $n \geq 1$, C_n computes f_n within size (depth, resp.) $n^{O(1)}$.*

A circuit characterization of OptSAC^1 was given in [1], also using the techniques of [12].

Proposition 2.2 *OptSAC^1 is the class of function families $\{f_n | n \geq 1\}$, computable by uniform families of non-monotone arithmetic circuits $\{C_n |\ n \geq 1\}$ over MAX or MIN, such that for all $n \geq 1$, C_n computes f_n within size $n^{O(1)}$ and degree $n^{O(1)}$.*

By the above definitions, $\mathrm{OptL} \subseteq \mathrm{OptSAC}^1 \subseteq \mathrm{OptP}$. In this paper, we focus on the subclasses of the above that are computable *without* access to the special leaf nodes.

Definition 2.1 *mOptL (mOptP) is the class of function families* $\{f_n |\ n \geq 1\}$*, computable by uniform families of skew monotone arithmetic circuits* $\{C_n |\ n \geq 1\}$ *over MAX or MIN, such that for all* $n \geq 1$*,* C_n *computes* f_n *within size (depth, resp.)* $n^{O(1)}$*. mOptSAC*^{1} *is the class of function families* $\{f_n |\ n \geq 1\}$*, computable by uniform families of monotone arithmetic circuits* $\{C_n |\ n \geq 1\}$ *over MAX or MIN, such that for all* $n \geq 1$*,* C_n *computes* f_n *within size* $n^{O(1)}$ *and degree* $n^{O(1)}$*.*

As before, mOptL $\subseteq$ mOptSAC^1 $\subseteq$ mOptP.

3 Main Result

In this section we exhibit a natural problem in mOptP that is not in mOptSAC^1. Let LMBPM $= \{LMBPM_m |\ m \geq 1\}$ be a function family with LMBPM$_m$: $\{\Sigma \cup \perp\}^m \rightarrow \Sigma^n \cup \{\perp\}$ defined as follows, where $m = n^2$:

Input: $n \times n$ matrix X $= [x_{ij}]$ with entries from the alphabet $\Sigma \cup \{\perp\}$, encoding a bipartite graph G with total order on its edges.
Output: the lexicographically minimum perfect matching, if G has one, $\perp$ otherwise.

LMBPM is a natural variant of the familiar monotone Boolean function family BPM $= \{BPM_m |\ m \geq 1\}$, with BPM$_m$: $\{0,1\}^m \rightarrow \{0,1\}$ defined as :

Input: $n \times n$ matrix Y $= [y_{ij}]$ with entries from $\{0,1\}$, encoding a bipartite graph G.
Output: 1, if G has a perfect matching, 0 otherwise.

Let $\mathcal{C} = \{C_m |\ m \geq 1\}$ be a uniform family of monotone arithmetic circuits that computes LMBPM. Let $\mathcal{B} = \{B_m |\ m \geq 1\}$ be a uniform family of monotone Boolean circuits in which each B_m is obtained from C_m by replacing each + gate with an $\vee$ gate and each $\times$ gate with a $\wedge$ gate. Moreover, if C_m has the matrix X $= [x_{ij}]$ as input, then Y $= [y_{ij}]$, the input to B_m, is derived as follows : if $x_{ij} \in \Sigma$, then $y_{ij} = 1$, otherwise $y_{ij} = 0$.

The following theorem relates the computations of C_m and B_m.

Theorem 3.1 *If C_m computes $LMBPM_m$ on input X, then B_m computes BPM_m on input Y.*

Proof : The bipartite graph G encoded by X is simply the one encoded by Y, with a total order on its edges. Let $P(C_m)$ and $P(B_m)$ be the formal polynomials associated with C_m and B_m respectively. There is clearly a bijection between the monomials of $P(C_m)$ and those of $P(B_m)$. Now, if G has a perfect matching, then there is at least one monomial in $P(C_m)$ all of whose variables receive values from Σ. By construction, all the variables in the corresponding monomial in $P(B_m)$ receive the value 1 on input Y. Therefore, B_m evaluates to 1. Conversely, if G doesn't have a perfect matching, then for every monomial in $P(C_m)$, there is at least one variable that receives the value $\perp$. Therefore, for every monomial in $P(B_m)$, there is at least one variable that gets a 0 value on input Y, causing B_m to evaluate to 0. □

This leads directly to our lower bound.

Corollary 3.1 *If C_m is a monotone arithmetic circuit computing $LMBPM_m$, then C_m must have size $\Omega(m^{logm})$.*

Proof: Suppose there is a monotone arithmetic circuit C_m that computes LMBPM_m within size $s = o(m^{logm})$. Then, by the theorem above, there exists a monotone Boolean circuit B_m that computes BPM_m within size s. But by [9], B_m must have size $\Omega(m^{logm})$, giving the desired contradiction. □

The above corollary implies that LMBPM does not belong to the class defined by polynomial size monotone arithmetic circuits. Therefore, LMBPM $\notin$ $\mathrm{mOptSAC}^1$. However, we show that it does belong to mOptP.

Theorem 3.2 *$LMBPM \in mOptP$.*

Proof: The following polynomial represents the function LMBPM_m, when $+$ denotes lexicographic minimum and $[x_{ij}]$ is the matrix X defined above.

$$\sum_{\pi \in S_n} \sum_{\sigma \in P_\pi} \sigma,$$

where S_n is the set of all permutations of $\{1, 2, \ldots, n\}$ and P_π is the set of all strings of length n over the set $\{x_{i,\pi(i)} | 1 \leq i \leq n\}$.

This polynomial can be easily implemented by a skew monotone arithmetic circuit over MIN, within linear depth. □

But by definition, $\mathrm{mOptSAC}^1 \subseteq$ mOptP. Therefore, we have,

Corollary 3.2 *$mOptSAC^1 \subsetneq mOptP$.*

4 Concluding Remarks

It is natural to enquire about the relative power of non-monotone arithmetic circuits over those that are monotone. It seems clear that there are functions that a monotone arithmetic circuit cannot compute, for instance, the function that computes the lexicographically minimum satisfying assignment of an input CNF formula [5]. But even in the context of computing only the lexicographically optimum versions of monotone Boolean functions, it is meaningful to ask

whether non-monotone circuits can perform the computation within less resources. This would require an understanding of how the special leaf nodes in a non-monotone circuit help the computation.

Improving our lower bound to hold for non-monotone arithmetic circuits would lead to a separation between OptSAC^1 and OptP. We would also like to improve this bound from super-polynomial to exponential.

Finally, we note that using an approach very similar to the one in this paper, we can obtain a separation between the monotone analogues of the counting classes $\sharp$LOGCFL and $\sharp$P. Such classes are defined using arithmetic circuits over $(+, \times)$, with the inputs being 0 or 1 [13, 12].

Acknowledgements The author would like to thank Eric Allender for clarifying several points and H. Venkateswaran for numerous discussions that led to this paper.

References

[1] E. Allender and J. Jiao, *Depth reduction for non-commutative arithmetic circuits*, Proc. 25th annual ACM Symposium on Theory of Computing, 1993, 515-522.

[2] A. Alvarez and B. Jenner, *A very hard logspace counting class*, Proc. 5th annual IEEE Conference on Structure in Complexity Theory, 1990, 154-168.

[3] R. Greenlaw, *Ordered vertex removal and subgraph problems*, JCSS, 39-3 (1989), 323-341.

[4] M. Grigni and M. Sipser, *Monotone separation of Logspace from NC^1*, Proc. 6th annual IEEE Conference on Structure in Complexity Theory, 1991, 294-298.

[5] M. Krentel, *The complexity of optimization problems*, JCSS, 36 (1988), 490-509.

[6] S. Miyano, *Δ_2^p-complete lexicographically first maximal subgraph problems*, Mathematical Foundations of Computer Science, LNCS (1988), 454-462.

[7] S. Miyano, *The lexicographically first maximal subgraph problems: P-completeness and NC algorithms*, Math. Systems Theory, 22 (1989), 47-73.

[8] N. Nisan, *Lower bounds for non-commutative computation*, Proc. 23rd annual ACM Symposium on Theory of Computing, 1991, 410-418.

[9] A. A. Razborov, *A lower bound on the monotone network complexity of the logical permanent*, Mathematischi Zametki, 37 (1985), pp. 887-900.

[10] W. L. Ruzzo, *On uniform circuit complexity*, JCSS, 22 (1981), 365-383.

[11] S. Toda, *The complexity of finding medians*, Proc. 30th annual IEEE Symposium on Foundation of Computer Science, 1990, 778-787.

[12] H. Venkateswaran, *Circuit definitions of non-deterministic complexity classes*, SIAM J. Comput. 21 (1992), 655-670.

[13] V. Vinay, *Counting auxilliary pushdown automata and semi-unbounded arithmetic circuits*, Proc. 6th annual IEEE Conference on Structure in Complexity Theory, 1991, 270-284.