Depth Reduction for Noncommutative Arithmetic Circuits (Extended Abstract)

Eric Allender*
Department of Computer Science
Princeton University
35 Olden Street
Princeton, NJ, 08544-2087
allender@cs.princeton.edu

Jia Jiao†
Department of Computer Science
Rutgers University
Hill Center, Busch Campus
New Brunswick, NJ, USA 08903
jjiao@paul.rutgers.edu

Abstract

We show that for every family of arithmetic circuits of polynomial size and degree over the algebra (Σ^*, max, concat), there is an equivalent family of arithmetic circuits of depth $\log^2 n$. (The depth can be reduced to $\log n$ if unbounded fan-in is allowed.) This is the first depth-reduction result for arithmetic circuits over a noncommutative semiring, and it complements the lower bounds of [Ni91, Ko90] showing that depth reduction cannot be done in the general noncommutative setting. The (max,concat) semiring is of interest, because it characterizes certain classes of optimization problems [AJ92, Vi91]. In particular, our results show that OptSAC^1 is contained in AC^1.

We also prove other results relating Boolean and arithmetic circuit complexity. We show that AC^1 has no more power than arithmetic circuits of polynomial size and degree $n^{O(\log \log n)}$ (improving the trivial bound of $n^{O(\log n)}$). Connections are drawn between TC^1 and arithmetic circuits of polynomial size and degree.

1 Introduction

One of the most striking early results of arithmetic circuit complexity is the theorem of [VSBR83], showing that any arithmetic circuit of polynomial size and polynomial algebraic degree, with $+$ and $\times$ gates defined over a commutative semiring, is equivalent to an arithmetic circuit of polynomial size having depth $\log^2 n$. (In fact, if the $+$ gates are allowed to have unbounded fan-in, the depth is logarithmic, as was observed in [Vi91].) The results of [VSBR83] were extended in [MRK88] by providing fast parallel algorithms for evaluating arithmetic circuits, and [MRK88] raised the question of whether analogous results could be proved in the presence of noncommutative multiplication.

Motivated by this question, [Ko90] and [Ni91] showed that commutativity is crucial for the results of [VSBR83, MRK88]. Namely, it was shown in [Ko90, Theorem 1] that for the particular semiring of 2^{Σ^*} with $\times$ denoting concatenation and $+$ denoting union, there is a circuit with linear size and degree that is not equivalent to any circuit with sublinear depth. An essentially equivalent example is presented in [Ni91, Theorem 4]. Although fast parallel algorithms were shown for certain limited sorts of noncommutative algebras (e.g., finite semirings) in [MT87], no examples were known of noncommutative semirings where depth reduction can be accomplished within the arithmetic circuit model.

We show that depth reduction can be carried out in the particular case of the algebra on Σ^* where $\times$ is concatenation and $+$ is lexicographic maximum. We do not have an interesting characterization of the class of algebras to which our techniques apply. However, this particular algebra is of interest for two reasons.

- Concatenation is in some sense the canonical example of a noncommutative multiplication operation.
- Natural classes of optimization problems (in particular the classes OptL [AJ93, AJ92] and OptSAC^1 [Vi91]) can be characterized in terms of arithmetic circuits over (max,concat).

In Section 4 and in the Appendix we include some related results concerning arithmetic circuits over commutative semirings, and connections to Boolean complexity classes.

*Supported in part by National Science Foundation grant CCR-9204874; currently on leave from Rutgers University.

†Supported in part by National Science Foundation grant CCR-9204874.

2 Arithmetic Circuits and Optimization Classes

A *semiring* is an algebra with two operations $+, \times$ satisfying the usual ring axioms, but not necessarily having additive inverses. (For more formal definitions see [JS82].) For any alphabet Σ, one obtains the semiring $(\Sigma^* \cup \{\perp\}, +, \times)$ where $\times$ denotes concatenation (and $\perp \times x = x \times \perp = \perp$ for all x) and $+$ denotes lexicographic maximum (where $x + \perp = \perp + x = x$ for all x). We will usually denote this semiring as (Σ^*,max,concat).

An *arithmetic circuit* over this semiring consists of gates labeled with the operations $+$ and $\times$. For ease of exposition, we will allow $+$ gates to have unbounded fan-in. Each $\times$ gate has a left input and a right input. The leaf nodes of the circuit are labeled either with an input variable x_i, or with some element of $\Sigma^* \cup \{\perp\}$. A circuit has one output node. A *circuit family* is a set of circuits $\{C_n : n = 1, 2, \ldots\}$, where C_n has n input variables. In order to capture the classes OptL and OptSAC^1 in terms of arithmetic circuits, it is necessary to (1) restrict the input variables to take on values in Σ (so C_n can be viewed as taking inputs of length n), and (2) allow leaf nodes that return λ if $x_i = a$ for a given $a \in \Sigma$, and return $\perp$ otherwise. Since our motivation comes in part from these optimization classes, this is the model of arithmetic circuit we will use throughout.

We will also have occasion to discuss arithmetic circuits over the naturals and over other commutative semirings. Our definitions in this setting are completely standard and conform to those of [VSBR83], for instance.

The *size* of an arithmetic circuit is the number of gates in it, and the *depth* is the length of the longest path from an input to the output. We will also need the notion of the (algebraic) *degree* of a node (which should not be confused with the fan-in of a node). The degree is defined inductively: a leaf node has degree 1, a $+$ node has degree equal to the maximum of the degrees of its inputs, and a $\times$ node has degree equal to the sum of the degrees of its inputs. The degree of a circuit is the degree of its output gate. A circuit family is *uniform* if the function $1^n \mapsto C_n$ is logspace-computable. (Note that uniform circuit families have polynomial size.)

A circuit is *skew* if each $\times$ gate has at most one non-leaf input. (Skew circuits have been used to characterize the complexity of the determinant [To92] as well as NLOG [Ve92].)

The class OptP was defined by Krentel [Kr88] as the class of functions that can be defined as $f(x) = \max\{y :$ there is some path of M that outputs y on input $x\}$, where M is a nondeterministic poly-time Turing machine. The analogous class OptL defined in terms of logspace-bounded nondeterministic machines was defined and studied in [AJ93, AJ92]. It is shown in [AJ92] that OptL is contained in AC^1 (a later proof may be found in [ABP92]); it is also shown in [AJ92] that iterated matrix multiplication over (Σ^*,max,concat) is complete for OptL. As was pointed out by Jenner [Je93], it is not hard to use the techniques of [To92, Ve92] to show:

Proposition 2.1 *OptL is the class of functions computed by uniform families of skew arithmetic circuits over (Σ^*, max,concat).*

Vinay considered the analogous class defined in terms of nondeterministic logspace-bounded AuxPDAs running in polynomial time [Vi91]. Since AuxPDAs characterize the class LOGCFL (consisting of problems logspace reducible to context-free languages), and LOGCFL in turn is precisely the class SAC^1 (consisting of problems computed by uniform log-depth circuits of bounded fan-in AND gates and unbounded fan-in OR gates), he called this class OptSAC^1. In [Vi91] it was claimed that OptSAC^1 is the class of functions computed by logarithmic-depth arithmetic circuits over (Σ^*,max,concat), but this claim was later retracted [Vi91a]. Instead, the following is easy to show (using the techniques of e.g., [Vi91, Ve91]):

Proposition 2.2 *OptSAC^1 is the class of functions computed by uniform families of arithmetic circuits of polynomial degree over (Σ^*, max,concat).*

Proof: ($\subseteq$) Let AuxPDA M be given; it can be assumed that the worktape of M keeps track of

- the number of output symbols that have been produced thus far in the computation, and
- the number of steps executed so far.

Also assume that each time an output symbol is produced, it is preceded by a push and followed by a pop, and that the stack changes height by one on all other moves.

The circuit we build will have gates with labels of the form (C, D, i, j, a, b), which should evaluate to the maximum of all words w that can be produced as bits i through j of a string output in a segment of computation beginning at time a and ending at time b, beginning in surface configuration C and ending in surface configuration D, where (C, D) is a realizable pair. (For definitions of "surface configuration" and "realizable pair," see [Co71].) The leaves will be of the form $(C, D, i, i + c, a, a + 1)$ (where $c \in \{-1, 0\}$) which will evaluate to $\sigma \in \Sigma \cup \{\lambda\}$ if M can move in one step from C to D outputting σ (and i, c and a agree with C and

D) and will evaluate to $\perp$ otherwise; note that this leaf will depend on the input.

Non-leaf nodes of the form (C, D, i, j, a, b) are the maximum over all E, F, k, and c of $\text{concat}((C, E, i, k, a, c),\ (E, D, k+1, j, c+1, b))$ and $(E, F, i, j, a+1, b-1)$ (where in this last expression only those E and F are considered where $C \vdash E$ via a push and $F \vdash D$ via a pop of the same symbol). Standard analysis ([Co71]) shows that gates defined in this way have the properties outlined in the preceding paragraph. The output of the circuit is the maximum over all m of $(C_{init}, D_{accept}, 1, m, 1, n^k)$. The degree of any node (C, D, i, j, a, b) can be seen to be $b - a$, and thus is polynomial.

($\subseteq$) This direction is also completely standard. The AuxPDA will start exploring the circuit C_n at the root. To explore a $\times$ gate, put the right child on the stack and explore the left child. To explore a $+$ gate, nondeterministically choose a child and explore it. To explore a leaf, output the value of the leaf (this might depend on the input); then pop the top node off the stack and explore it.

It is easy to see by induction that the time required to explore a gate g is $O(\text{depth}(g)\times \text{degree}(g)\times t(n))$ where t is the time required to check connectivity between gates. Thus the entire running time is polynomial. ∎

This paper investigates the degree to which this class can be characterized in terms of circuits of small depth. No parallel algorithm for OptSAC^1 is presented in [Vi91a], and in fact this is explicitly listed as an open problem there; instead, attention is drawn to the negative results of [Ni91, Ko90] showing that depth reduction is not possible in general for noncommutative semirings.

3 Depth Reduction for (max, concat) Circuits

In this section we show that (max,concat) circuits of polynomial size and degree can be simulated by (max,concat) circuits of polynomial size and logarithmic depth, when unbounded fan-in gates are allowed.

Our first proof of this fact was rather complicated, and was similar in spirit to the proof given in the appendix showing that in the case of commutative semirings, the class of (uniform) arithmetic circuits of polynomial size and degree is *equivalent* to the class of (uniform) arithmetic circuits of polynomial size and logarithmic depth, where $+$ gates have unbounded fan-in and $\times$ gates have fan-in two. Although we feel that a proof in this vein is instructive, the proof given below is extremely simple, and is very similar to the argument in [ABP92].

Note that, in order to achieve logarithmic depth, unbounded fan-in concat gates must be allowed; it remains an open question if the equality "poly-size and degree arithmetic circuits = poly-size, log-depth semi-unbounded arithmetic circuits" holds also for (max,concat), as is the case for commutative semirings.

Theorem 3.1 *If f is computed by a family of arithmetic circuits over $(\Sigma^*$, max, concat$)$ of polynomial size and degree, then f is computed by a family of arithmetic circuits over $(\Sigma^*$, max, concat$)$ of polynomial size with depth $O(\log^2 n)$.*

Proof: The following definition is similar to the notion of "accepting subtree" studied in [VT89]: Let g be a $+$ gate, and let h be an input to g, and consider the behavior of the circuit when given some input x. We say that h *contributes* to the value of g if the value of h is equal to the value of g (that is, the value of h is the largest value that is input to g). More generally, we say that a gate g contributes to the value of g' (where g' is not necessarily adjacent to g) if there is a path from g to g' such that every edge $h \to h'$ on this path (where h' is a $+$ gate) has the property that h contributes to the value of h'. We say that g contributes to the value of the circuit if it contributes to the value of the output gate. A *contributing subcircuit at h* is a subcircuit where each $+$ gate has one child and each $\times$ gate has two children, and all nodes in the subcircuit contribute to the value of h.

Lemma 3.2 *For any circuit family of polynomial size and degree, there is an equivalent circuit family of polynomial size and degree such that each node (other than the output node) is labeled with a pair i, j, and if node h is labeled with i, j, then it contributes to the value of the circuit only if the value of h is equal to symbols i through j of the output. (For convenience later on, we will number symbols from the right, starting with position 0 at the rightmost end.)*

This is immediate from the proof of Proposition 2.2.

The outline of our proof is now as follows: Given an input x and a circuit in the form guaranteed by Lemma 3.2, we first build an equivalent circuit over the (commutative) semiring ($\mathbf{Z}$,max,+). Then we evaluate this circuit using the [MRK88] algorithm, which in this setting can be implemented in AC^1. Then we turn this AC^1 algorithm into a family of arithmetic circuits.

Assume that the alphabet $\Sigma = \{0, 1\}$; the argument for other alphabets is similar.

Let input x and circuit C_1 be given. Replace each leaf of C_1 that evaluates to 1 (0) with a pair of leaves evaluating to (11) (10); this has the effect of forcing any output of non-zero length to have some 1's in it. Call this new circuit C. Let n^k be an upper bound on the number of bits in the output of $C(x)$ (this follows from the degree bound on C).

Now build a (max,+) circuit (operating over the integers) as follows.[1] Recall that each leaf node of C is labeled with a pair as in Lemma 3.2. For each leaf labeled with pair (i,i) that evaluates to 1, change that leaf to the number 2^i. Any leaf that evaluates to $\perp$ will be replaced by a leaf evaluating to -2^{1+n^k}. All other leaves receive the value 0. Call the new circuit C'.

It is now easy to observe that the output of C' is the number whose binary representation is the value of the output gate of C.

The circuit C' can be evaluated using the algorithm of [MRK88], which consists of $O(\log n)$ applications of a routine called *Phase*, where a single application of *Phase* consists of matrix multiplication over $(\mathbf{Z},\max,+)$, and hence can be done in AC^0. Thus $O(\log n)$ applications of *Phase* can be done in AC^1.

Let f be the function computed by the original (max,concat) circuit family. We have seen that the language $L = \{x, i, b : \text{ the } i\text{th bit of } f(x) \text{ is } b\}$ is in AC^1. By a trivial transformation, the language $L_1 = \{x, i, b : \text{ the } i\text{th bit of } C_1(x) \text{ is } b\}$ is also in AC^1. Now we can build log-depth arithmetic circuits over (max,concat) for C_1 in an essentially trivial way. Namely, note that $(\{\perp, \lambda\}, \max, \text{concat})$ is isomorphic to $(\{0,1\}, \vee, \wedge)$. Thus we can build log-depth arithmetic circuits (using unbounded fan-in max and concat gates) of the form $[i, b]$ that evaluate to λ if x, i, b is in L_1, and evaluate to $\perp$ otherwise. The final arithmetic circuit is the maximum (over all output lengths m) of the result of concatenating (for $m \geq i \geq 1$) the maximum over all bits b of $\text{concat}(b, [i, b])$. ∎

4 Relationships among Arithmetic and Boolean Complexity Classes

Computing the determinant of integer matrices is known to be hard for NLOG, and it can be done in TC^1 (TC^1 denotes the class of things computable by threshold circuits (equivalently, MAJORITY circuits) of polynomial size and depth $O(\log n)$). However, no relationship is known between SAC^1 or AC^1 and the determinant. In this section we review some known results about arithmetic circuits that bear on these questions, and present some new inclusions and characterizations.

Definition 4.1 *#L is the class of functions of the form $\#acc_M(x)$, where M is a NLOG machine. ($\#acc_M(x)$ counts the number of accepting computations of M on input x.)*

$\#SAC^1$ is the class of functions of the form $\#acc_M(x)$, where M is a polynomial-time bounded nondeterministic AuxPDA.

(Perhaps a more natural definition of $\#\mathrm{SAC}^1$ is as the class of functions computed by uniform polydegree arithmetic circuits over the natural numbers; see [Vi91]. Equivalently, we may restrict these arithmetic circuits to be of depth $O(\log n)$, with unbounded fan-in $+$ gates and fan-in two $\times$ gates.)

It is known that the complexity of the determinant is roughly determined by #L. More specifically, f is logspace many-one reducible to the determinant[2] iff it is the difference of two #L functions (see [Vi91a, Da91, To91a]; an essentially equivalent result is also proved in [Va92, Theorem 2]). Also, this class of functions is precisely the class computed by polynomial-size skew arithmetic circuits over the integers [To92].

The question of the relationship between #L and $\#\mathrm{SAC}^1$ is thus exactly the question asked in [Va79], concerning the relationship between the determinant and circuits of polynomial size and degree.

First we note that there is a tantalizing connection between TC^1 and $\#\mathrm{SAC}^1$.

Theorem 4.2 *A function is computed by TC^1 circuits iff it is computed by arithmetic circuits over the natural numbers, with depth $O(\log n)$, polynomial size, with unbounded fan-in $+$ gates, and fan-in two $\times$ and $\div$ gates.*

Here, $\div$ is integer division, with the remainder discarded.

Proof: Since unbounded fan-in $+$, and $\times$ and $\div$ can be computed by TC^0 circuits [RT92], inclusion from right-to-left is straightforward. (This is true even for logspace-uniformity, since it follows from [BCH86] and [RT92] that division can be done in uniform TC^0 if the number N is given, where N is the product of the first n^2 primes. But N can be computed in TC^1.)

To see the other direction, note that the MAJORITY of $x_1, \ldots, x_n$ is equal to $(\sum_{i=1}^n x_i) \div 2^{\lfloor \log n \rfloor}$. The subcircuits computing powers of 2 can be re-used; thus $O(\log n)$ layers of MAJORITY gates can be simulated with $O(\log n)$ levels of arithmetic gates. ∎

We note that other (less trivial) connections between TC^0 and classes of arithmetic circuits over finite fields are also known [RT92, BFS92].

In spite of Theorem 4.2, it is not known if TC^1 or even AC^1 can be reduced to arithmetic circuits of polynomial size and degree ($\#\mathrm{SAC}^1$). It is a trivial

[1] Formally, it is necessary to include the element $-\infty$ in order to make this structure a semiring. This is irrelevant for our purposes.

[2] That is, there is a logspace-computable g such that $f(x) = \text{determinant}(g(x))$.

observation that AC^1 can be reduced to arithmetic circuits over the integers of polynomial size and degree $n^{O(\log n)}$. The following result improves this trivial bound to $n^{O(\log \log n)}$, at least in the non-uniform setting. (Note that arithmetic circuits of nonpolynomial degree can produce output of more than polynomial length. The following proof does not make use of this capability; only the information in the low-order $O(\log n)$ bits is used.)

Theorem 4.3 *Every language in AC^1 is efficiently reducible to a function computed by polynomial-size, degree $n^{O(\log \log n)}$ arithmetic circuits over the natural numbers.*

Proof: We use the by-now-standard simulation of AND and OR gates by parity gates and ANDs of small fan-in (used, for example, in [VV86, To91, AH93, KVV93]). More precisely, in Lemma 13 of [AH93] there is an explicit construction showing how, for any polynomial $p(n)$, one can construct a probabilistic, constant depth circuit of AND gates with fan-in $O(\log n)$ and PARITY gates, simulating an AND or OR gate with error probability at most $1/p(n)$. If we take an AC^1 circuit and perform this substitution simultaneously for all the AND and OR gates, one obtains a probabilistic, polynomial-size circuit that, with high probability, provides the same output as the original circuit. Note that replacing AND gates by $\times$ and PARITY gates by $+$, one obtains an arithmetic circuit, the low-order bit of whose output is the same as the output of the original AC^1 circuit with high probability. The degree of this circuit is $O(\log n)^{O(\log n)} = n^{O(\log \log n)}$

It remains to make the circuit deterministic. First we make use of the "Toda polynomials" introduced in [To91]. For example, there is an explicit construction in [BT91] of a polynomial P_k of degree $2k-1$ such that $P_k(y) \bmod 2^k = y \bmod 2$. If we implement this polynomial in the obvious way and apply it to the arithmetic circuit constructed in the preceding paragraph, we obtain an arithmetic circuit of degree $n^{O(\log \log n)}$ and polynomial size, whose low order bit is the same as the output of the original AC^1 circuit with high probability, and with the additional property that the other $2 \log n$ low-order bits of the result are always zero.

If we now take $2n$ independent copies of this probabilistic circuit, and add the results, then bit number $1 + \lfloor \log n \rfloor$ yields the correct answer with only an exponentially small error probability (since this represents the MAJORITY of $2n$ independent trials). It now follows (via the standard argument of e.g. [Ad78]) that there is a single setting of the probabilistic bits that gives the correct answer on all inputs. Hardwiring this setting of the probabilistic bits into the circuit yields the desired arithmetic circuit. ∎

An obvious question is whether this also holds in the uniform setting.

5 Open Problems

It is known that for commutative semirings, arithmetic circuits of polynomial degree are exactly as powerful as circuits of logarithmic depth (where $\times$ gates have bounded fan-in, and $+$ gates have unbounded fan-in). It is natural to wonder if the same result is true also over $(\Sigma^*, \max, \text{concat})$.

Is OptSAC^1 any more difficult than LOGCFL? We remark that it is still not known if OptL is contained in LOGCFL.

Is $\#SAC^1$ hard for TC^1 under any reasonable notion of hardness?

6 Appendix

Here, we show that the [VSBR83] depth-reduction results for arithmetic circuits hold also in the uniform setting.[3] For the particular case of the Boolean ring this is proved in [Ve91], and for the integers it is proved in [Vi91]. The proof for the case of general commutative semirings does not seem to have appeared before.

Theorem 6.1 *Let R be any commutative semiring. The class of functions computed by uniform arithmetic circuits over R of polynomial degree is equal to the class of functions computed by uniform semi-unbounded arithmetic circuits over R of depth $O(\log n)$.*

(Recall that semi-unbounded circuits have $\times$ gates with fan-in two, and $+$ gates with unbounded fan-in.

Proof: The first step is to show that for any uniform polynomial-degree circuit family there is an equivalent one with the property that each gate is labeled with its formal degree. Let C be an arithmetic circuit and assume that there are no consecutive $+$ gates. (If necessary, insert "$\times 1$" gates between each two consecutive $+$ gates; this does not cause the degree to become non-polynomial.) Now let C' be the circuit constructed as follows: for each gate g in C, build gates $(g,1), (g,2), \ldots, (g, n^k)$ (where n^k is an upper bound on the degree of C). If g is a $+$ gate, then (g,i) is a $+$ gate with children $\{(h,i) : h \text{ is a child of } g\}$. If g is a $\times$ gate, then (g,i) is a $+$ gate with children that are $\times$ gates of the form $(h_1, j) \times (h_2, i-j)$, where h_1 and h_2 are the children of g, and $1 \le j \le i-1$. If h is *not* a leaf, then make $(h,1)$ a leaf with value 0. If h *is*

[3] The argument provided in [VSBR83] requires that the degree of each gate be known. Although this can be computed quickly in parallel [MRK88] it is easy to see that this is hard for NLOG, and thus cannot be assumed in the logspace-uniform circuit model.

a leaf and $i > 1$, then make (h, i) the root of a trivial subcircuit with degree i and value 0. The output gate of C' is the sum of all gates (g, i), where g is the output gate of C.

It is easy to prove by induction on i that each gate (h, i) in C' has as its value the sum of all monomials of degree i in the formal polynomial corresponding to gate h in circuit C. Thus C' is equivalent to C.

The following technical definition will be useful later in the argument. Say that g and h are *+-adjacent* if there is a path from g to h where all intermediate nodes are + gates. Note that C' has the property that if g and h are +-adjacent, then the path of + edges connecting g and h must be unique and have length at most 3, and there is not any path from g to h through a $\times$ gate. Among other things, this guarantees that it is easy to check in $O(\log n)$ space if g and h are +-adjacent.

Also, note that it is easy to re-write C' so that the first child of any $\times$ gate has degree no more than the degree of the second child.

An *exploration* of gate g is a depth-first search of a subcircuit rooted at g, with the property that

- For each + node, a child is chosen nondeterministically to explore.
- For each $\times$ node the second child is put on the stack and the first child is explored.
- When a leaf is encountered, the stack is popped and the node on top of the stack is explored (unless the stack is empty, in which case the exploration stops). The leaf that is the last node visited is the *terminal node* of the exploration.

Note that, by our guarantee that the degree of the first child of a $\times$ node is no more than half the degree of its parent, it follows that the degree of the node being explored decreases by 1/2 each time the stack height increases. Thus the stack height is logarithmic on any exploration. Let the *exploration height* of a node be the maximum stack height of any exploration of the node.

Clearly the value of g is the sum (over all explorations e of g) of the product of all leaves encountered on e. (This can be verified by an easy induction starting at the leaves.)

Given gate g and leaf l, define $[g, l]$ to be the sum over all explorations e of g having terminal leaf l of the product of all leaves encountered on e. Thus the value of g is the sum over all leaves l of $[g, l]$.

More generally, for any two gates g and h where h is *not* a leaf, let $[g, h]$ denote the value determined by the definition in the preceding paragraph, where gate h is replaced by a leaf with value 1. We will show how to build a circuit computing the values $[g, h]$ for *all* h (leaf and non-leaf).

If g is a leaf, then $[g, g]$ is a leaf returning the value of g, and for all other h, $[g, h]$ is a leaf with value 0.

If g is a + gate, then $[g, h]$ is simply the sum of all $[g', h]$, where g' is a child of g.

If g is a $\times$ gate and g and h are +-adjacent and h *is* a leaf, then $[g, h]$ should simply return $h \times$ (the value of g_1), where g_1 is the first child of g. (This is because there is a one-to-one correspondence between explorations of g and explorations of g_1, because of the uniqueness of the + path connecting g and h). Thus if g is a $\times$ gate and g and h are +-adjacent and h is a leaf, then $[g, h]$ is $h \times$ (the sum over all leaves l of $[g_1, l]$).

Similarly, if g is a $\times$ gate and g and h are +-adjacent and h is *not* a leaf, then $[g, h]$ is the sum over all leaves l of $[g_1, l]$. (This is because h is treated as a leaf with value 1.)

If g is a $\times$ gate and g and h are *not* +-adjacent, then the definitions imply that $[g, h]$ is equal to 0 unless there is an exploration of g with h as a terminal node (where h is treated as a leaf). For each such exploration there is a *unique* sequence of gates $g = g_0, g_1, \ldots, g_m = h$ such that the *second* child of g_i is +-adjacent to g_{i+1}, and each g_i is a $\times$ gate (except possibly $g_m = h$). (That is, being the terminal node of an exploration is equivalent to being reachable via a path using only + gates and the *second* edges out of $\times$ gates.) In such a sequence there is *exactly* one g_i such that[4] $\text{degree}(g_i) \geq (\text{degree}(g) + \text{degree}(h))/2 >$ degree of the second child of g_i. The product of the leaves encountered along this exploration is the product of the leaves encountered before g_i and those encountered after g_i. It follows that $[g, h]$ is the sum over $\{g_i$ such that $\text{degree}(g_i) \geq (\text{degree}(g) + \text{degree}(h))/2 >$ degree of the second child of $g_i\}$ of $([g, g_i] \times [g_i, h])$.

Clearly the resulting circuit is of polynomial size, and is semi-unbounded. To analyze the height of the circuit, observe that the subcircuit evaluating $[g, h]$ when g and h are +-adjacent depends on subcircuits of the form $[g', h']$, where the exploration height of g' is one less than the exploration height of g. Also note that if g and h are not +-adjacent, then the subcircuit evaluating $[g, h]$ depends on subcircuits of the form $[g', h']$ where $\text{degree}(g') -$ $\text{degree}(h')$ is no more than half of $\text{degree}(g) -$ $\text{degree}(h)$. It follows that the height is $O(\log n)$. ∎

Acknowledgments

We thank Birgit Jenner and V. Vinay for their helpful comments.

[4]In this expression, "degree(g)" refers to the degree of g in circuit C', not its degree in the subcircuit being explored (where h is a leaf). Similarly, "degree(h)" is the degree of h in C'. Note that by the construction of C', we can assume that these degrees are explicitly encoded in the names of g and h.

References

[Ad78] L. Adleman, *Two theorems on random polynomial time* Proc. 19th FOCS (1978) pp. 75–83.

[Al89] E. Allender, *A note on the power of threshold circuits*, Proc. 30th FOCS, pp. 580–584.

[ABP92] E. Allender, D. Bruschi, and G. Pighizzini, *The complexity of computing maximal word functions*, DIMACS tech report 92-15.

[AH90] E. Allender and U. Hertrampf, *On the power of uniform families of constant depth threshold circuits*, Proc. 15th MFCS, 1990, Lecture Notes in Computer Science 452, pp. 158–164.

[AH93] E. Allender and U. Hertrampf, *Depth reduction for circuits of unbounded fan-in*, to appear in Information and Computation. Preliminary versions appeared as [Al89, AH90]

[AJ92] C. Álvarez and B. Jenner, *A note on log space optimization*, report, L.S.I., Universitat Politècnica Catalunya, Barcelona, 1992.

[AJ93] C. Álvarez and B. Jenner, *A very hard log-space counting class*, Theoretical Computer Science 107 (1993) 3–30.

[BCH86] P. W. Beame, S. A. Cook, and H. J. Hoover, *Log depth circuits for division and related problems*, SIAM J. Comput. 15 (1986) 994–1003.

[BT91] R. Beigel and J. Tarui, *On ACC*, Proc. 32nd FOCS (1991) 783–792.

[BFS92] J. Boyar, G. Frandsen, and G. Sturtivant, *An arithmetical model of computation equivalent to threshold circuits*, Theoretical Computer Science 93 (1992) 303–319.

[Co71] S. Cook, *Characterization of pushdown machines in terms of time-bounded computers*, J. ACM 18 (1971) 4–18.

[Da91] C. Damm, $DET = L^{\#L}$*?*, Informatik-Preprint 8, Fachbereich Informatik der Humboldt-Universität zu Berlin, 1991.

[Je93] B. Jenner, personal communication.

[JS82] M. Jerrum and M. Snir, *Some exact complexity results for straight-line computations over semirings*, J. ACM 29 (1982) 874–897.

[KVV93] R. Kannan, H. Venkateswaran, V. Vinay, and A. Yao, *A circuit-based proof of Toda's theorem*, to appear in Information and Computation.

[Ko90] S. R. Kosaraju, *On the parallel evaluation of classes of circuits*, Proc. 10th FST&TCS, Lecture Notes in Computer Science 472 (1990) 232–237.

[Kr88] M. Krentel, *The complexity of optimization problems*, JCSS 36 (1988) 490–509.

[MRK88] G. Miller, V. Ramachandran, and E. Kaltofen, *Efficient parallel evaluation of straight-line code and arithmetic circuits*, SIAM J. Comput. 17 (1988) 687–695.

[MT87] G. Miller and S.-H. Teng, *Dynamic parallel complexity of computational circuits*, Proc. 19th STOC (1987) 478–489.

[Ni91] N. Nisan, *Lower bounds for non-commutative computation*, Proc. 23rd STOC (1991) 410–418.

[RT92] J. Reif and S. Tate, *On threshold circuits and polynomial computation*, SIAM J. Comput. 21 (1992) 896–908.

[To91] S. Toda, *PP is as hard as the polynomial-time hierarchy* SIAM J. Comput. 20 (1991) 865–877.

[To91a] S. Toda, *Counting problems computationally equivalent to the determinant*, manuscript.

[To92] S. Toda, *Classes of arithmetic circuits capturing the complexity of computing the determinant*, IEICE Trans. Inf. and Syst., vol. E75-D (1992) 116–124.

[Va79] L. Valiant, *Completeness classes in algebra*, Proc. 11th STOC (1979) 249–261.

[Va92] L. Valiant, *Why is Boolean complexity theory difficult?* in *Boolean Function Complexity*, edited by M. S. Paterson, London Mathematical Society Lecture Notes Series 169, Cambridge University Press, 1992.

[VSBR83] L. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff, *Fast parallel computation of polynomials using few processors*, SIAM J. Comput. 12 (1983) 641–644.

[VV86] L. Valiant and V. Vazirani, V, *NP is as easy as detecting unique solutions* Theoretical Computer Science 47 (1986) 85–93.

[Ve91] H. Venkateswaran, *Properties that characterize LOGCFL*, JCSS 42 (1991) 380–404.

[Ve92] H. Venkateswaran, *Circuit definitions of nondeterministic complexity classes*, SIAM J. Comput. 21 (1992) 655–670.

[VT89] H. Venkateswaran and M. Tompa, *A new pebble game that characterizes parallel complexity classes*, SIAM J. Comput. 18 (1989) 533–549.

[Vi91] V. Vinay, *Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits*, Proc. 6th IEEE Structure in Complexity Theory Conference (1991) 270–284.

[Vi91a] V. Vinay, *Semi-unboundedness and complexity classes*, doctoral dissertation, Indian Institute of Science, Bangalore.