On Monotone Planar Circuits

David A. Mix Barrington
Comp. Science Dept.
Univ. of Massachusetts
Amherst, MA 01003 U.S.A.
barring@cs.umass.edu

Chi-Jen Lu
Dept. Comp. Sci. & Inf. Eng.
National Chi-Nan University
Pu-Li, Nan-Tou, Taiwan
cjlu@csie.ncnu.edu.tw

Peter Bro Miltersen and Sven Skyum
BRICS and Dept. Comp. Sci.
University of Aarhus
8000 Aarhus C, Denmark
{bromille,sskyum}@brics.dk

Abstract

In this paper we show several results about monotone planar circuits. We show that monotone planar circuits of bounded width, with access to negated input variables, compute exactly the functions in non-uniform $\mathbf{AC}^0$*. This provides a striking contrast to the non-planar case, where exactly* $\mathbf{NC}^1$ *is computed. We show that the circuit value problem for monotone planar circuits, with inputs on the outer face only, can be solved in* **LOGDCFL** $\subseteq$ **SC***, improving a* **LOGCFL** *upper bound due to Dymond and Cook. We show that for monotone planar circuits, with inputs on the outer face only, excessive depth compared to width is useless; any function computed by a monotone planar circuit of width* w *with inputs on the outer face can be computed by a monotone planar circuit of width* $O(w)$ *and depth* $w^{O(1)}$*. Finally, we show that monotone planar read-once circuits, with inputs on the outer face only, can be efficiently learned using membership queries.*

1 Introduction

In this paper, we prove a number of results about the computational properties of *planar*[1] Boolean circuits of $\vee$ and $\wedge$ gates, continuing a line of research of Goldschlager [10] and of Dymond and Cook [9]. A circuit is defined to be planar if it is embedded in $\mathbf{R}^2$ without any cross-over of wires. Negation gates are not allowed in our circuits, but in the case of circuits defining Boolean functions, we do allow input variables to be negated so that they can compute non-monotone Boolean functions.

Our first result concerns planar circuits of restricted *width*. It is well known that there is a rough relationship between *width* restricted and *depth* restricted circuits, but that this relationship is not a complete equivalence [11, 2]. For instance, the class of functions computed by a family of circuits of *quasi*-polynomial size and polylogarithmic depth is equal to the class of functions computed by a family of circuits of quasi-polynomial size and polylogarithmic width. On the other hand, the class of functions computed by a family of circuits of *polynomial* size and polylogarithmic width (non-uniform **SC**) is, in general, conjectured to be different from the class of functions computed by a family of circuits of polynomial size and polylogarithmic depth (non-uniform **NC**). For the case of *constant* depth and width, there is a provable discrepancy; the class of functions computable by constant depth circuits of polynomial size, i.e, $\mathbf{AC}^0$, is a proper subset of the functions computable by constant width circuits of polynomial size, the latter being, by Barrington's Theorem [1], $\mathbf{NC}^1$. We show that if we restrict ourselves to planar circuits, and only consider circuits with no negation gates (though inputs can be negated) the discrepancy between constant width and constant depth disappears! It is easy to see that polynomial size, constant depth circuits can be made planar and negation-free with only a polynomial blow up in size, so such circuits still characterize non-uniform $\mathbf{AC}^0$. (Here we assume that each input is available to the circuit everywhere.) On the other hand, we prove that if we place the same restrictions of planarity and monotonicity on polynomial size, constant width circuits they drop in computational power from $\mathbf{NC}^1$ to (exactly) $\mathbf{AC}^0$. To be precise, we prove:

Theorem 1 *The class of Boolean functions computed by a family of planar, constant width, polynomial size Boolean circuits without negation gates is* exactly *non-uniform* $\mathbf{AC}^0$.

The same result holds in the uniform setting (with any standard desired level of uniformity, e.g., that of [3]).

Our second result concerns monotone planar circuits with no restriction on width. However, we restrict the inputs of the circuit to appear on the outer face of the circuit only. Let us call such circuits *stratified*. A classical result of

[1] There is a useful distinction to be made between *planar* circuits (those that *can* be embedded in the plane) and *plane* circuits (those that *are* embedded in the plane). This paper deals entirely with the latter, as we always assume that the embedding is given with the circuit. However, we will use the word "planar" for consistency with the prior literature.

Dymond and Cook [9] states[2] that the circuit value problem for this class of circuits is in **LOGCFL**. We improve this to the following

Theorem 2 *The circuit value problem for stratified monotone planar circuits is in* **LOGDCFL***.*

As **LOGCFL** is a subset of $\mathbf{NC}^2$, it is a corollary of Dymond and Cook's result that the class **MPC** of functions computed by polynomial size stratified monotone planar circuits is a subset of non-uniform $\mathbf{NC}^2$. As **LOGDCFL** is a subset of $\mathbf{SC}^2$ by a result of Cook [7], it is a corollary of our result that **MPC** is a subset of non-uniform $\mathbf{SC}^2$ as well. Johnson [13], in his survey on complexity classes, commenting on Dymond and Cook's result, remarks that it is not known if the circuit value problem for stratified monotone planar circuits is *complete* for **LOGCFL**. Theorem 2 gives strong indication that this is *not* the case: if it is, **LOGCFL** and hence **NL** is a subset of **SC**.

Our next result concerns the relationship between the the number of input nodes and the width and the depth of a stratified planar circuit. We show that it is useless to have a circuit of dimensions much bigger than the number of input nodes. To be precise:

Theorem 3 *If a function is computed by a stratified planar circuit with at most n input nodes, it is also computed by a stratified planar circuit of width n and depth n^3.*

Our final result concerns the learnability of stratified monotone planar circuits. Suppose we are given a black box with n inputs and one output and are told that the box contains a stratified monotone planar circuit. Furthermore, each input of the box corresponds to exactly one input wire of the circuit. Finally, we are told that the input wires on the box appear in the same sequence as they appear on the outer face of the circuit. We show that with $n^{O(1)}$ queries of the form "Given assignment v to the inputs, what is the output?", we can efficiently deduce the function computed by the box and also exhibit a circuit of the correct kind computing the same function. Summarizing,

Theorem 4 *Ordered read-once, stratified, monotone planar circuits are polynomially learnable using membership queries.*

The keys to all our results are a few simple propositions about the way stratified planar monotone circuits compute. Even Theorem 1, though it is about non-stratified circuits, depends in large part on our observations of stratified circuits. The propositions are very similar to observations by Goldschlager [10] (leading, for instance, to Dymond and Cook's **LOGCFL** result), but with one crucial difference. Goldschlager takes a top-down view of the computation. He starts at the output and goes backwards in time, constructing a *proof tree* for the computation. A vertex in such a proof tree is an *interval* of true gates at a layer of the circuit. We take a bottom-up view, going forward in time. Consider the computation on a fixed input. At any point in time (i.e. in any layer of the circuit), the computation has arrived at a number of intervals, each of which contains only gates that evaluate to true. At the next point in time, these intervals may merge or disappear, *but they do not split up*. All our results are proved by taking advantage of this simple fact. It is also interesting to notice that Goldschlager's proof trees are trees, rather than graphs, *precisely* because intervals do not split up.

[2] One should note that it is actually not explicitly mentioned by Dymond and Cook that the circuits have to be stratified, but it is clear from their proof that this is a necessity. It is also an explicitly mentioned assumption in Goldchlager [10] whose algorithm forms the basis for Dymond and Cook's, and it is also mentioned by Delcher and Kosaraju [8] (whose objective it was to eliminate the assumption) that Dymond and Cook's algorithm only works under this assumption.

1.1 Other related research

We show that constant width planar circuits can be used to characterize $\mathbf{AC}^0$. Istrail and Zivkovic [12] had previously shown that constant width Boolean *formulas* have exactly the power of $\mathbf{AC}^0$. The present authors showed in [4] that $\mathbf{AC}^0$ can also be characterized by constant width planar *branching programs*. Interestingly, the techniques used in [4] are completely different from the ones used in the present paper; in [4] we gave a direct transformation of planar branching programs to circuits, while we in this paper have to use an indirect algebraic argument.

We improve Cook and Dymond's upper bound on the power of polynomial sized planar stratified circuits from **LOGCFL** to **LOGDCFL**. It is of course interesting to ask what happens if the circuit is not stratified. It is not known if Cook and Dymond's **LOGCFL** upper bound holds if the circuit is not stratified, but Delcher and Kosaraju [8] and Ramachandran and Yang [15] have shown that there still is an **NC** upper bound.

1.2 Organization of paper

In Section 2, we review the definition of a planar monotone circuit in standard (layered) form on which we will base our discussion. In Section 4, we discuss monotone planar circuits of constant width and prove Theorem 1. In Section 5, we prove Theorem 2. In Section 6, we discuss the proofs of Theorem 3 and Theorem 4. As these proofs are somewhat more involved than the proofs of Theorem 1 and Theorem 2, they will only be sketched in this version of the paper.

2 Preliminaries

We shall adopt the definition of Goldschlager and Dymond and Cook of a monotone planar circuit in standard (i.e., layered) form. It is straightforward to convert any circuit that can be embedded in the plane without crossover of wires into this form. Thus, a monotone planar circuit is a collection of nodes g_i^l, $l = 0, \dots, L$, $1 \le i \le w_l$, together with their edges. The nodes g_i^l, $1 \le i \le w_l$ are called the nodes at *layer* l. The width of the circuit at layer l is w_l. The width of the circuit is the maximum value of w_l, taken over all l. The depth of the circuit is L. A node g_i^l at layer l is one of the following:

- An input node of fan-in 0. When considering circuits as inputs to the circuit value problem, such a node is assigned a Boolean input, 0 or 1. When considering circuits computing Boolean functions, such a node is assigned an input variable or its negation.

- An AND or OR gate of fan-in two, taking two inputs from layer $l - 1$.

- A COPY gate of fan-in one, taking one input from layer $l - 1$ which it copies.

The planarity constraint is captured as follows: When an AND or OR gate g_i^l takes as inputs g_a^{l-1} and g_b^{l-1} with $a < b$, then, if $j < i$, the gate g_j^l is only allowed to take inputs g_c^{l-1} for values of $c \le a$. Similarly, if $j > i$, the gate g_j^l is only allowed to take inputs g_c^{l-1} for values of $c \ge b$. A similar rule holds for COPY gates: If a copy gate g_i^l takes as input g_a^{l-1}, gates g_j^l for $j < i$ only takes inputs g_c^{l-1} for $c \le a$ and gates g_j^l for $j > i$ only takes inputs g_c^{l-1} for $c \ge a$.

One gate in the circuit is designated as the output gate. Without loss of generality, this gate is at the last layer L. Note that all nodes in layer 0 must be input nodes. We say that a circuit is *stratified* if, in fact, all input nodes occur at layer 0.

3 Stratified circuits and maximal true intervals

Given a circuit C and an input assignment, each gate in C takes a value, 0 or 1. At each layer l of the circuit, we can read off a vector v_l in $\{0,1\}^{w_l}$ of truth values of $g_1^l, g_2^l, \dots g_{w_l}^l$. It will be convenient to identify such a vector with its *set of maximal true intervals*. For example, we shall identify a vector $(0,0,1,1,1,0,1,0,1,1)$ with the set of intervals $\{[3,5],[7],[9,10]\}$. A singleton $\{I\}$ will be identified with the interval I itself. For a stratified circuit C, we use the notation $C_l(v_0)$ for the value of v_l given assignment v_0 to the input layer. Furthermore, $C(v_0)$ denotes the value $C_L(v_0)$ of the final layer.

In the following, let C be a stratified monotone planar circuit. The basic property of stratified monotone planar circuits which we shall take advantage of is this (all propositions in this section can be proved by simple induction):

Proposition 5 *Suppose v_0 consists of a single maximal true interval. Then, for all l, v_l is either a single interval or the all-0 vector.*

If v_0 is an interval, and v_l is the all 0-vector, we say that the interval *is dead* at layer l. The interval *dies* at the earliest layer where it is dead.

Let I be an interval in v_0. By monotonicity, $C_l(I)$ is a subinterval of some interval J in $C_l(v_0)$. We say that I *gives rise* to J at layer l. If two intervals I and K in v_0 together give rise to only the single interval J at layer l, we say that *I and K have merged* at layer l. If l is the earliest layer where this is the case, we say that *I and K merge at layer l*. Note that the interval J may be bigger than the union of the two intervals that the original two intervals in v_0 would have given rise to on their own. Indeed, it could be that the two intervals each would have died on their own on layer l. On the other hand, *if v_l* has the same number of intervals as v_0, there is no interference:

Proposition 6 *If the number of intervals in $C_l(v_0)$ is the same as the number of intervals in v_0, then $C_l(v_0) = \{C_l(I) | I \in v_0\}$, i.e., each interval in v_l corresponds to an interval in v_0 and the interval in v_l is exactly the same as it would have been had the corresponding interval in v_0 been the only one.*

If we extend an interval all the way to the left, the same thing will happen to all intervals the interval gives rise to.

Proposition 7 *If $C_l([i,j]) = [i',j']$ then $C_l([1,j]) = [1,j']$ and $C_l([i,n]) = [i',n]$.*

Intervals may merge or die. When evaluating a circuit, this fact provides a way of simplifying the input without changing the output. Suppose two intervals merge. Then a 0-region is enclosed in the 1-region of these two intervals. So switching 0's to 1's between the two intervals in the input does not change the outer boundary of this 1-region, nor the output value. Dually, suppose an interval dies before merging with others or reaching the output layer. It means that two 0-regions merge and enclose the 1-region. We can safely discard this interval. Formally, we have the following two propositions:

Proposition 8 *Let I and J be two maximal intervals in v_0. Suppose I and J merge in the evaluation of $C(v_0)$. Then $C(v_0) = C((v_0 \setminus \{I,J\}) \cup \{I \vee J\})$, where $I \vee J$ is the smallest interval containing I and J.*

Proposition 9 *Let I be an interval in v_0. Suppose I dies before merging with other intervals or reaching the output layer. Then $C(v_0) = C(v_0 \setminus \{I\})$.*

4 Constant width monotone planar circuits capture $\mathbf{AC}^0$

In this section we prove Theorem 1, essentially by reducing the evaluation of a constant width planar circuit to the word problem of a certain aperiodic monoid.

Let C be a stratified planar monotone circuit of depth L. Assume the number of nodes w_L at layer L equals the number of nodes w_0 in layer 0, i.e. $w_L = w_0 = k$. Recall that we use $C(v)$ to denote the vector of truth values at the L'th layer, given the assignment v to layer 0. So C is a map from $\{0,1\}^k$ to itself.

Let M_k be the set of functions from $\{0,1\}^k \to \{0,1\}^k$ computable as the function from x to $C(x)$ for some monotone stratified circuit C with k nodes in layers 0 and L. This set forms a monoid under functional composition. Recall that a monoid M is *aperiodic* if there is a non-negative integer m such that for any a in M, $a^{m+1} = a^m$. We shall show that M_k is aperiodic. First, a lemma.

Lemma 10 *For any vector v, if $C^t(v) = v$ for some $t \geq 1$, then $C(v) = v$.*

Proof For any vector y, the number of intervals in $C(y)$ is less than or equal to the number of intervals in y. As $C^t(v) = v$, the number of intervals in $C^1(v), C^2(v), \ldots, C^t(v)$ are the same. Furthermore, by Proposition 6 each interval I in v corresponds to exactly one interval in each of $C^1(v), C^2(v), \ldots$, and these intervals are the same as they would have been, had I been propagated alone. Thus, we have $C^t(I) = I$. We shall argue that for each such interval I in v, $C(I) = I$, and we will be done.

Let $I = [i,j]$ be a maximal true interval in v. Assume, to the contrary, that $C([i,j]) \neq [i,j]$. There are four cases we have to rule out:

- *C maps $[i,j]$ to a proper subinterval of $[i,j]$.* By monotonicity, $C^t([i,j])$ would then also be a proper subinterval of $[i,j]$, contradicting the fact that $C^t([i,j]) = [i,j]$.
- *C maps $[i,j]$ to a proper superinterval of $[i,j]$.* By monotonicity, $C^t([i,j])$ would then also be a proper superinterval of $[i,j]$, contradicting the fact that $C^t([i,j]) = [i,j]$.
- C maps $[i,j]$ to $[i',j']$ with $i' < i$ and $j' < j$. By Proposition 7, this means that $C([1,j]) = C([1,j'])$. By monotonicity, $C^t([1,j]) = [1,j'']$ for some $j'' \leq j'$ and, again by monotonicity, $C^t([i,j])$ is a subinterval of $[1,j'']$. As $j'' < j$, this contradicts the fact that $C^t([i,j]) = [i,j]$.
- C maps $[i,j]$ to $[i',j']$ with $i' > i$ and $j' > j$. By Proposition 7, this means that $C([i,n]) = C([i',n])$. By monotonicity, $C^t([i,n]) = [i'',n]$ for some $i'' \geq i'$ and, again by monotonicity, $C^t([i,j])$ is a subinterval of $[i'',n]$. As $i'' > i$, this contradicts the fact that $C^t([i,j]) = [i,j]$.

So, $C(I) = I$ whenever I is a maximal true subinterval, and so $C(v) = v$. □

Lemma 11 *For any k, M_k is aperiodic.*

Proof Consider M_k and any element $C \in M_k$. We want to show that $C^{m+1} = C^m$ for some m. Because M_k has only a finite number of elements, the sequence $C^0, C^1, C^2, \ldots$ must enter a loop at some point. So there exist natural numbers m and $l \geq 1$ such that $C^{m+l} = C^m$. Let v be any input vector, and let $C^m(v) = v'$. Then $v' = C^m(v) = C^{m+l}(v) = C^l(v')$. From Lemma 10, $C(v') = v'$, and $C^{m+1}(v) = C(v') = v' = C^m(v)$. As this works for arbitrary v, $C^{m+1} = C^m$, and since C was arbitrary, M_k is aperiodic. □

The iterated multiplication problem for a finite aperiodic monoid is in uniform $\mathbf{AC}^0$, by a result of Chandra, Fortune and Lipton [6] (see also Barrington and Thérien [5]). Now suppose we are given a width k and depth L stratified monotone planar circuit computing a Boolean function. First, assume that the width is k at all layers — if it is less, we can add dummy COPY nodes. If k is a constant, we can split the circuit into L segments of depth 1, view each as an element of M_k and compute the iterated product in $\mathbf{AC}^0$ (i.e. by a Boolean circuit of depth k' and size $L^{k''}$ where k' and k'' are constants depending on k). This gives us a description of a single element of M_k (i.e. a constant piece of information, for example in form of a lookup table) describing exactly which function we are computing. We then evaluate this function on our input.

The problem with the above approach, is that it only works for stratified circuits, i.e. all input nodes must be in the layer 0. But as the width is constant, we can only have a constant number of inputs. Any function on a constant number of inputs is trivially in $\mathbf{AC}^0$ anyway! To show Theorem 1, we have to consider non-stratified circuits.

To evaluate a non-stratified circuit in $\mathbf{AC}^0$ in a way similar to the above, we have to consider the following monoid N_k: Let N_k be the set of functions from $\{0,1\}^k$ to $\{0,1\}^k$ that are computable as the function from x to $C(x)$ for some planar monotone circuit C, with k nodes in each of layers 0 and L. The circuit does not have to be stratified, but any internal input node is pre-assigned a constant value 0 or 1.

Now, suppose we can show that N_k is aperiodic. Then we can evaluate any width k monotone planar circuit, not necessarily stratified, as follows. Take the input assignment and assign the correct values to all input nodes not in layer 0. Split the circuit into L segments of depth 1, view each as an element of N_k and compute the iterated product in $\mathbf{AC}^0$. This gives us a description of a single element of N_k, i.e. a constant piece of information. This function we evaluate on the input of the layer 0 by table lookup.

Thus, if we can show N_k to be aperiodic, we have the following Theorem:

Theorem 12 *For any fixed k, the circuit value problem for monotone planar circuits of width k is in uniform* $\mathbf{AC}^0$*, i.e. it can be solved by an alternating Turing machine with random access to the input, running in logarithmic time and making* $O(1)$ *alternations.*

This immediately gives us one direction of Theorem 1 of the introduction. The other direction is easy: An $\mathbf{AC}^0$ circuit of depth d can easily be converted into a monotone planar circuit of width d by first expanding the circuit into a formula and then replacing each unbounded fan-in AND or OR gate with a chain of fan-in two AND or OR gates. (This is essentially the argument of [12].)

To complete the proof of Theorem 12, we only need to show that N_k is aperiodic. Fortunately, we can show this by reducing to the case of M_k.

Lemma 13 *For any constant k,* N_k *is aperiodic.*

Proof Consider N_k and any element $C \in N_k$. In C, some output gates may be forced to constants because of the existing constants in the circuit. Then in C^2, more output gates may be forced to constants because some inputs to the second copy of C are fixed. By composing more copies of C, possibly more and more output gates are forced to constants. However, for some i, the set of output gates fixed in C^j, for all $j > i$, remains the same as the set fixed in C^i. On top of that, each additional C, with k' inputs and k' outputs fixed, behaves like a function from $\{0,1\}^{k-k'}$ to itself. It's not hard to see that this corresponding function belongs to $M_{k-k'}$. From Lemma 11, for some natural number m, $C^{i+m+1} = C^{m+1} \circ C^i = C^m \circ C^i = C^{i+m}$. As C was arbitrary, N_k is aperiodic and we are done. ☐

This completes the proof of Theorem 12. It is interesting to note that the way we prove Theorem 12 and hence Theorem 1 does not give any good bound on blowup of constants when going from a constant width planar circuit to a constant depth circuit. A naive estimate, based only on the fact that N_k is aperiodic and inspecting the proof of the fact that the word problem for aperiodic monoids can be solved in $\mathbf{AC}^0$, bounds the depth only by a doubly exponential function of the width! Naturally, we conjecture that this blowup is very far from the best possible; in fact we conjecture that a linear blowup is sufficient (as is the case when going from depth to width). Presumably, a better understanding of the structure of the monoid N_k would lead to a better upper bound.

An interesting corollary of Theorem 12 which can be shown using standard "lifting" techniques (so we omit the proof from this version of the paper) is the following. Consider a planar circuit of width k and size $2^{n^{O(1)}}$ taking n input variables, copies of which and negations of which can appear at input nodes anywhere in the circuit. We say that the circuit is polynomial time uniform if there is a deterministic polynomial time algorithm which on input $\langle l, i\rangle$ outputs the type of node g_i^l and, if it is a gate, the indices of its predecessors at layer $l-1$, and, if it is an input node, the associated input variable and whether it is negated. Then,

Corollary 14 *A Boolean function can be computed by a polynomial time uniform, exponential sized, constant width planar circuit if and only if it is in the polynomial hierarchy* **PH**.

5 Evaluating stratified circuits in LOGDCFL

In this section, we prove Theorem 2, i.e., we show that the circuit value problem for stratified monotone planar circuits is in **LOGDCFL**. Recall that **LOGDCFL** can be characterized as the class of languages computed by a deterministic Turing machine using polynomial time, logarithmic space, and access to a *pushdown store*.

We need the following lemma which tells us that if we restrict the input, the evaluation problem is even simpler.

Lemma 15 *For any constant k, the circuit value problem for stratified monotone planar circuits restricted to input vectors containing only k maximal true intervals, can be solved in deterministic logarithmic space.*

Proof If the input is of this form, the configuration at each layer is made up of at most k intervals; knowing this, we can represents the configuration in logspace. Furthermore, it is easy, using only logspace, to compute the configuration of layer $l+1$ from the configuration of layer l. ☐

We now construct a **LOGDCFL** algorithm for evaluating stratified monotone planar circuits. The algorithm views the input x as a set of maximal true intervals, and considers them from left to right. At each end, for convenience, we add a "virtual" interval that never dies or merges. At any point in time the algorithm has a number of intervals on its pushdown store, ordered so that the leftmost ones are at the bottom of the pushdown store, and the rightmost ones are at the top. To the right of all intervals on the pushdown store, the algorithm maintains a "current" interval J. A number of input intervals have not been considered yet – these are all to the right of J. The intervals on the pushdown store, the current interval, and the unseen intervals together define an input y. The invariant of the algorithm is that the value of the output gate of the circuits is the same on input x as on input y.

The algorithm works as follows:

1. Push the first interval I onto the pushdown store, and let J be the next unseen interval.

2. Repeat the following:

   (a) If I and J are the two virtual intervals, then halt and output 0.

   (b) Evaluate the circuit on the input vector $\{I, J\}$ consisting of I and J only. Four cases could happen:

   Case 1, the circuit accepts $\{I, J\}$: halt and output 1.

   Case 2, I and J merge: set J to $I \vee J$, pop I off the pushdown store and set I to the new top.

   Case 3, J dies at an earlier layer than I: push J as the new I and let J be the next unseen interval.

   Case 4, otherwise: pop I off the pushdown store and set I to the new top. Continue with the same J.

First, let us convince ourselves that this is a **LOGDCFL** algorithm. The loop is executed a polynomial number of times, because Cases 2 and 4 each decrease the number of intervals by one, and so Cases 2 and 4 can occur only $O(n)$ times between two occurrences of Case 3. In each loop, the evaluation of C on $\{I, J\}$ can be done using logarithmic space by Lemma 15. So this algorithm uses deterministic polynomial time, logarithmic space, and a pushdown store.

Secondly, we should convince ourselves that the correct answer is computed. At any time there is a set of intervals under consideration, which defines an input to the circuit. This set is altered by merging or discarding as we go through each loop, and we shall justify that the output of the circuit remains the same.

Merging is correct: If I and J merge in the evaluation of $\{I, J\}$, so do they in the presence of other intervals. From Proposition 8, we can replace them by $I \vee J$ without changing the output.

Discarding an interval is correct: Let H be the interval on the pushdown store below I. Since H was pushed when considered with I, H dies after I. Now we want to discard I, so J grows as high as I. From Proposition 7, I has no chance of merging with others. From Proposition 9, I can be discarded without changing the output.

By induction, the output remains the same after all changes to the input. The algorithm always halts and outputs either 0 or 1. When it outputs 1, the output gate is covered by the 1-region of some interval, so the answer is correct. When it outputs 0, all real intervals have been considered, all mergings have occurred, but the output gate is not covered by any 1-region, so the answer is correct too.

This completes the proof of Theorem 2.

6 Representing and learning monotone planar functions

The **LOGDCFL** algorithm of the preceding section actually used very little information about the stratified circuit it evaluated. It needed to know, for any two input intervals I and J, what happens to the circuit if the input consists only of I and J — in particular, do the intervals merge? This information could be stored in a table of $O(n^4)$ bits, $O(1)$ bits for each of the $O(n^4)$ pairs of intervals. Note that *any* stratified monotone planar circuit, of *any* size, can be so represented. This *suggests* that any such circuit is equivalent to such a circuit of polynomial size. In this concluding section we sketch a proof of this fact, together with some consequences on learnability.

The interval-merging algorithm can be used for lower bounds: for example, the majority function on five inputs is *not* computable by a stratified monotone planar circuit. (This lower bound is not new: McColl [14] has characterized the threshold functions computable by monotone planar circuits.) If it were, consider the computation of this circuit on input 10101. If any of the three input intervals causes the output to be one without merging, then the circuit would also incorrectly accept one of the inputs 10000, 00100, or 00001. If, on the other hand, two of these intervals merged, the circuit would also be incorrect. For example, if the first two intervals merged the circuit would be unable to distinguish 10100, which it should reject, from 11100, which it should accept.

We can analyze circuit behavior more closely by determining whether the circuit allows specific *crossing events*. Consider the behavior of the circuit on an input in one of the regular languages 0^*1^* or 1^*0^*. There is a boundary between the region of the circuit set to one and that set to zero — the output value depends on whether this boundary leaves the circuit to the left or right of the output node. When the circuit acts on an input in $0^*1^*0^*$ or $1^*0^*1^*$, two of these boundary lines exist, and the behavior of the circuit depends on whether these boundary lines *cross*.

In the full paper we will show that if we know the output of the circuit on $O(n^3)$ inputs, specifically all those in $0^*1^*0^*1^*$ and $1^*0^*1^*0^*$, we can build another stratified monotone planar circuit, in a particular form, that computes the same function as the original circuit. Theorems 3 and

4 will follow directly from this construction, because the new circuit will have width $O(n)$ and depth $O(n^3)$, and the construction requires only polynomial time. In particular, two stratified monotone planar circuits that agree on these special inputs also agree on all other inputs.

Let f be the function computed by our original circuit. We will show that for every input w in $0^+1^+0^+$, $f(w) = 0$ iff f can be computed by a circuit in which a particular crossing event occurs. Similarly, if w is in $1^+0^+1^+$, then $f(w) = 1$ iff f can be computed by a circuit in which another crossing event occurs. So these values of f tell us *which* crossing events are to occur in the new circuit, but that is not enough to determine the function.

For every pair of crossing events that involve the *same boundary line*, we need to know which occurs *first*. We will show that the value of f on each input in $0^+1^+0^+1^+$ or $1^+0^+1^+0^+$ determines the order of one such pair of crossing events. Thus, the table of values for all such strings determines a partial order on all the crossing events on this circuit (there are $O(n^2)$ of them). We construct the circuit as follows: choose a total order on these events consistent with this partial order and use a circuit segment (of $O(n)$ width and $O(n)$ depth) to cause each crossing event to occur in this order. The total width is $O(n)$ and the total depth is $O(n^3)$. We conjecture that this construction can be improved to make the total depth $O(n^2)$.

The learning algorithm actually requires only nondeterminstic logspace and is thus in $\mathbf{NC}^2$, for example. Every part of the construction of the normal-form circuit can be done in deterministic logspace except for the topological sorting of the crossing events to respect all the known orders on pairs of crossing events.

7 Acknowledgements

The research of Peter Bro Miltersen and Sven Skyum was supported by BRICS (Basic Research in Computer Science), a Centre of the Danish National Research Foundation, and by the ESPRIT Long Term Research Programme of the EU under project number 20244 (ALCOM-IT). These sources also facilitated a visit to Aarhus by the other two authors, during which much of this research was carried out. Chi-Jen Lu's research at the University of Massachusetts was also supported in part by the National Science Foundation (U.S.A.) under grant CCR-9505446.

References

[1] D.A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. System Sci.* **38**:150–164, 1989.

[2] D. A. M. Barrington and N. Immerman. Time, hardware, and uniformity. In *Complexity Theory Retrospective II*, L. Hemaspaandra and A. Selman, editors, Springer-Verlag, 1–22, 1997.

[3] D. A. M. Barrington, N. Immerman and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.

[4] D. A. M. Barrington, C.-J. Lu, P. B. Miltersen, and S. Skyum. Searching constant width mazes captures the AC^0 hierarchy. *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science*, 73–83, 1998.

[5] D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of ACM*, 35(4):941–952, 1988.

[6] A.K. Chandra, S. Fortune and R. Lipton. Unbounded fan-in circuits and associative functions. *Proc. 19th ACM Symp. on Theory of Computing*, 123–131, 1987.

[7] S. Cook. Deterministic CFLs are accepted simultaneously in polynomial time and log space. *Proceedings of the 11th Annual ACM Symposium of Theory of Computing*, 338–345, 1979.

[8] A.L. Delcher and S.R. Kosaraju. An NC Algorithm for Evaluating Monotone Planar Circuits. *SIAM Journal on Computing* **24**: 369–375, 1995

[9] P. W. Dymond and S. Cook. Complexity theory of parallel time and hardware. *Information and Computation*, **80**:205–226, 1989.

[10] L.M. Goldschlager. A space efficient algorithm for the monotone planar circuit value problem. *Information Processing Letters* **10**:25–27, 1980.

[11] J.-W. Hong. *Computation: Computability, Similarity and Duality.* John Wiley & Sons, 1986.

[12] S. Istrail and D. Zivkovic. Bounded-width polynomial-size Boolean formulas compute exactly those functions in $\mathbf{AC}^0$. *Information Processing Letters* **50**:211–216, 1994.

[13] D.S. Johnson, A catalog of complexity classes. In: *Handbook of Theoretical Computer Science, Volume A.*, 67–161. Elsevier, Amsterdam, 1990.

[14] G. McColl. On the planar monotone computation of threshold functions. *Proceedings of the 2nd Annual Symposium on the Theoretical Aspects of Computer Science, LNCS* **182**:219–230, Springer Verlag, 1985.

[15] V. Ramachandran and H. Yang. An efficient parallel algorithm for the general planar monotone circuit value problem. *SIAM Journal on Computing* **25**:312–339, 1996.