Separating the Power of EREW and CREW PRAMs with Small Communication Width*

Paul Beame[1] Faith E. Fich[2] Rakesh K. Sinha[1]

[1] Computer Science and Engineering, University of Washington, Seattle, WA 98195, USA

[2] Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4

Abstract

We prove that evaluating a Boolean decision tree of height h requires $\Omega(\frac{h}{m+\log^* h})$ time on any EREW PRAM with communication width m and any number of processors. Since this function can be easily computed in time $O(\sqrt{h})$ on a CREW PRAM with communication width one using $2^{O(h)}$ processors, this gives a separation between the two models whenever the EREW PRAM has communication width $m \in o(\sqrt{h})$.

PROPOSED RUNNING HEAD: Separating EREW and CREW PRAMs

1 Introduction

Parallel random access machines (PRAMs) have been the model of choice for describing parallel algorithms and analyzing the parallel complexity of problems. Depending on whether or not we allow more than one processor to concurrently read from or write to a memory cell, we obtain different models of PRAMs and complexity classes associated with them. The three most popular models are the CRCW (concurrent read and write), CREW (concurrent read, but exclusive write), and EREW (exclusive read and write) PRAMs (see [JáJ92, Fic93].)

A basic issue in parallel complexity theory is to understand the relative power of different variants of PRAMs. Cook, Dwork, and Reischuk [CDR86], by an elegant argument, showed a separation between the powers of CRCW and CREW PRAMs. They proved that the OR of n bits, which can be easily computed in constant time on a CRCW PRAM, requires $\Omega(\log n)$ time on any CREW PRAM. This result was improved by Kutyłowski [Kut91] and Dietzfelbinger et al. [DKR90] who determined the exact complexity of OR.

Snir [Sni85] proved that the problem of searching a sorted list is more difficult on the EREW PRAM than on the CREW PRAM. Gafni, Naor, and Ragde [GNR89] extended

*Beame and Sinha's research supported by NSF/DARPA under grant CCR-8907960 and NSF under grant CCR-8858799. Fich's research supported by the Natural Science and Engineering Research Council of Canada and the Information Technology Research Centre of Ontario.

this result to a problem defined on a full domain. Because they use Ramsey theory, both the lower bounds rely crucially on the problems having an extremely large domain relative to the number of inputs (at least doubly exponential in the number of inputs). Essentially, they show that there is a large subset of the domain for which the state of the computation at each point depends only on the relative ordering of the input values.

Unfortunately, a lower bound proof that relies crucially on a problem's extremely large domain is not very satisfying. It may say more about the difficulty of handling very large numbers rather than the inherent difficulty of solving the problem on reasonable size domains. An open question that remains is whether or not there is a function $f : \{0,1\}^n \longrightarrow \{0,1\}$ that can be computed more quickly by a CREW PRAM than by an EREW PRAM.

Fich and Wigderson [FW90] have made some progress by resolving this question in a special case when there is a restriction imposed on where in shared memory processors can write. The EROW PRAM is an EREW PRAM in which each processor is said to "own" one shared memory cell and that is the only shared memory cell to which it is allowed to write. In this model, the choice of processor that may write into a given cell at a given point in time is independent of the input. Processors are still allowed to read from any shared memory cell. (The machine is *semi-oblivious* in the terminology of [CDR86].) The CROW PRAM [DR86] is the CREW PRAM restricted in the same manner. (Allowing processors to own more than one shared memory cell does not change the power of the CROW PRAM model.) Fich and Wigderson proved that the EROW PRAM requires $\Omega(\sqrt{\log n})$ time to compute a Boolean function that requires only $O(\log\log n)$ time on the CROW PRAM. The CROW PRAM never requires more than a constant factor more time than the CREW PRAM to compute any function defined on a complete domain (although the simulation may require a substantial increase in the number of processors) [Nis91]. However, the restriction to the owner write model with a single memory cell per processor seems much more drastic for exclusive read machines. A fast simulation of EREW PRAMs by EROW PRAMs seems unlikely. In fact, we do not see any obvious way to extend the lower bound proof of Fich and Wigderson to allow each processor to own an arbitrary number of memory cells.

This leaves open the following question: Is there a separation between CREW and EREW PRAMs for any function $f : \{0,1\}^n \rightarrow \{0,1\}$ without the owner-write restriction? We cannot answer this question in general but we can when the amount of shared memory through which processors can communicate is small. The *communication width* of a PRAM [VW85] is defined as the number of shared memory cells that are available for both reading and writing. (A separate read-only memory is used to store the input.) We denote by EREW(m), CREW(m), and CRCW(m) the respective PRAM models with communication width m. We show that a special case of the problem considered by Fich and Wigderson can be solved in time $O(\sqrt{\log n})$ by a CREW(1) PRAM, but requires $\Omega(\frac{\log n}{\log^* n})$ time on any EREW(1) PRAM. It is easy to see that the sequential time complexity of this problem is $\log_2 n$, which is almost matched by our lower bound for

the EREW(1) PRAM. For higher communication width we can prove a lower bound of $\Omega(\frac{\log n}{m+\log^* n})$ on any EREW(m) PRAM.

We would like to extend our separation between CREW and EREW PRAMs to larger communication width. Our hope is that some of the techniques developed for small communication widths will turn out to be useful even for the general case. For example, the technique in the lower bound result for the OR function on CREW(1) PRAMs [VW85, Bea86] is very similar to the technique that Kutyłowski [Kut91] eventually used in his optimal bound for the OR on general CREW PRAMs.

Our lower bound proof consists of three parts. First we show that any EREW(1) PRAM running for a short time can only have a small number of processors doing useful work. We then determine the time complexity of our problem on CREW(1) and CRCW(1) PRAMs with limited numbers of processors. This also implies an $\Omega(\log^{2/3} n)$ time bound for the EREW(1) PRAM. Finally, we show that there are subproblems (obtained via restrictions) on which the number of processors doing useful work is drastically reduced. Applying this result recursively, we obtain a nearly optimal EREW(m) PRAM lower bound for small m.

2 A Bound on the Number of Processors Doing Useful Work

For a function defined on n variables, we assume that a PRAM starts with its input stored in n read-only memory cells. We assume that it has m other shared read/write memory cells we call common cells. The output will be the contents of a designated common cell at the end of the computation. Computations proceed in steps. At every step, each processor may read a memory cell, perform some local computation, and then write into a common cell. We do not place any restrictions on the number of processors, word size, or the computational power of individual processors.

For any EREW or CREW PRAM computing a function f, we say that a processor p *writes by time* t if, on some input to f, p writes into some memory cell during the first t steps. Since the shared memory is the only means of communication, we can assume that for any PRAM running for t steps only the processors that write by time t are involved in the computation.

The bounds in [Bea86] (see also [VW85]) show that for any CREW(1) PRAM at time t for any given input vector e at least a $2^{-O(t^2)}$ fraction of all inputs vectors are indistinguishable from e from the point of view of any individual processor. Thus the number of processors that write by time t can easily be seen to be $2^{O(t^2)}$ since the machine only allows exclusive writes. (This bound is tight; see, for example, the algorithm given in the proof of Theorem 4.) For EREW(m) PRAMs, we now show considerably smaller bounds.

Lemma 1: Consider any EREW(m) PRAM computing a function with domain $\{0,1\}^n$. For all $t \geq 0$, at most $m(2^{t+1} + 2^t - 3) \leq m2^{t+2}$ processors write by time t.

Proof: For any processor P, time t, and input x, let $P^t(x)$ be the set of input variables that P reads during the first t steps on input x. Since a processor reads at most one cell per step, $|P^t(x)| \leq t$.

For $1 \leq j \leq t$, let $R(j)$ be the set of processors that do not read any common cell on any input for the first $j-1$ steps, but do read one of them on some input at step j.

Consider any processor $P \in R(j)$ and suppose that, on input $x \in \{0,1\}^n$, P reads some common cell at step j. At step j, processor P must decide whether to read that common cell based on the values of the variables in $P^{j-1}(x)$. The fraction of all inputs that agree with x on these variables is at least $2^{-(j-1)}$, since $|P^{j-1}(x)| \leq j-1$. On all these inputs, P reads the same common cell in step j. At most one processor can read that cell in step j on any particular input, so there are at most 2^{j-1} processors in $R(j)$ that do so. Since there are m such cells, $|R(j)| \leq m2^{j-1}$.

Similarly, if $W(j)$ is the set of processors that do not read any common cell on any input during the first j steps, but do write into one of them on some input at step j, then $|W(j)| \leq m2^j$. Thus the number of processors that write by time t is bounded above by

$$\sum_{j=1}^{t} |R(j)| + \sum_{j=1}^{t} |W(j)| \leq \sum_{j=1}^{t} m2^{j-1} + \sum_{j=1}^{t} m2^j \leq m(2^t - 1 + 2^{t+1} - 2). \quad \square$$

We will now construct an EREW(m) PRAM computing a function with domain $\{0,1\}^{m2^t}$ such that there are at least $m2^{t-2}$ processors that write by time t. Thus the bound in Lemma 1 is optimal, to within a small constant factor. We begin by considering the case $m = 1$.

Lemma 2: There is an EREW(1) PRAM computing a function of $\{0,1\}^{2^t}$ for which there are at least 2^{t-2} processors that write during step t.

Proof: Before beginning the construction, we examine the write operation of an EREW(1) PRAM. The selection of a processor to write during step t can be viewed as a competition between processors that is arbitrated by the input vector. A processor is a 'potential winner' if there is some setting of the input bits that would cause it to write during step t. Let $b_1 = 1$, $b_j = \sum_{i=1}^{j-1}[b_i + 1] < 2^j$, $k_1 = 1$, and $k_j = \sum_{i=1}^{j-1} k_i = 2^{j-2}$, for $j \geq 2$. For any j, we construct an EREW(1) PRAM algorithm for selecting a winning processor from among k_j potential winning processors. This algorithm runs for j steps, does not access the common cell, and is arbitrated by only b_j bits of input. Notice that this is enough to prove the lemma as we can modify the algorithm to make the winning processor write at step j.

The claim is proved by induction on j. The case $j = 1$ is trivial. For larger values of j, consider an input of length b_j which is partitioned into disjoint groups $X_1, X_2, \ldots, X_{j-1}$ of

length $b_1, b_2, \ldots, b_{j-1}$, respectively, as well as one extra group of $j-1$ bits: $y_1, y_2, \ldots, y_{j-1}$. Let $G_1, \ldots, G_{j-1}$ be disjoint sets containing $k_1, \ldots, k_{j-1}$ processors, respectively, for a total of k_j. By our induction hypothesis, for each $i < j$, we can select a winner from among G_i during the first i steps based on the input in X_i. The winner from G_i reads $y_1, y_2, \ldots, y_{j-i}$ in steps $i+1, i+2, \ldots, j$ respectively. This processor is a winner in step j if and only if $y_1 = y_2 = \cdots = y_{j-i-1} = 0$ and $y_{j-i} = 1$. It is easy to verify that read conflicts never occur. □

Corollary 3: There is an EREW(m) PRAM computing a function of $\{0,1\}^{m2^t}$ for which there are at least $m2^{t-2}$ processors that write during step t.

Proof: Run m separate copies of the algorithm in Lemma 1, one per common cell, on separate portions of the input. □

3 A CREW(1) Upper Bound for Evaluating Decision Trees

We now define a Boolean function and show that it can be computed in $O(\sqrt{\log n})$ time on a CREW(1) PRAM. In the next two sections, we will prove that this function requires significantly more time to be computed on an EREW(1) PRAM. Specifically, we interpret the $n = 2^h - 1$ input variables as the labels of the internal nodes in a complete Boolean decision tree D_h of height h, taken in some fixed (e.g. breadth-first) order. The leaves of D_h that are left children are labelled 0; those that are right children are labelled 1. Given an input, proceed down from the root, going left when a node labelled by a variable with value 0 is encountered and going right when a node labelled by a variable with value 1 is encountered. The value of the function $F_h : \{0,1\}^{2^h-1} \rightarrow \{0,1\}$ is the label of the leaf node that is reached.

There is a trivial sequential algorithm that computes F_h in h steps. It is unknown whether one can do better than this on an EREW(1) PRAM. However, the following lemma shows that F_h can be computed substantially faster on a CREW(1) PRAM.

Theorem 4: If $\binom{t}{2} \geq h$, then there is a CREW(1) PRAM that computes F_h in t steps.

Proof: For each of the 2^h root-leaf paths in the decision tree D_h, we assign a group of $t-1$ processors. Exactly one of these root-leaf paths is the correct path. In the following algorithm, the common cell will contain the values of the labels of the first $\binom{j+1}{2}$ nodes on the correct root-leaf path at the end of step $j+1$.

The jth processor in each group is active for the first $j+1$ steps. During the first j steps, it reads the j variables labelling nodes $\binom{j}{2}+1$ through $\binom{j+1}{2}$ on its root-leaf path. At step $j+1$, it reads the common cell, which contains the values of the labels of the first $\binom{j}{2}$ nodes on the correct root-leaf path. (When $j = 1$, the common cell contains no

information.) At this point, the jth processor in each group knows whether or not its root-leaf path agrees with the correct root-leaf path at the first $\binom{j+1}{2}$ nodes. Among the processors whose paths agree, a prespecified one (e.g. the jth processor in the leftmost of these groups) appends the bits that it has read to the previous contents of the common cell. Thus, at the end of step $j+1$, the common cell contains the values of the labels of the first $\binom{j+1}{2}$ nodes along the correct root-leaf path.

To compute F_h, we modify the algorithm slightly so that at the last step, instead of appending bits to the common cell, a processor writes down the value of the leaf determined by the h internal nodes on its path, i.e. the leaf node in D_h that is reached. □

4 Lower Bounds for Processor-limited PRAMs

In this section, we show that to compute F_h quickly we need to have many processors doing useful work even on a CRCW(1) PRAM. Using our bounds from Section 2 on the number of processors doing useful work, this will give an $\Omega(h^{2/3})$ lower bound for the EREW(1) PRAM. In the next section, we will improve this bound to a nearly optimal $\Omega(\frac{h}{\log^* h})$ by combining the techniques of this section with a new restriction technique.

A *restriction* is a partial function that sets the values of some input variables. For any restriction r that sets input variables to 0 or 1, let $r(F_h)$ be the function F_h with restriction r applied to it. Define the *depth* of r, $d(r)$, to be the minimum depth of any node v in D_h, the underlying decision tree of F_h, such that the path from the root to v is consistent with r and the subtree rooted at v does not contain any variables set by r. Note that $F_{h-d(r)}$ is a subfunction of $r(F_h)$.

Define the *history* of the common cells on any input to be the sequence of vectors of values that they take on that input. Our lower bound proofs proceed by fixing the history of the common cells. We use the following result of Vishkin and Wigderson [VW85].

Lemma 5: [VW85] For any CRCW(m) PRAM running for t steps, there is a restriction r which sets at most $m\binom{t+1}{2}$ variables such that the history of the common cells for the first t steps is the same for all inputs consistent with r.

Lemma 6: For any integer $h \geq m\binom{t+1}{2}$ and any CRCW(m) PRAM running for t steps, there is a restriction r' of depth at most $m\binom{t+1}{2}$ such that the history of the common cells for the first t steps is the same for all inputs to F_h consistent with r'.

Proof: By Lemma 5, there is a restriction r which sets at most $m\binom{t+1}{2}$ variables such that the history of the common cells for the first t steps is the same for all inputs consistent with r.

We define a restriction r' that is consistent with r by tracing a path of length at most $m\binom{t+1}{2}$ from the root of F_h one node at a time, as follows. If r sets the variable at the current node then let r' set this variable to be consistent with r and take the branch corresponding to this value. Otherwise, consider the number of variables that are set by r in each subtree and take the branch which leads to the subtree with the smaller number of these variables. Each time we extend the path by one node there is at least one less variable set by r in the subtree reached by the path. So, by the time the path reaches length $m\binom{t+1}{2}$, we will have reached the root of a subtree with none of its variables set. We set all variables outside this subtree in some manner consistent with r. Clearly, the resulting restriction r' has depth at most $m\binom{t+1}{2}$ and, for all inputs consistent with r', the common cells have the same history for the first t steps. □

In particular, this implies that the CREW(1) PRAM algorithm to compute F_h given in the proof of Theorem 4 is within one step of optimal.

Theorem 7: If a CRCW(1) PRAM computes F_h in t steps, then $\binom{t+1}{2} \geq h$.

Proof: Consider any CRCW(1) PRAM that computes F_h in t steps. Suppose, to the contrary, that $\binom{t+1}{2} \leq h-1$. Then, by Lemma 6, there is a restriction r' of depth at most $\binom{t+1}{2}$ such that the history of the common cell for the entire computation is the same for all inputs to F_h consistent with r'. In particular, the answer produced by the computation is the same for all these inputs. However, since $d(r') \leq h-1$, $r'(F_h)$ is a not a constant function. This contradicts the assumption that the CRCW(1) PRAM computes F_h in t steps. □

The following theorem shows that, with a limited number of processors, a larger lower bound may be obtained.

Theorem 8: Any CRCW(1) PRAM with p processors that computes F_h requires at least $\frac{2h}{3\lceil 1+\sqrt{\log p}\rceil}$ steps.

Proof: If $p = 1$, an easy adversary argument shows that computing F_h has complexity $h+1$. Therefore, assume $p \geq 2$. Let $t = \lceil\sqrt{\log p}\rceil \geq 1$. The proof proceeds by induction on h.

Suppose there is a CRCW(1) PRAM with p processors that computes F_h in T steps. Then, from Theorem 7, $\binom{T+1}{2} \geq h$.

If $\binom{t+1}{2} \geq h/3$ (which is always true when $h = 0$), then $T(t+1)/2 \geq h/3$ so $T \geq 2h/3(t+1)$ as required. Therefore, assume that $\binom{t+1}{2} < h/3$.

By Lemma 6, there is a restriction r' of depth at most $\binom{t+1}{2}$ that fixes the history of the common cell for the first t steps. Consider the computations of the CRCW(1) PRAM on all inputs consistent with r'. Since each input variable has at most two different values

and the value of the common cell is fixed at each time step, it follows that each processor is in one of at most 2^i states at the end of step $i < t$. Now the state of a processor determines which memory cell it will read next. Thus at most $p \sum_{i=0}^{t-1} 2^i < 2^{t^2+t}$ different input variables are read during the first t steps by all processors on all these inputs.

Let v be any node of depth $d(r')$ such that the path from the root to v is consistent with r' and the subtree rooted at v does not contain any variables set by r'. Consider the 2^{t^2+t} nodes at distance $t(t+1)$ from v. There is at least one node w such that the subtree rooted at w contains input variables that no processor can possibly read in the first t steps. Let r'' be a restriction that extends r' by setting the variables labelling all ancestors of w so as to cause the path from the root of D_h to w to be followed. Only the variables labelling nodes in the subtree rooted at w are left unset. All remaining variables are set arbitrarily. The restriction r'' has depth at most $\binom{t+1}{2} + t(t+1) = 3\binom{t+1}{2} < h$. By construction, the functions $F_{h-d(r'')}$ and $r''(F_h)$ are identical, up to the renaming of variables. Since no processors have read any input variables of this subfunction at time t, it follows from the induction hypothesis that at least $\left[2h - 6\binom{t+1}{2}\right]/3(t+1)$ additional steps are required to compute this subfunction. Therefore $T \geq t + \left[2h - 6\binom{t+1}{2}\right]/3(t+1) = \frac{2h}{3(t+1)}$. □

We note that this lower bound is asymptotically optimal even for a CREW(1) PRAM.

Corollary 9: For all integers $1 \leq p \leq 2^h$, the complexity of F_h on a CRCW(1) or CREW(1) PRAM with p processors is $\Theta(h/\sqrt{\log p})$.

Proof: The lower bound follows from Theorem 8. For the upper bound, notice that the algorithm in Theorem 4 shows that, with p processors, a CREW(1) PRAM can evaluate a decision tree of height $\Theta(\log p)$ in time $O(\sqrt{\log p})$. To compute F_h with p processors on a CREW(1) PRAM, we simply apply this algorithm sequentially $O(h/\log p)$ times to obtain a running time of $O(h/\sqrt{\log p})$. □

Using the bounds of Section 2 we have:

Corollary 10: Any EREW(1) PRAM computing F_h must run for at least $\frac{1}{3}h^{\frac{2}{3}}$ steps.

Proof: Suppose the EREW(1) PRAM runs for T steps. Then, by Lemma 1, we can assume that it has at most $p = 2^{T+2}$ processors. Now, applying Theorem 8, it follows that $T \geq \frac{2h}{3\lceil 1+\sqrt{T+2}\rceil}$. Since $\lceil 1 + \sqrt{T+2}\rceil \leq 3\sqrt{T}$ for all integers $T > 0$, solving we obtain $T \geq \frac{1}{3}h^{2/3}$, as required. □

5 A Near Optimal EREW(m) Lower Bound for Small m

We strengthen the arguments of the previous section to prove a nearly optimal $\Omega(\frac{h}{\log^* h})$ lower bound on the time for an EREW(1) PRAM to compute F_h and, more generally, to

obtain an $\Omega(h/(m + \log^* h))$ lower bound for an EREW(m) PRAM computing F_h. The key to the improvement is a new lemma that replaces Lemma 1 in the argument which shows that we can select a large subset of inputs on which very few processors ever do useful work. We use this to obtain a stronger version of Lemma 6 for the EREW(m) PRAM. This involves recursively applying the argument of the previous section to obtain a better lower bound.

Lemma 11: For any integers T, $h \geq 2T + 2 + \lceil \log m \rceil$, and any EREW($m$) PRAM computing F_h, there is a restriction r of depth $2T + 2 + \lceil \log m \rceil$ such that at most $2mT$ processors write by time T on inputs consistent with r.

Proof: From Lemma 1, we can assume that the EREW(m) PRAM has at most $m2^{T+2}$ processors. For each processor P, let $s(P)$ denote the set of input variables that P reads during the first T steps of computation on any input to F_h, but before it reads the common cell. Define $S = \bigcup_P s(P)$. As in the proof of Theorem 8, since each input variable has at most two different values, $|s(P)| < 2^T$ and thus $|S| < m2^{2T+2}$.

Consider the $m2^{2T+2}$ nodes at depth $2T+2+\lceil \log m \rceil$ from the root of D_h. For at least one such node v, none of the nodes in the subtree rooted at v is labelled by variables in S. Set the variables labelling ancestors of v so that the path from the root to v is followed in D_h. All variables labelling nodes in the subtree rooted at v are left unset. Set all other variables outside this subtree arbitrarily. Let r be the resulting restriction.

From now on, consider only those inputs consistent with r. The subtree rooted at v does not contain any input variables from S, so no processor reads any unset variable until after it has read one of the common cells. Since the PRAM is exclusive read, for each step $j \leq T$, for each common cell there is at most one processor that does not read any common cell on any input for the first $j - 1$ steps but does read that cell on some input at step j. Hence, at most mT processors read some common cell in the first T steps. Similarly, at most mT processors from among those that have not read any common cell may write in the first T steps. So, altogether, there are at most $2mT$ processors that can write into some common cell on inputs consistent with r. □

In order to describe the behavior of our recursive construction it is convenient to introduce a simple notation for the bounds that we obtain. Let $A_m(2) = 3m$ and, for $T \geq 3$, let

$$A_m(T) = (2T + 2 + \lceil \log m \rceil) + \left\lceil \frac{T}{\lceil \log T \rceil} \right\rceil (A_m(\lceil \log T \rceil) + 2\lceil \log T \rceil + 1 + \lceil \log m \rceil).$$

It can be easily verified by induction that $A_m(T) \in O(Tm + T \log^* T)$.

Lemma 12: For any integer T, and $h \geq A_m(T)$, and any EREW(m) PRAM computing F_h, there is a restriction r of depth at most $A_m(T)$ such that the history of the common cells for the first T steps is the same for all inputs to F_h consistent with r.

Proof: The proof is by induction on the value of T.

For $T = 2$, $A_m(T) = 3m = m\binom{T+1}{2}$ and the claim follows from Lemma 6.

For larger values of T, first use Lemma 11 to obtain a restriction r_0 of depth $2T + 2 + \lceil \log m \rceil$ such that at most $2mT$ processors write by time T on inputs consistent with r_0. Let $\ell = \lceil \log T \rceil$. Break the T steps of the computation into $\lceil T/\ell \rceil$ subintervals each of length at most ℓ. It is sufficient to prove that, for $i \leq \ell$, there is an extension r_i of r_0 of total depth at most $2T + 2 + \lceil \log m \rceil + i(A_m(\ell) + 2\ell + 1 + \lceil \log m \rceil)$ so that none of these $2mT$ processors read any variables left unset by r_i during the first i subintervals on all inputs to F_h consistent with r_i. This is proved by induction on i.

The case $i = 0$ is trivial, so suppose $i \geq 1$. Assume that a restriction r_{i-1} with the desired properties exists. Without loss of generality, we may also assume that r_{i-1} sets all variables labelling nodes outside of some tree of height $h - d(r_{i-1})$. Then $r_{i-1}(F_h)$ is the same as $F_{h-d(r_{i-1})}$ up to the renaming of variables. Since $\ell < T$, it follows from the induction hypothesis that there is a restriction of depth $A_m(\ell)$ such that the history of the common cells for the first ℓ steps is the same for all inputs to $F_{h-d(r_{i-1})}$ consistent with this restriction. None of the $2mT$ processors have read any variables left unset by r_{i-1} during the first $i - 1$ subintervals on inputs to F_h consistent with r_{i-1}. Therefore there is also a restriction r'_i that extends r_{i-1} with depth $d(r_{i-1}) + A_m(\ell)$, such that the history of the common cells for the first i subintervals is the same for all inputs to F_h consistent with r'_i.

As in the proof of Theorem 8, each of the $2mT$ processors can read at most $2^\ell - 1$ input variables during the ith subinterval. Thus there is an extension r_i of r'_i with depth $d(r'_i) + \lceil \log(2mT) \rceil + \ell \leq 2T + 2 + i(A_m(\ell) + 2\ell + 1 + \lceil \log m \rceil)$ so that none of the processors have read any inputs of F_h left unset by r_i during the first i subintervals. This is what was required. □

We can use this lemma to derive the near optimal lower bound for the EREW(m) PRAM for small m.

Theorem 13: Any EREW(m) PRAM computing F_h must run for $\Omega(\frac{h}{m + \log^* h})$ steps.

Proof: Suppose there is an EREW(m) PRAM computing F_h that runs for T steps, where $A_m(T) \leq h - 1$. By Lemma 12, there is a restriction r of depth $A_m(T)$ such that the answer in the common memory cells is the same for all inputs to $r(F_h)$. However, there are two inputs in $r(F_h)$ that reach the same leaf, but differ in the value of that leaf. These inputs should have different answers. Thus $A_m(T) \geq h$. Since $A_m(T) \in O(Tm + T \log^* T)$, it follows that $T \in \Omega(\frac{h}{m + \log^* h})$. □

Theorem 13 and Theorem 4 immediately give our main separation theorem.

Theorem 14: There is a function on n Boolean variables that can be computed in $O(\sqrt{\log n})$ time on a CREW(1) PRAM but requires $\Omega(\frac{\log n}{\log^* n})$ time on every EREW(1) PRAM.

We also get a separation between CREW(1) and EREW(m) PRAMs when m is small.

Theorem 15: For all $m \in o(\sqrt{\log n})$, there is a function on n Boolean variables that can be solved asymptotically faster on a CREW(1) PRAM than on any EREW(m) PRAM.

References

[Bea86] P. Beame. *Lower Bounds in Parallel Machine Computation.* PhD thesis, Department of Computer Science, University of Toronto, 1986. Also appears as Technical Report TR 198/87.

[CDR86] Steven A. Cook, Cynthia Dwork, and Rudiger Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM Journal on Computing*, 15(1):87–97, February 1986.

[DKR90] M. Dietzfelbinger, M. Kutyłowski, and R. Reischuk. Exact time bounds for computing Boolean functions on PRAMs without simultaneous writes. In *Proceedings of the 1990 ACM Symposium on Parallel Algorithms and Architectures*, pages 125–135, Crete, Greece, June 1990.

[DR86] P. W. Dymond and W. L. Ruzzo. Parallel random access machines with owned global memory and deterministic context-free language recognition. In Laurent Kott, editor, *Automata, Languages, and Programming: 13th International Colloquium*, volume 226 of *Lecture Notes in Computer Science*, pages 95–104, Rennes, France, July 1986. Springer-Verlag.

[Fic93] Faith E. Fich. The complexity of computation on the parallel random access machine. In John H. Reif, editor, *Synthesis of Parallel Algorithms*, pages 843–900. Morgan Kaufman, San Mateo, CA, 1993.

[FW90] Faith E. Fich and Avi Wigderson. Towards understanding exclusive read. *SIAM Journal on Computing*, 19(4):717–727, 1990.

[GNR89] E. Gafni, J. Naor, and P. Ragde. On separating the EREW and CREW PRAM models. *Theoretical Computer Science*, 68(3):343–346, 1989.

[JáJ92] Joseph JáJá. *An Introduction to Parallel Algorithms.* Addison-Wesley, 1992.

[Kut91] M. Kutyłoswski. The complexity of Boolean functions on CREW PRAMs. *SIAM Journal on Computing*, 20(5):824–833, 1991.

[Nis91] Noam Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, December 1991.

[Sni85] M. Snir. On parallel searching. *SIAM Journal on Computing*, 14(3):688–708, 1985.

[VW85] U. Vishkin and A. Wigderson. Trade-offs between depth and width in parallel computation. *SIAM Journal on Computing*, 14(2):303–314, May 1985.