SIAM J. COMPUT.
Vol. 19, No. 4, pp. 718–727, August 1990
© 1990 Society for Industrial and Applied Mathematics
011

TOWARD UNDERSTANDING EXCLUSIVE READ*

FAITH E. FICH† AND AVI WIGDERSON‡

Abstract. The ability of many processors to simultaneously read from the same cell of shared memory can give additional power to a parallel random access machine. In this paper, a natural Boolean function of n variables is described, and it is shown that the expected running time of any probabilistic EROW PRAM computing this function is in $\Omega(\sqrt{\log n})$, although it can be computed by a CROW PRAM in $O(\log \log n)$ steps.

Key words. shared memory parallel computation, PRAM, lower bounds, exclusive read, decision trees

AMS(MOS) subject classifications. 68Q05, 68Q10

1. Introduction. In [8], Snir proved that the following range search problem has time complexity $\Theta(\sqrt{\log n})$ on an EREW PRAM:

> Given distinct inputs $x_1, \cdots, x_n$, and y, with $x_1 < \cdots < x_n$, determine the maximum index i such that $x_i < y$.

This problem can be solved in a constant number of steps on a CREW PRAM. Thus, in certain situations, CREW PRAMs are more powerful than EREW PRAMs.

But this result does not tell us everything we would like to know. For example, consider the relationship between the CROW and CREW PRAMs. The OR of n Boolean values, at most one of which is 1, can be determined in a constant number of steps on a CREW PRAM, but $\log_2 n$ steps are required on a CROW PRAM [2]. In contrast, Nisan [7] proved that any Boolean function $f: \{0,1\}^n \to \{0,1\}$ has, to within a small constant factor, the same time complexity on CREW and CROW PRAMs.

Two features of Snir's result are important in this regard. The first is that, like the restricted version of OR in the previous paragraph, the domain of his range search problem is not complete. (A complete domain is one of the form D^n for some set D.) Such a situation can be viewed as having information about the inputs built into the program. This information can be used by the algorithm to ensure that no conflict arises during a potentially concurrent read or write. In particular, it enables the range search problem to be solved quickly on a CREW PRAM. Gafni, Naor, and Ragde [5] recently improved Snir's result by exhibiting a function with a complete domain that is easy to solve on a CROW PRAM and still difficult to solve on an EREW PRAM.

Another feature of these results is that the proofs of the lower bounds depend on the domains of the functions being very large. The essential idea is to show that, using Ramsey theory, there is a large subset of the domain for which the states of all processors and the contents of all shared memory cells at each point in the computation depend only on the relative order of the input values, not on their values.

* Received by the editors September 17, 1989; accepted for publication (in revised form) November 30, 1989. This work was partially supported by the Information Technology Research Centre of Ontario, Ontario, Canada.

† University of Toronto, Department of Computer Science, Toronto, Ontario, Canada M5S 1A4. The research of this author was partially supported by Natural Sciences and Engineering Research Council of Canada grant A9176 and a grant from the University of Toronto, Toronto, Ontario, Canada.

‡ Hebrew University of Jerusalem, Computer Science Department, Jerusalem 91904, Israel. The research of this author was partially supported by an Alon Fellowship and the American-Israeli Binational Science Foundation.

It remains open whet
EREW PRAM as by a CR
by defining a natural Bo
PRAM and we prove that
if the algorithm is allowed
is used to obtain this lowe
solve on an EREW PRA
attempting to extend the

2. Models. In this
machines (PRAMs) with
that can contain arbitrarily
n cells of shared memory
cell at the end of the co
solve a problem. At each
then perform an arbitrary
of shared memory.

In the concurrent rea
not write to the same me
number of processors may
exclusive write (EREW) P
cell for either reading or

Complete networks o
tion. Again we assume th
synchronously. At each st
of its choice, performs a
new message. The input to
and, at the end of the co

This model is equiva
there is a one-to-one cor
and only the processor c
Many algorithms designed
and Ruzzo, who introduce
(CROW) PRAM.

If we further restrict t
can read from each share
(EROW) PRAM. This is
PRAM corresponds to a c
by at most one processor
which processor, if any, i

In many respects, th
machine. However, this de
On the other hand, the al
implemented on less pow

For our lower bound
PRAM processor *knowin*
inductively for each step
knows the input bit it has
No processor knows any
after it reads the message

It remains open whether all Boolean functions can be computed as quickly by an EREW PRAM as by a CREW PRAM. We make progress toward solving this problem by defining a natural Boolean function that can be computed quickly on a CROW PRAM and we prove that it requires a long time to solve on an EROW PRAM, even if the algorithm is allowed to make probabilistic choices. A new probabilistic technique is used to obtain this lower bound. We also give evidence that this function is hard to solve on an EREW PRAM. Finally, we explain where the difficulties arise when attempting to extend the lower bound to the EREW PRAM.

2. Models. In this paper, we consider nonuniform parallel random access machines (PRAMs) with an infinite number of processors and shared memory cells that can contain arbitrarily large values. The n input values initially appear in the first n cells of shared memory and the answer is the contents of the first shared memory cell at the end of the computation. The processors work together synchronously to solve a problem. At each step, a processor may read from one cell of shared memory, then perform an arbitrary amount of local computation, and finally write to one cell of shared memory.

In the concurrent read, exclusive write (CREW) PRAM, multiple processors may not write to the same memory cell at the same step of a computation, although any number of processors may simultaneously read from a single cell. The exclusive read, exclusive write (EREW) PRAM does not allow simultaneous access to a shared memory cell for either reading or writing.

Complete networks of processors are also interesting models of parallel computation. Again we assume that there is an infinite number of processors and they work synchronously. At each step, a processor reads the message posted by one processor of its choice, performs an arbitrary amount of local computation, and then posts a new message. The input to a problem is initially distributed among the first n processors and, at the end of the computation, the first processor has determined the answer.

This model is equivalent to a restricted version of the CREW PRAM in which there is a one-to-one correspondence between processors and shared memory cells and only the processor corresponding to a particular memory cell may write to it. Many algorithms designed for CREW PRAMs avoid write conflicts in this way. Dymond and Ruzzo, who introduced this model in [3], call it the concurrent read, owner write (CROW) PRAM.

If we further restrict the CROW PRAM so that, at each step, at most one processor can read from each shared memory cell, we obtain the exclusive read, owner write (EROW) PRAM. This is the model for which we prove a lower bound. The EROW PRAM corresponds to a complete network in which each posted message can be read by at most one processor at a time. (Note that a processor is not required to know which processor, if any, is reading its message at a given step.)

In many respects, these models are more powerful than any realistic parallel machine. However, this does not affect the significance of the lower bounds we obtain. On the other hand, the algorithms presented in this paper are quite simple and easily implemented on less powerful models.

For our lower bound proof, it is necessary to introduce the concept of a CROW PRAM processor *knowing* certain input bits. The processors' knowledge is defined inductively for each step of the computation. Initially, each of the first n processors knows the input bit it has been given (i.e., the ith processor knows the ith input bit). No processor knows any other bit of the input. The input bits known by a processor after it reads the message posted by another processor are the union of the bits known

by the two processors before the read occurred. Changing the value of any input bits that a processor does not know at any given point in time cannot change the state of the processor at that time.

The following two lemmas describe properties of knowledge that are important for our lower bound proof.

LEMMA 1 (Cook, Dwork, and Reischuk [2]). *For a* CROW PRAM, *on any input, every processor knows at most* 2^t *input bits immediately after step t.*

LEMMA 2 (Beame [1]). *For an* EROW PRAM, *on any input, each input bit is known by at most* 2^t *processors immediately after step t.*

3. The Boolean decision-tree evaluation problem. Suppose we are given a decision tree, each node of which is labeled by a Boolean variable (called a *query*). Suppose we are also given an outcome for each query. Consider the path that starts at the root, goes left whenever the query at the current node has outcome 0 and goes right whenever the query has outcome 1. The decision-tree evaluation problem is to determine the outcome of the query labeling the leaf reached by this path.

For example, given the decision tree in Fig. 1 and the outcomes $x_0 = 1$, $x_1 = 1$, $x_2 = 1$, and $x_3 = 0$, the second leaf from the left is reached and, hence, the decision tree has value 1.

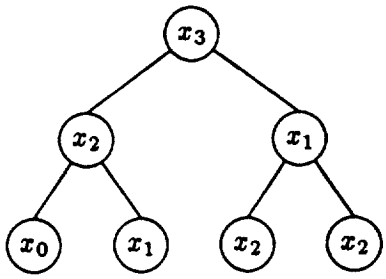

FIG. 1. *A decision tree.*

Let $D_{m,h}: \{0,1\}^{m(2^{h+1}-1)+2^m} \to \{0,1\}$ be a Boolean function representing a Boolean decision-tree evaluation problem for a complete binary tree of height h in which every node is labeled by one of 2^m queries. This can be done by dividing the first $m(2^{h+1}-1)$ input bits into $2^{h+1}-1$ blocks of length m. The value y_i of the ith block denotes the index of the query labeling the ith node in the tree. The last 2^m bits, $x_0, \cdots, x_{2^m-1}$, denote the outcomes of the queries. For example, $D_{2,2}(111001000110101110) = 1$. (See Fig. 1.)

Clearly, the Boolean function $D_{m,h}$ can be computed by a (sequential) random access machine in $(m+1)h$ steps, following the path through the tree, alternately reading the label of the next node and then the outcome of the query labeling it.

Using an EROW PRAM, an $O(\log m + h)$ upper bound can be obtained by first collecting the m bits comprising the label of each node into one memory cell, using $O(m/\log m)$ processors and $O(\log m)$ time. This is done in parallel for each node in the tree. Then the path through the tree can be followed in $O(h)$ more steps.

This can be improved to $O(\log m + \log h)$ on a CROW PRAM. After the bits of the node labels have been collected, as above, one processor is assigned to each node of the decision tree. In one step, each processor reads the outcome of the query labeling its node. Because many nodes in the decision tree can have the same label, concurrent

read is essential for this
pointer from the node to
path from the root to a
EROW PRAM.

Another (although les
for the function computed
the query outcomes. For
of $2^{h+1}-1$ processors is al
of the decision tree. Using
in each group determine
outcomes associated with
at this point in the algori
from the actual outcomes
reading from the appropri
is $O(m+\log h)$.

Only the first step of
the 2^{2^m} copies of the deci
an $O(2^m+\log h)$ upper b

4. The lower bound.

THEOREM 3. *The ex*
PRAM *to solve the Boolea*
more than $T/2$.

Proof. Let $X_0, \cdots, X$
and uniformly chosen fro
$(X_0, \cdots, X_{2^m-1})$ is used t
be random variables with
is used to denote a labeli
(i.e., the value of the ran
uniform distribution. Onc
node is chosen uniformly
gives us a uniformly chos
nodes along any path fron
to show [9] that the avera
performed by any determi

Imagine the decision tr
For $t = 0, \cdots, T$, the node
X in the decision tree lab
$R_t(Y, X)$ is denoted by S
$R_0(Y, X)$ is its root, and I
2. Finally, let $U_t(Y, X)$ de
of $R_t(Y, X)$. Then $U_0(Y,$
$U_T(Y, X)$, and $|U_t(Y, X)$

CLAIM. *The probabil*
both the outcome of a que
subtree $S_t(Y, X)$ *is at mos*

In particular, from th
(which is supposed to det
$U_T(Y, X)$ and (a bit of) t
probability that processor

read is essential for this step. The outcome of the query labeling a node defines a pointer from the node to one of its two children. Using pointer jumper [6], the unique path from the root to a leaf can then be determined in $O(\log h)$ steps, even by an EROW PRAM.

Another (although less efficient) CROW PRAM algorithm creates a table of values for the function computed by the decision tree and then performs table look up using the query outcomes. For each of the 2^{2^m} possible outcomes for the queries, a group of $2^{h+1}-1$ processors is allocated. In one step, each group of processors makes a copy of the decision tree. Using pointer jumping, as in the previous algorithm, the processors in each group determine the answer that would be obtained assuming the query outcomes associated with their group. (The actual query outcomes have not been read at this point in the algorithm.) In $O(m)$ steps, the correct group can be determined from the actual outcomes of the 2^m queries. Then the answer can be determined by reading from the appropriate place in the table. The total time taken by this algorithm is $O(m+\log h)$.

Only the first step of this algorithm uses concurrent read. With exclusive read, the 2^{2^m} copies of the decision tree can be constructed in 2^m steps. This gives rise to an $O(2^m+\log h)$ upper bound on the EROW PRAM.

4. The lower bound.

THEOREM 3. *The expected number of steps performed by a probabilistic* EROW PRAM *to solve the Boolean decision tree evaluation problem for* $m=3T$ *and* $h=6T^2$ *is more than* $T/2$.

Proof. Let $X_0, \cdots, X_{2^m-1}$ be random variables whose values are independently and uniformly chosen from the range $\{0, 1\}$. The sequence of random variables $X=(X_0, \cdots, X_{2^m-1})$ is used to denote the outcomes of the queries. Let $Y_1, \cdots, Y_{2^{h+1}-1}$ be random variables with range $\{0, \cdots, 2^m-1\}$. The sequence $Y=(Y_1, \cdots, Y_{2^{h+1}-1})$ is used to denote a labeling of the nodes in the decision tree. The label of the root (i.e., the value of the random variable corresponding to the root) is chosen using a uniform distribution. Once all ancestors of a node have been labeled, the label of the node is chosen uniformly among those queries not labeling any of its ancestors. This gives us a uniformly chosen labeling of the decision tree with the property that all nodes along any path from the root to a leaf are labeled by different queries. It suffices to show [9] that the average number of steps (with respect to this input distribution) performed by any deterministic EROW PRAM solving this problem is more than $T/2$.

Imagine the decision tree sliced horizontally into T pieces, each of height $k=6T$. For $t=0, \cdots, T$, the node at depth kt on the path determined by the query outcomes X in the decision tree labeled by Y is denoted by $R_t(Y, X)$ and the subtree rooted at $R_t(Y, X)$ is denoted by $S_t(Y, X)$. In particular, $S_0(Y, X)$ is the entire decision tree, $R_0(Y, X)$ is its root, and $R_T(Y, X)$ is the leaf that is reached. This is illustrated in Fig. 2. Finally, let $U_t(Y, X)$ denote the set of queries that do not label any proper ancestor of $R_t(Y, X)$. Then $U_0(Y, X)$ is the set of all 2^m queries, $U_0(Y, X) \supseteq U_1(Y, X) \supseteq \cdots \supseteq U_T(Y, X)$, and $|U_t(Y, X)| = 2^m - kt$.

CLAIM. *The probability that there is a processor that, at the end of step* t, *knows both the outcome of a query in* $U_t(Y, X)$ *and (any bit of) the label of a node in the subtree* $S_t(Y, X)$ *is at most* $t2^{12-T}$.

In particular, from the claim, the probability is at most $T2^{12-T}$ that processor P_1 (which is supposed to determine the answer) knows both the outcome of a query in $U_T(Y, X)$ and (a bit of) the label of $R_T(Y, X)$ within T steps. We will show that the probability that processor P_1 has the correct answer is only slightly more than $\frac{1}{2}$.

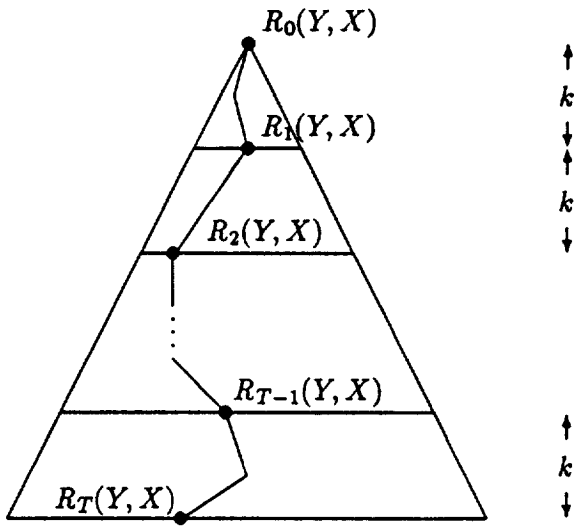

FIG. 2. *A decision tree sliced into pieces.*

Let C denote the event that processor P_1 has determined the correct answer within T steps, let B denote the event that P_1 knows the label of $R_T(Y, X)$ within T steps, and let A denote the event that the label of $R_T(Y, X)$ is a query whose outcome is known by P_1 within T steps. If the label of $R_T(Y, X)$ is a query whose outcome is not known by P_1, then changing only the outcome of this query does not change P_1's state; although, for the algorithm to be correct, it should. Since the outcome of the query labeling $R_T(Y, X)$ is equally likely to be 0 or 1, it follows that

$$\text{pr}[C|\bar{A}] \leqq \tfrac{1}{2}.$$

If P_1 does not know the label of $R_T(Y, X)$, changing the label of $R_T(Y, X)$ to anything else in $U_T(Y, X)$ does not change P_1's state. Because the label of $R_T(Y, X)$ is equally likely to be any query in $U_T(Y, X)$ and, by Lemma 1, P_1 knows the outcomes of at most 2^T queries,

$$\text{pr}[A|\bar{B}] \leqq \frac{2^T}{|U_T(Y, X)|} = \frac{2^T}{2^{3T} - 6T^2} \leqq 2^{2-2T}.$$

Thus

$$\begin{aligned}\text{pr}[C] &= \text{pr}[C|A \wedge B] \cdot \text{pr}[A \wedge B] + \text{pr}[C|\bar{A}] \cdot \text{pr}[\bar{A}] + \text{pr}[C|A \wedge \bar{B}] \cdot \text{pr}[A \wedge \bar{B}] \\ &\leqq 1 \cdot \text{pr}[A \wedge B] + \text{pr}[C|\bar{A}] \cdot 1 + 1 \cdot \text{pr}[A|\bar{B}] \\ &\leqq \tfrac{1}{2} + T2^{12-T} + 2^{2-2T}.\end{aligned}$$

A correct algorithm always performs at least one step. If the correct answer has not been determined within T steps, the algorithm must perform at least one more step. Therefore, the expected number of steps performed by a correct algorithm is at least

$$\begin{aligned}\text{pr}[C] \cdot 1 + \text{pr}[\bar{C}] \cdot (T+1) &= 1 + T(1 - \text{pr}[C]) \\ &\geqq 1 + T/2 - T^2 2^{12-T} - T2^{2-2T} \\ &> T/2 \quad \text{for } T \geqq 21.\end{aligned}$$

Proof of the Claim. Le
of a query in $U_t(Y, X)$ ir
Lemma 2, the outcome of
after step t, it follows that
processors that know the
prove by induction on t t

pr

Before the first step, $t = 0$. Now assu

$$\text{pr}[Q_{t+1}(Y, X) \cap L$$
$$\leqq \text{pr}[Q_t(Y, X$$
$$+ \text{pr}[Q_{t+1}($$

By the induction hypothe

pr

Therefore, we suppose Q
If $Q_{t+1}(Y, X) \cap L_{t+1}($
(1) There is a proces
at the end of step t and,
(2) There is a proce
(in $L_t(Y, X)$) that knows
We handle these two
First, consider the s
processors in $Q_t(Y, X)$.
most one processor at step
Let N be the set of nodes
at least one of these proc
most 2^t nodes; therefore
they do not know the ou
of some of these queries
When the outcomes
$R_{t+1}(Y, X)$ is equally like
is because no label is rep
are chosen independently
the subtrees of $S_t(Y, X)$
probability that $S_{t+1}(Y, X$
Hence 2^{-T} is an upper bo
that knows the label of
reads from a processor in
Next, we show the p
which, at step $t+1$, read
node in $S_{t+1}(Y, X)$ at the
For any processor P,
are known by P immedia
more, it follows from Le
the processor that Q read
processor Q does not kn

Proof of the Claim. Let $Q_t(Y, X)$ denote the set of processors that know the outcome of a query in $U_t(Y, X)$ immediately after step t. (Since there are 2^m queries and, by Lemma 2, the outcome of no query is known by more than 2^t processors immediately after step t, it follows that $|Q_t(Y, X)| \leqq 2^{m+t}$.) Similarly, let $L_t(Y, X)$ denote the set of processors that know the label of some node in $S_t(Y, X)$ immediately after step t. We prove by induction on t that

$$\text{pr}[Q_t(Y, X) \cap L_t(Y, X) \neq \phi] \leqq t2^{12-T}.$$

Before the first step, each processor knows at most one input bit. Hence the claim is true for $t = 0$. Now assume the claim is true for t, where $0 \leqq t < T$. Then

$$\begin{aligned}\text{pr}[Q_{t+1}(Y, X) \cap L_{t+1}(Y, X) \neq \phi] &\leqq \text{pr}[Q_t(Y, X) \cap L_t(Y, X) \neq \phi] \\ &\quad + \text{pr}[Q_{t+1}(Y, X) \cap L_{t+1}(Y, X) \neq \phi \mid Q_t(Y, X) \cap L_t(Y, X) = \phi].\end{aligned}$$

By the induction hypothesis,

$$\text{pr}[Q_t(Y, X) \cap L_t(Y, X) \neq \phi] \leqq t2^{12-T}.$$

Therefore, we suppose $Q_t(Y, X) \cap L_t(Y, X) = \phi$.

If $Q_{t+1}(Y, X) \cap L_{t+1}(Y, X) \neq \phi$ then either

(1) There is a processor (in $L_t(Y, X)$) that knows the label of a node in $S_{t+1}(Y, X)$ at the end of step t and, at step $t+1$, reads from a processor in $Q_t(Y, X)$, or

(2) There is a processor in $Q_t(Y, X)$ that, at step $t+1$, reads from a processor (in $L_t(Y, X)$) that knows the label of a node in $S_{t+1}(Y, X)$ at the end of step t.

We handle these two cases one at a time.

First, consider the set of processors in $L_t(Y, X)$ that, at step $t+1$, read from processors in $Q_t(Y, X)$. Each processor in $Q_t(Y, X)$ can have its message read by at most one processor at step $t+1$, so there are at most $|Q_t(Y, X)| \leqq 2^{m+t}$ such processors. Let N be the set of nodes in $S_t(Y, X)$ whose labels are known at the end of step t by at least one of these processors. By Lemma 1, each processor knows the label of at most 2^t nodes; therefore $|N| \leqq 2^{m+2t}$. Since no processors in $L_t(Y, X)$ are in $Q_t(Y, X)$, they do not know the outcome of any query in $U_t(Y, X)$, so changing the outcomes of some of these queries cannot change the set N.

When the outcomes of the queries in $U_t(Y, X)$ are allowed to vary, the node $R_{t+1}(Y, X)$ is equally likely to be any one of the 2^k nodes of depth k in $S_t(Y, X)$. This is because no label is repeated along any path and the labels of the nodes in $S_t(Y, X)$ are chosen independently of the outcomes of the queries in $U_t(Y, X)$. At most $|N|$ of the subtrees of $S_t(Y, X)$ rooted at these 2^k nodes can contain elements of N. Thus the probability that $S_{t+1}(Y, X)$ contains some node in N is at most $|N|2^{-k} \leqq 2^{m+2t-k} \leqq 2^{-T}$. Hence 2^{-T} is an upper bound on the probability that there is a processor (in $L_t(Y, X)$) that knows the label of a node in $S_{t+1}(Y, X)$ at the end of step t and, at step $t+1$, reads from a processor in $Q_t(Y, X)$.

Next, we show the probability is also small that there is a processor in $Q_t(Y, X)$ which, at step $t+1$, reads from a processor (in $L_t(Y, X)$) that knows the label of a node in $S_{t+1}(Y, X)$ at the end of step t.

For any processor P, let $N_P(Y, X)$ be the set of nodes in $S_{t+1}(Y, X)$ whose labels are known by P immediately after step t. If $P \notin L_t(Y, X)$, then $N_P(Y, X) = \phi$. Furthermore, it follows from Lemma 1 that $|N_P(Y, X)| \leqq 2^t$. Let $Q \in Q_t(Y, X)$ and let P_Q be the processor that Q reads from at step $t+1$. Note that, since $Q_t(Y, X) \cap L_t(Y, X) = \phi$, processor Q does not know the label of any node in $S_t(Y, X)$, so changing the label

of any node in $S_t(Y, X)$ does not change Q's state and, hence, which processor Q reads from at step $t+1$.

We prove that, with probability less than $2^{-T}+2^{11-T}$, the subtree $S_{t+1}(Y, X)$ contains a node whose label is known by some processor in $L_t(Y, X)$ that was read by a processor in $Q_t(Y, X)$. Since there are no more than $2^{m+t} \leqq 2^{4T}$ processors in $Q_t(Y, X)$, it suffices to prove that for an arbitrary processor $Q \in Q_t(Y, X)$, the probability $S_{t+1}(Y, X)$ contains a node whose label is known by processor P_Q is less than $2^{-5T}+2^{11-5T}$.

On input (Y, X), the state of processor Q is determined by the outcomes of at most 2^t queries. For any other input in which these queries have the same outcomes, processor Q will also be in the same state. Thus we may partition the set of possible outcomes for the queries into those that give rise to the same state of Q. Since we may assume, without loss of generality, that processors do not forget information, each class of the partition can be specified by a set Z of at most 2^t queries and outcomes z for those queries. Then

$$\begin{aligned}&\operatorname{pr}[N_{P_Q}(Y, X) \cap S_{t+1}(Y, X) \neq \phi] \\ &\quad = \sum_{(Z,z)} \operatorname{pr}[N_{P_Q}(Y, X) \cap S_{t+1}(Y, X) \neq \phi \mid Z = z] \operatorname{pr}[Z = z],\end{aligned}$$

where the sum is taken over pairs (Z, z), one for each class of the partition. Now $\sum_{(Z,z)} \operatorname{pr}[Z = z] = 1$, so it suffices to show that

$$\operatorname{pr}[N_{P_Q}(Y, X) \cap S_{t+1}(Y, X) \neq \phi \mid Z = z] < 2^{-5T}+2^{11-5T}$$

for any pair (Z, z) that specifies a class of the partition.

We will show that for most labelings y of the decision tree, the nodes whose labels are known by processor P_Q are unlikely to be contained in $S_{t+1}(y, X)$, where the probability is taken over all inputs that satisfy $Z = z$. Note that the set of nodes whose labels are known by processor P_Q may be a function of the labels of the nodes in the decision tree.

A path in $S_t(Y, X)$ from $R_t(Y, X)$ to a node at depth k is said to be *constrained* if it contains at least 11 nodes labeled by variables in Z. Let E be the event that no path of length k, starting from $R_t(Y, X)$, is constrained. Then

$$\begin{aligned}&\operatorname{pr}[N_{P_Q}(Y, X) \cap S_{t+1}(Y, X) \neq \phi \mid Z = z] \\ &\quad = \operatorname{pr}[N_{P_Q}(Y, X) \cap S_{t+1}(Y, X) \neq \phi \mid Z = z \wedge \bar{E}] \operatorname{pr}[\bar{E}] \\ &\qquad + \operatorname{pr}[N_{P_Q}(Y, X) \cap S_{t+1}(Y, X) \neq \phi \mid Z = z \wedge E] \operatorname{pr}[E] \\ &\quad \leqq \operatorname{pr}[\bar{E}] + \operatorname{pr}[N_{P_Q}(Y, X) \cap S_{t+1}(Y, X) \neq \phi \mid Z = z \wedge E].\end{aligned}$$

The labels of the nodes on a path can be viewed as being selected without replacement from the set $U_t(Y, X)$. Thus the probability that a particular path is constrained is at most

$$\binom{k}{11}\left(\frac{|Z|}{|U_t(Y, X)|-k}\right)^{11} < \left(\frac{k|Z|}{|U_t(Y, X)|-k}\right)^{11}$$

and the probability $\operatorname{pr}[\bar{E}]$ that some path in the tree is constrained is less than

$$2^k\left(\frac{k|Z|}{|U_t(Y, X)|-k}\right)^{11} \leqq 2^k\left(\frac{k2^t}{2^m-k(t+1)}\right)^{11} \leqq 2^{-5T} \quad \text{for } T \geqq 5.$$

Now consider any la
to $R_t(Y, X)$ and in whi
probability that $R_{t+1}(y, X$
Thus the probability tha
contains a node in $N_{P_Q}($

Hence, $\operatorname{pr}[N_{P_Q}(y, X) \cap$
$S_{t+1}(Y, X) \neq \phi \mid Z = z] \leqq$
Combining the infor

$$\operatorname{pr}[Q_{t+$$
$$<2$$

and

$$\operatorname{pr}[Q_{t+1}(Y, X) \cap$$
$$\leqq \operatorname{pr}[Q_t(Y,$$
$$+ \operatorname{pr}[Q_{t+$$
$$< t2^{12-T}+2$$

Thus the claim is true.

5. Conclusions. This
that can be computed by
that computes it has ex
similar separation betwe
ing Boolean functions, C
a constant factor) and, I
this is not necessarily th
the OR of n Boolean va

There is a very clos
[4]. If a function (over a
then it can be computed
that any function compu
in constant time on a PI
be computed by a decis
PRAM in time $\lceil \log_2 h$
complete for CROW I
$f: \{0, 1\}^n \to R$ can be com
by an EREW PRAM (o
it takes to compute $D_{n,2}$

We conjecture that
obtained for the Boolea
m and h. Unfortunately
straightforward way. Th
information about some
the fact that no value wa
[2] for details.) The defi

Now consider any labeling y of the decision tree that agrees with Y from the root to $R_t(Y, X)$ and in which no path of length k from $R_t(Y, X)$ is constrained. The probability that $R_{t+1}(y, X)$ is any particular node at level k of $S_t(y, X)$ is at most 2^{11-k}. Thus the probability that $S_{t+1}(y, X)$, the subtree of $S_t(y, X)$ rooted at $R_{t\geq 1}(y, X)$, contains a node in $N_{P_Q}(y, X)$ is at most

$$2^{11-k}|N_{P_Q}(y, X)| \leqq 2^{11-k+t} \leqq 2^{11-5T}.$$

Hence, $\operatorname{pr}[N_{P_Q}(y, X) \cap S_{t+1}(y, X) \neq \phi \mid Z = z \wedge E] \leqq 2^{11-5T}$ and $\operatorname{pr}[N_{P_Q}(Y, X) \cap S_{t+1}(Y, X) \neq \phi \mid Z = z] \leqq 2^{-5T} + 2^{11-5T}$, as desired.

Combining the information about both cases, we get that

$$\begin{aligned}&\operatorname{pr}[Q_{t+1}(Y, X) \cap L_{t+1}(Y, X) \neq \phi \mid Q_t(Y, X) \cap L_t(Y, X) = \phi] \\ &\quad < 2^{-T} + 2^{-T} + 2^{11-T} < 2^{12-T}\end{aligned}$$

and

$$\begin{aligned}&\operatorname{pr}[Q_{t+1}(Y, X) \cap L_{t+1}(Y, X) \neq \phi] \\ &\quad \leqq \operatorname{pr}[Q_t(Y, X) \cap L_t(Y, X) \neq \phi] \\ &\qquad + \operatorname{pr}[Q_{t+1}(Y, X) \cap L_{t+1}(Y, X) \neq \phi \mid Q_t(Y, X) \cap L_t(Y, X) = \phi] \\ &\quad < t2^{12-T} + 2^{12-T} = (t+1)2^{12-T}.\end{aligned}$$

Thus the claim is true.

5. Conclusions. This paper shows that there is a Boolean function of n variables that can be computed by a CROW PRAM in $O(\log \log n)$ steps, but any EROW PRAM that computes it has expected running time in $\Omega(\sqrt{\log n})$. We think that there is a similar separation between CROW PRAMS and EREW PRAMs. Note that, for computing Boolean functions, CROW PRAMs are as powerful as CREW PRAMs (to within a constant factor) and, hence, are at least as powerful as EREW PRAMs. However, this is not necessarily the case if the domain is not complete. (For example, consider the OR of n Boolean values, at most one of which is 1.)

There is a very close correspondence between CROW PRAMs and decision trees [4]. If a function (over any domain) can be computed by a CROW PRAM in time T, then it can be computed by a decision tree of height 2^T. (In particular, this implies that any function computable in constant time on a CROW PRAM is also computable in constant time on a PRAM with only one processor.) Conversely, if a function can be computed by a decision tree of height h, then it can be computed by a CROW PRAM in time $\lceil \log_2 h \rceil + 1$. Thus the Boolean decision-tree evaluation problem is complete for CROW PRAM computation in the following sense. If a function $f: \{0, 1\}^n \to R$ can be computed by a CROW PRAM in time $t(n)$, then f can be computed by an EREW PRAM (or any other model of computation) using no more time than it takes to compute $D_{n,2^{t(n)}}$.

We conjecture that, on the EREW PRAM, a $(\log n)^{\Omega(1)}$ lower bound can be obtained for the Boolean decision-tree evaluation problem for appropriate choices of m and h. Unfortunately, the proof of Theorem 3 does not appear to generalize in a straightforward way. The essential problem is that, on CREW and EREW PRAMs, information about some input bits can be transmitted to a memory cell by virtue of the fact that no value was written there during a particular step of a computation. (See [2] for details.) The definition of knowledge must be modified to take this into account.

For their CREW PRAM lower bound, Cook, Dwork, and Reischuk [2] used the following definition to capture certain properties of knowledge. A processor or memory cell is said to be *affected* by a particular input bit at time t on input x if the state of the processor or the contents of the memory cell immediately after step t of the computation is different for x than for the input obtained from x by changing the value of the specified input bit. This definition supports lemmas analogous to Lemmas 1 and 2, provided the input domain is assumed to be $\{0, 1\}^n$.

LEMMA 4 (Cook, Dwork, and Reischuk [2]). *For a* CREW PRAM, *on any input, every processor and memory cell is affected by at most* $(\frac{1}{2}(5+\sqrt{21}))^t$ *input bits immediately after step t.*

LEMMA 5 (Beame [1]). *For an* EREW PRAM, *on any input, each input bit affects at most* $(2+\sqrt{3})^t$ *processors and memory cells immediately after step t.*

There is another fact about knowledge used in the proof of Theorem 3 that, unfortunately, is not shared by the affects relation. If a processor or memory cell does not know certain input bits, then changing all of their values does not change the state of the processor or the contents of the memory cell. Moreover, the set of input bits that the processor or memory cell knows remains unchanged.

This motivates the following definition. A set of input bits is a *dependency set* for a processor or memory cell at time t on input x if the state of the processor or the contents of the memory cell immediately after step t of the computation is the same for x as it is for the inputs obtained from x by changing the values of any set of bits not in the dependency set. Furthermore, there is another version of Lemma 1 that holds for the CREW PRAM with the complete input domain $\{0, 1\}^n$.

LEMMA 6 (Nisan [7]). *For a* CREW PRAM, *on any input, every processor and memory cell as a dependency set containing at most* $(\frac{1}{2}(5+\sqrt{21}))^{2t}$ *input bits immediately after step t.*

Note that this lemma does not imply that, after a small number of steps, all (minimal) dependency sets are small. For example, consider the contents of the output cell at the end of the computation of $D_{m,1}: \{0, 1\}^{m+2^m} \to \{0, 1\}$. Recall that for $y \in \{0, 1\}^m$ and $x_0, \cdots, x_{2^m-1} \in \{0, 1\}$,

$$D_{m,1}(y, x_0, \cdots, x_{2^m-1}) = x_y$$

and this function can be computed in $O(\log m)$ steps. On input 0^{m+2^m}, the last 2^m bits comprise a minimal dependency set.

Even if we could somehow associate a small dependency set with each processor and memory cell, the corresponding version of Lemma 2 would also be false. Suppose, for example, that during the first $O(t)$ steps of a computation, a processor accumulates the values of 2^t input bits and then writes a special value to the memory indexed by this 2^t-tuple. Each of the 2^{2^t} memory cells in which the special value can appear must have at least one of these 2^t input bits in its associated dependency set. Hence, at least one of these input bits must be a member of the dependency sets associated with at least 2^{2^t-t} different shared memory cells.

The notions of affects and dependency set do not suffice to extend the proof of Theorem 3. However, understanding these and other related definitions will provide us with additional insight into the nature of exclusive read. We also believe that a definition of knowledge can be obtained to show the Boolean decision-tree evaluation problem is hard on EREW PRAMs.

The CROW PRAM model can be extended by allowing each processor to own many different shared memory cells, instead of just one. At each timestep, a processor could write to any one of the memory cells it owns. However, each memory cell would

still be owned by only o
CROW PRAM because
memory cell to record th
the locations to which th
then read this informatio

The EROW PRAM
whether the resulting mo
is less powerful than the
desired separation betwe
that the Boolean decisio

[1] P. BEAME, *Lower bounds in*
Science, University of
[2] S. COOK, C. DWORK, AN
machines without simu
[3] P. DYMOND AND W. L. RU
language recognition, i
ming, 1986, pp. 95–10
[4] F. FICH AND P. RAGDE, u
[5] E. GAFNI, J. NAOR, AND
Sci. to appear.
[6] R. KARP AND V. RAMACH
Report UCB/CSD 88/
[7] N. NISAN, CREW PRAM
Computing, Associati
[8] M. SNIR, *On parallel searc*
[9] A. YAO, *Probabilistic compu*
Symposium on Found
1977, pp. 222–227.

still be owned by only one processor. This new model is no more powerful than the CROW PRAM because each CROW PRAM processor could use its single shared memory cell to record the entire sequence of values that would have been written and the locations to which they would have been written. All interested processors could then read this information.

The EROW PRAM model can be extended in the same way. But it is not clear whether the resulting model is more powerful than the EROW PRAM and whether it is less powerful than the EREW or CROW PRAMs. One approach to obtaining our desired separation between EREW and CROW PRAMs is to first attempt to prove that the Boolean decision-tree evaluation problem is hard on this model.

REFERENCES

[1] P. BEAME, *Lower bounds in parallel machine computation*, Tech. Report 198/87, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 1987.

[2] S. COOK, C. DWORK, AND R. REISCHUK, *Upper and lower time bounds for parallel random access machines without simultaneous writes*, SIAM J. Comput., 15 (1986), pp. 87-98.

[3] P. DYMOND AND W. L. RUZZO, *Parallel* RAMs *with owned global memory and deterministic context-free language recognition*, in Proc. 13th International Colloquium on Automata, Languages and Programming, 1986, pp. 95-104.

[4] F. FICH AND P. RAGDE, unpublished manuscript.

[5] E. GAFNI, J. NAOR, AND P. RAGDE, *On separating the* EREW *and* CROW *models*, Theoret. Comput. Sci. to appear.

[6] R. KARP AND V. RAMACHANDRAN, *A survey of parallel algorithms for shared-memory machines*, Tech. Report UCB/CSD 88/408, University of California, Berkeley, CA, 1988.

[7] N. NISAN, CREW PRAMs *and decision trees*, in Proc. 21st Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, 1989, pp. 327-335.

[8] M. SNIR, *On parallel searching*, SIAM J. Comput., 14 (1985), pp. 688-708.

[9] A. YAO, *Probabilistic computations: toward a unified measure of complexity*, in Proc. 18th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, 1977, pp. 222-227.