Towards Understanding Exclusive Read

Faith E. Fich
University of Toronto
Toronto, Canada

Avi Wigderson
Hebrew University
Jerusalem, Israel

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, we describe a natural Boolean function of n variables and show 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.

1 Introduction

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

> given distinct inputs $x_1, \ldots, x_n$, 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 doesn't 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 [CDR]. In contrast, Nisan [N] proved that any Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ has, to within a small constant factor, the same time complexity on CREW and CROW PRAMs.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and / or specific permission.

© 1989 ACM 0-89791-323-X/89/0006/0076 $1.50

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 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 [GNR] 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.

It remains open whether all Boolean functions can be computed as quickly by an EREW PRAM as by a CREW PRAM. We make progress towards 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.

AUG-26-2002 15:51 NYU#ILL#FAX(212)995-4071 P.07/13

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 are 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 [DR], 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 corresponds to a complete network in which each posted message can be read by at most one processor at a time. (Notice 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 is the union of the bits known by the two processors before the read occurred. Changing the value of any input bits that a processor doesn't 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 [CDR]) *For a CROW PRAM, on any input, every processor knows at most 2^t input bits immediately after step t.*

Lemma 2 (Beame [B]) *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 labelled 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 labelling the leaf reached by this path.

For example, given the decision tree in Figure 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.

Let $D_{m,h} : \{0,1\}^{m(2^{h+1}-1)+2^m} \rightarrow \{0,1\}$ be a Boolean function representing a Boolean decision tree evaluation problem for a complete binary tree of height h in which every node is labelled 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 labelling the ith node in the tree. The last 2^m bits, $x_0, \ldots, x_{2^m-1}$, denote the outcomes of the

AUG-26-2002 15:51 NYU#ILL#FAX(212)995-4071 P.08/13

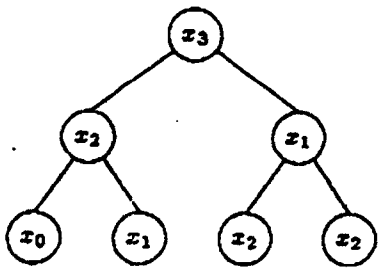

Figure 1: A decision tree.

queries. For example, $D_{2,2}(1110010001101011 0) = 1$. (See Figure 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 labelling it.

An $O(\log m + \log h)$ upper bound for this problem can be obtained on the CROW PRAM as follows. One processor is assigned to each node of the decision tree. Each of these processors begins by reading the label of its node (in $O(\log m)$ steps with the help of $O(m/\log m)$ additional processors). In the next step, the processor reads the outcome of the appropriate query. This defines a pointer from the node to one of its two children. Using pointer jumping [KR], the unique path from the root to a leaf can be determined in $O(\log h)$ steps, even by an EROW PRAM. Notice that in this solution, there are potentially many processors that may read the outcome of the same query at the same time.

The following CROW PRAM algorithm, although less efficient, is more easily converted into an algorithm that uses only exclusive reads. For each of the 2^{2^m} possible outcomes for the queries, a group of 2^h 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.) This creates the table of values for the function. 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 $X_0, \ldots, X_{2^m-1}$ be random variables whose values are independently and uniformly chosen from the range $\{0, 1\}$. The sequence of random variables $X = (X_0, \ldots, X_{2^m-1})$ is used to denote the outcomes of the queries. Let $Y_1, \ldots, Y_{2^{h+1}-1}$ be random variables with range $\{0, \ldots, 2^m - 1\}$. The sequence $Y = (Y_1, \ldots, Y_{2^{h+1}-1})$ is used to denote a labelling 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 labelled, the label of the node is chosen uniformly among those queries not labelling any of its ancestors. This gives us a uniformly chosen labelling of the decision tree with the property that all nodes along any path from the root to a leaf are labelled by different queries. It suffices to show [Y] 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 $t = 0, \ldots, T$, the node at depth kt on the path determined by the query outcomes X in the decision tree labelled 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 Figure 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 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 $R_T(Y, X)$

AUG-26-2002 15:52 NYU#ILL#FAX(212)995-4071 P.09/13

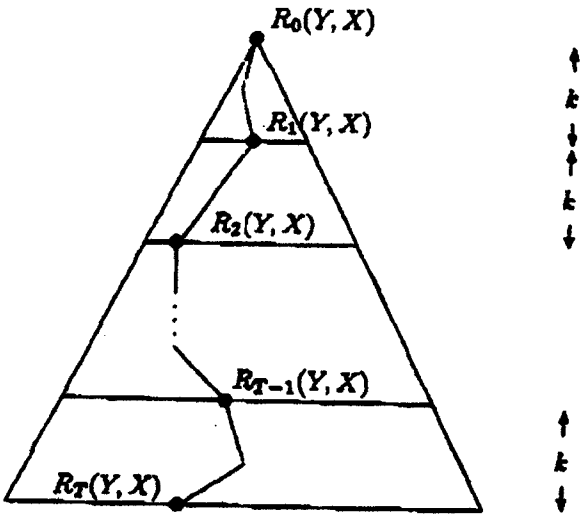

Figure 2: A decision tree sliced into pieces.

within T steps. We will show that the probability processor P_1 has the correct answer is only slightly more than $\frac{1}{2}$.

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 labelling $R_T(Y,X)$ is equally likely to be 0 or 1, it follows that

$$pr[C|\bar{A}] \leq \frac{1}{2}.$$

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 $U_T(Y,X)$ doesn't 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,

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

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

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 $pr[C]\cdot 1 + pr[\bar{C}]\cdot(T+1) = 1 + T(1-pr[C]) \geq 1 + T/2 - T^2 2^{12-T} - T2^{2-2T} > T/2$ for $T \geq 21$.

Proof of 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 $|Q_t(Y,X)| \leq 2^{m+t}$.) 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

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

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 $0 \leq t < T$. Then

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

By the induction hypothesis,

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

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 $|Q_t(Y,X)| \leq 2^{m+t}$ 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 $|N| \leq 2^{m+2t}$. 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

AUG-26-2002 15:53 NYU#ILL#FAX(212)995-4071 P.10/13

changing the outcomes of some of these queries cannot change the set N.

When the outcome 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 $|N|2^{-k} \leq 2^{m+2t-k} \leq 2^{-T}$. 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 $|N_P(Y,X)| \leq 2^t$. 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 $S_t(Y,X)$ doesn't 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 $2^{m+t} \leq 2^{4T}$ 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 $pr[N_{P_Q}(Y,X) \cap S_{t+1}(Y,X) \neq \phi] =$

$$\sum_{(Z,z)} pr[N_{P_Q}(Y,X) \cap S_{t+1}(Y,X) \neq \phi | Z = z] pr[Z = z],$$

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

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

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

We will show that for most labellings 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 labelled by variables in Z. Let E be the event that no path of length k, starting from $R_t(Y,X)$, is constrained. Then

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

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

$$\begin{aligned} 2^k \left(\frac{k|Z|}{|U_t(Y,X)| - k}\right)^{11} &\leq 2^k \left(\frac{k2^t}{2^m - k(t+1)}\right)^{11} \\ &\leq 2^{-5T}, \text{ for } T \geq 5. \end{aligned}$$

Now consider any labelling 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 $R_{t+1}(y,X)$, contains a node in $N_{P_Q}(y,X)$ is at most

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

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

Combining the information about both cases, we get that

AUG-26-2002 15:53 NYU#ILL#FAX(212)995-4071 P.11/13

$$
\begin{aligned}
&pr[Q_{t+1}(Y,X) \cap L_{t+1}(Y,X) \neq \phi \mid \\
&\quad Q_t(Y,X) \cap L_t(Y,X) = \phi] \\
&< 2^{-T} + 2^{-T} + 2^{11-T} < 2^{12-T} \\
\text{and } &pr[Q_{t+1}(Y,X) \cap L_{t+1}(Y,X) \neq \phi] \\
&\leq pr[Q_t(Y,X) \cap L_t(Y,X) \neq \phi] \\
&\quad + pr[Q_{t+1}(Y,X) \cap L_{t+1}(Y,X) \neq \phi \mid \\
&\quad Q_t(Y,X) \cap L_t(Y,X) = \phi] \\
&< t2^{12-T} + 2^{12-T} = (t+1)2^{12-T}.
\end{aligned}
$$

Thus the claim is true.

5 Conclusions

This paper shows that there is a Boolean function of n variables which 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 [FR]. 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 $f : \{0,1\}^n \rightarrow R$ 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 [CDR] for details.) The definition of knowledge has to be modified to take this into account.

For their CREW PRAM lower bound, Cook, Dwork, and Reischuk [CDR] 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 [CDR]) *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 [B]) *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 know remain 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 [N]) *For a CREW PRAM, on any input, every processor and memory cell has a dependency set containing at most* $(\frac{1}{2}(5+\sqrt{21}))^{2t}$ *input bits immediately after step* t.

Notice, 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 $D_{m,1}$:

AUG-26-2002 15:54 NYU#ILL#FAX(212)995-4071 P.12/13

TOTAL P.13

$\{0,1\}^{m+2^m} \rightarrow \{0,1\}$. Recall that for $y \in \{0,1\}^m$ and $z_0, \ldots, z_{2^m-1} \in \{0,1\}$,

$$D_{m,1}(y, z_0, \ldots, z_{2^m-1}) = z_y$$

and that 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 time step, a processor could write to any one of the memory cells it owns. However, each memory cell would 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 they would have been written to. 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.

Acknowledgements

This work was partially supported by the Information Technology Research Centre of Ontario. The first author was also partially supported by the Natural Sciences and Engineering Research Council of Canada grant A9176 and a University of Toronto grant. The second author was also partially supported by an Alon Fellowship and the American-Israeli Binational Science Foundation.

References

[B] P. Beame, "Lower Bounds in Parallel Machine Computation", Tech. Report 198/87, Department of Computer Science, University of Toronto, 1987.

[CDR] S. Cook, C. Dwork, and R. Reischuk, "Upper and Lower Time Bounds for Parallel Random Access Machines Without Simultaneous Writes", *SIAM J. Comput.*, vol. 15, no. 1, February 1986, pages 87-98.

[DR] P. Dymond and W.L. Ruzzo, "Parallel RAMs with Owned Global Memory and Deterministic Context-Free Language Recognition", *Proceedings of the Thirteenth International Colloquium on Automata, Languages and Programming*, 1986, pages 95-104.

[FR] F. Fich and P. Ragde, unpublished manuscript.

[GNR] E. Gafni, J. Naor, and P. Ragde, "On Separating the EREW and CROW Models", to appear in Theoretical Computer Science.

[KR] R. Karp and V. Ramachandran, "A Survey of Parallel Algorithms for Shared-Memory Machines", Report UCB/CSD 88/408, University of California, Berkeley, 1988.

[N] N. Nisan, "CREW PRAMS and Decision Trees", to appear in the *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, 1989.

[S] M. Snir, "On Parallel Searching", *SIAM J. Comput.*, vol. 14, no. 3, August 1985, pages 688-708.

[Y] A. Yao, "Probabilistic Computations: Toward a Unified Measure of Complexity", *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science*, 1977, pages 222-227.

AUG-26-2002 15:54 NYU#ILL#FAX(212)995-4071 P.13/13