Proposed running title: Decision Trees with Boolean Threshold queries

Mailing address:
Ilan Newman
Dept. of Mathematics and Computer Science
University of Haifa
Haifa, 31905, Israel

Decision trees with Boolean threshold queries

Yosi Ben-Asher and Ilan Newman
Dept. of Mathematics and Computer Science, Haifa University

Abstract

We investigate decision trees in which one is allowed to query threshold functions of subsets of variables. We are mainly interested in the case were only queries of AND and OR are allowed. This model is a generalization of the classical decision tree model. Its complexity (depth) is related to the parallel time that is required to compute Boolean functions in certain CRCW PRAM machines with only one cell of constant size. It is also related to the computation using Ethernet channel.

We prove a tight lower bound of $\theta(k \log(n/k))$ for the required depth of a decision tree for the threshold-k function. As a corollary of the method we also prove a tight lower bound for the 'direct sum' problem of computing simultaneously k copies of threshold-2 in this model.

Next, the size complexity is considered. A relation to depth-three circuits is established and a lower bound is proven. Finally the relation between randomized, nondeterminism and determinism is also investigated, we show separation results between these models.

1 Introduction

The model of Boolean decision tree is a fundamental model in the theory of Boolean functions. It provides a very important measure of complexity and it has been investigated thoroughly in the literature [23, 20, 13, 24, 16, 18, 7].

In the basic model only queries of single variables are allowed. Several generalizations have been considered in the past [6, 17, 7]. Recently [7] investigated the complexity of such a model when general threshold functions are allowed as queries. We are interested in case were the set of queries is limited to Boolean thresholds. In particular, we are interested in the case where we allow only queries of OR and AND (threshold 1 and threshold n respectively). Obviously even the model of OR (AND) queries is a proper generalization of the standard decision tree model. The case were we allow both AND and OR queries is however more interesting because it is also related to a certain complexity measure in the PRAM model. Consider a computation of a Boolean function in a CRCW PRAM machine that is equipped only with a one bit-cell (constant number of cells) and the n-bit input is shared between a set of n processors $p_1, ..., p_n$ so that p_i gets x_i (for a more formal definition see subsection 2.1). Each round can be viewed as computing the OR of all processors that potentially write on input '1', or the AND of all processors that potentially write on input '0'. Thus an algorithm is in fact a decision tree in which queries of $\vee, \wedge$ of subsets of inputs are allowed.

Lower bounds for similar PRAM models with one cell but in which the input is written on a separate read-only tape were considered in the literature. Some general methods for proving lower bounds were developed in [22, 4]. However, for our model, none of the existing methods seem to help even in the 'simple' question of determining the complexity of the threshold-k function. Another related question is yet another PRAM model in which there is a broadcast channel that can be either CREW or with the following property: if more than one processor tries to broadcast, there is a signal that indicates a violation. The first model is equivalent to $(\vee, \wedge)$-decision trees and to CRCW PRAM with a cell of one bit. The second model lies between the arbitrary and the common model. Its power is not well understood, however, it seems that technologically it is quite interesting as it models, to that extent, the Ethernet channel. The related decision tree in this case would be a tree with queries of thresholds-1 and threshold-2 of variables or their negations.

In this paper, we consider the case of $\vee, \wedge$ queries. In that case it is quite easy to prove lower bounds for parity and majority. As will be explained latter quite strong size lower bounds for these functions can also be proven due to their 'hardness' for AC^0. We will focus mainly on the problem of computing the threshold-k function, T_k^n. We are interested specifically in the computation of threshold functions for several reasons. Usually threshold functions are considered a 'first' test case to check because of their simplicity and importance. Here, as we use $\wedge$ and $\vee$ queries (the extreme thresholds), computing T_k^n is especially interesting as it exhibits the relative power of different thresholds. Another reason stems from the motivation of computing in the presence of Ethernet channels or buses. In these cases the value of T_k^n is used as approximate counting which is often used as a building block for many algorithms such as routing etc. [19, 5]. Finally a 'technical' reason is that T_k^n has relatively small depth-three circuits. As will be explained latter, functions with high depth-three circuit complexity (e.g. parity) cannot be 'efficiently' computed by $(\vee, \wedge)$-decision trees.

We prove a tight lower bound of $\theta(k \log \frac{n}{k})$ for the threshold-k function (on n variables). As a corollary we get the same bounds for PRAMs with one cell and for CRCW channel. The method we use yields also two other corollaries: A lower bound of $\Omega(k \log \frac{n}{k})$ for any function f for which its value is fixed on the k-slice and is different from the value on the $(k-1)$-slice. The other is a lower bound for a certain 'direct-sum' problem.

In general, a 'direct-sum' problem is to compute simultaneously r copies of a given task on disjoint set of variables. In the standard decision tree model the question is quite uninteresting as queries do not involve variables of more than one of the problem instances at a time. However, here, as in more 'complex' models of computation, the question is non trivial, quite interesting and sometimes with possibly important implications, [14]. In our case it can be shown that to compute T_2^n (threshold-2 of n variables), requires $\theta(\log n)$ $\wedge$, $\vee$ queries. What about computing simultaneously r copies of T_2^n on r disjoint sets of n variables? Indeed we can show that $\Omega(r \log n)$ queries is required.

Generally, the depth complexity is of primal interest for decision trees. However, as bounds on the size imply bounds on the depth, the size complexity should be considered too. As it turns out proving lower bounds on the size is quite harder. We discuss the size for general Boolean functions and show a connection to depth-three circuit complexity. This connection may be used to prove lower bounds on $(\vee, \wedge)$ decision tree size for functions with known 'strong' lower bounds for depth-three circuits, e.g. parity and majority. For T_k^n this is not the case. We prove a direct lower bound on the size for T_k^n. The lower bound is quite 'weak', however, it competes with the known lower bounds that come from depth-three circuits for small k. Again, as a corollary we get a size lower bound for any function with a certain fixed behavior on some slices.

Finally, nondeterministic and a randomized versions are defined. We show that, in contrast to the the standard decision tree model, there is no relation between these measures of complexity. In particular, the non deterministic complexity of any Boolean function is at most 2.

The rest of the paper is organized as follows: In section 2 we present the formal definitions, and also show the relations to the different PRAM models that were mentioned. Section 3 contains upper and lower bounds for threshold-k and for the 'direct-sum' problem. Section 4 contains a discussion on the size complexity and a lower bound. In Section 5 we define and discuss the nondeterministic and randomized related models. Section 6 contains some open problems and a discussion.

Acknowledgment: We wish thank Avi Wigderson for his valuable comments.

2 Definitions and Notations

We denote by $T_k^n(x_1, ..., x_n), 1 \leq k \leq n$ the Boolean function of n Boolean variables $(x_1, ..., x_n)$ that has the value 1 iff $\sum_{i=1}^n x_i \geq k$. T_k^n is called the 'threshold-k function'.

Let $X = \{x_1, ..., x_n\}$ be a set of Boolean variables. A Boolean threshold decision tree on X is a binary tree. Each internal node is labeled by a subset of variables S, and a Boolean threshold function on S. Each leaf is labeled by either 0 or 1 and the two out-going edges of each internal node are labeled by 0 and 1. Such a tree naturally defines a Boolean function: Each assignment to the variables determines a unique path from the root to a leaf by the value of the thresholds in the internal nodes. The label of the leaf which is the end of that path is the value of the function. The complexity of such a tree is its depth (i.e the number of queries for the worst case input).

Clearly every Boolean function has a Boolean threshold decision tree (in particular, this is clear as this is a proper generalization of the classic decision tree model). For a Boolean function $g : \{0,1\}^n \to \{0,1\}$ we denote by $DT_T(g)$ the minimum complexity of all possible Boolean threshold decision trees for g. We use the subscript T for 'threshold', to refer to the type of functions that are allowed at the internal nodes ('queries'). In general the subscript will denote a set of allowed Boolean function. We will mainly deal with a restricted model, where we allow only $\wedge, \vee$ queries. In this case we will say that the decision tree is a $(\wedge, \vee)$-decision tree, and we will denote the respective complexity as $DT_{\wedge,\vee}(g)$.

Let $g : \{0,1\}^X \to \{0,1\}$ be a Boolean function on X. Along the sequel we will identify each input of g, i.e a truth assignment to the variables of X with a subset of X, namely, the subset of variables that are set to '1'. We denote by $[n]$ the set $\{1, ..., n\}$ and by $\binom{X}{k}$ the set $\{U \subseteq X \mid |U| = k\}$.

2.1 The relation of decision trees with Boolean threshold queries to PRAM models

We formally define here the two PRAM models that were introduced in section 1.

- Consider the following 'One cell PRAM' model. In such a machine (program) there are n processors that can communicate via a single cell of one bit size (equivalently, constant number of cells with constant size). In this model the input to the problem is an n bit string $x_1 ... x_n$ that is distributed among the n processors, so that the ith processor gets x_i. A step of the machine is a write cycle followed by a read cycle to the cell. If the access to this one bit cell is CREW, then the computation is equivalent to the standard decision tree model. This is so, as the computation branches as any step according to what is being written by a single processor p_i and thus depends only on the value x_i (and the history of the computation so far). If the model of access is CRCW (with one bit cell common, priority and arbitrary are equivalent up to multiplying the number of steps by a factor of two), the computation is equivalent to an ($\wedge$, $\vee$)-decision trees. Indeed such a machine can simulate an ($\wedge$, $\vee$)-decision trees as each step is just OR (or AND), which can be done in a single step in a CRCW PRAM. For the other direction, consider a step of the computation in such a machine. As remarked before, by multiplying the number of steps by two, we may assume that all processors that write wish to write '1'. In the write cycle, there may be some processors that may wish to write if their input is '1'. Others may want to write if their input is '0'. We redefine the step by splitting it into two steps according to the above distinction. Now, the first steps is just writing the OR of the inputs of all those who want to write. The second step (of those who wish to write on input '0') is just writing the negation of the OR on the inputs of these that potentially write. The read cycle makes the output of the write phase to become a global knowledge and the computation branches accordingly. Thus by multiplying the number of steps by (at most) four. Each such computation can be simulated by an ($\wedge$, $\vee$)-decision tree.

  We note that the model defined here is different from that of [22], in which there is a one cell for communication and the input string is given on a read only tape. In the case of [22], several groups of processors may find out the values of some inputs in one step by accessing the read only tape.

- Consider a network of buses (or Ethernet channels). In such a network, groups of processors are interconnected by buses or broadcast channels ([2, 1, 4, 5] etc.). A simplification is when all the processors are connected by just one bus. In that case, a step of the computation is just as in a PRAM with one cell. The access to the bus in this case may be CRCW common which is just as the model presented before. It could also be with the property that if two or more processors access the bus (for broadcasting) a noise signal is heard by all these who listen. In that case what is heard provides a ternary answer, namely - no transmission, exactly one processor transmitted, or more than one tried. It could be seen that this is equivalent a decision tree with the ability to query $\wedge$, $\vee$ and threshold two, or its negation.

3 The complexity of a $(\wedge,\ \vee)$-Decision Tree for the Threshold-k function

In this section we investigate the complexity of T_k^n when $\wedge, \vee$ queries are allowed. We prove a tight lower bound of $\theta(k \log \frac{n}{k})$ and use the methods we develop to answer the 'direct-sum' question that was presented in the introduction. It worth noting that a trivial adversary that fixes a variable on each query shows that at least k queries are required for T_k^n which yields a (optimal) lower bound of $\frac{n}{2}$ for majority. A similar argument shows that parity requires n queries.

Let us start by examining first the more restricted models.

3.1 Facts

1. T_k^n is evasive for all $1 \leq k \leq n$, that is, if only queries of single variables are allowed (the classical decision tree model) then n queries are required.

2. $DT_{\vee}(T_k^n) = \theta(k \log(n/k))$. In words, if we allow OR queries then one can proof a tight upper bound of $\theta(k \log(n/k))$. The upper bound follows from the following recursive procedure:

   If $n \leq 2k$ we query all variables and we are done. Otherwise we split the variables to $2k$ sets of equal size and query the OR on each (if n is not divisible by $2k$ then the last set may be smaller then the rest). If we get k or more positive answers we are done as there are at least k 1's in the input, otherwise, we got at least $k+1$ negative answers. These $k+1$ sets (of which one might be the small set in the case n is not divisible to $2k$) contain at least $\frac{kn}{2k} \geq \frac{n}{2}$ variables which we know are '0'. Thus we repeat the procedure recursively on the variables that are left until we get a '1' answer or the number of variables decreases below $2k$.

   This gives the following recursion $f(n) = 2k + f(n/2), \quad f(2k) = 2k$ for which the answer is $f(n) = O(k \log \frac{n}{k})$.

   The lower bound can be deduced by a stronger result, as stated in the following fact.

3. The size of any $\vee$-decision tree for T_k^n is at least $\binom{n}{k-1}$.

   Proof: Let T be a $\vee$-decision tree for T_k^n. Consider the set of input strings each containing exactly $k-1$ 1's. As explained before, we identify each such string with a subset of X of size $k-1$. As each such set is a '0'-input of T_k^n, it defines in T a path from the root to a leaf that is labeled by '0'. We claim that no two distinct $(k-1)$-sets define the same path. Indeed, assume that A, B are two distinct $(k-1)$-sets that arrive at the same leaf. As these two sets follow the same path from the root, they must get the same answers to queries along this path. It can be easily verified that in this case, $A \cup B$ is also consistent with the same path. However, $|A \cup B| \geq k$ and thus should arrive at a leaf labeled by '1'. Thus, the number of leaves is at least $\binom{n}{k-1}$ $\quad \square$.

3.2 Lower bounds

We turn now to the $(\wedge, \vee)$-decision tree complexity of T_k^n. As shown before it is at most $O(k \log(n/k))$. We show here that there is a matching lower bound.

Theorem 3.1 *$DT_{\wedge,\vee}(T_k^n) = \Omega(k \log \frac{n}{k})$.*

Again, we identify inputs of T_k^n with subsets of the variable set X. Thus subsets of size $k-1$ are 0-inputs of T_k^n while subsets of size k are 1-inputs.

The theorem will be proven using an adversary argument. The adversary will make sure that as long as not too many queries have been made, there are still a '1' input and a '0' input that are both consistent with the queries so far. Thus the value of T_k^n will not yet be determined. The difficulty will be to control the decrease in number of '1' inputs and '0' inputs simultaneously.

Some inter-relation between the family of consistent k-sets and the family of consistent $(k-1)$-sets will be required.

Definition 3.1 *Let $I_0 \subseteq \binom{X}{k-1}$ and $I_1 \subseteq \binom{X}{k}$ be two families of inputs (families of subsets of X of size $k-1$ and k respectively). An $(\wedge, \vee)$-decision tree, T,* **separates** *I_0 and I_1 if it answers '0' for each $a \in I_0$ and it answers '1' on each $b \in I_1$. Let $DT_{\wedge,\vee}(I_0, I_1)$ denote the minimum complexity of a decision tree that separates I_0 and I_1.*

Note, if T separates I_0 and I_1 then it also separates any I_0' and I_1' for which $I_0' \subseteq I_0$, $I_1' \subseteq I_1$. Further more, if v is an internal node of T and $A \subseteq I_0$, $B \subseteq I_1$ are families of inputs that are consistent with the path from the root to v, then the subtree $T(v)$ that is rooted at v, separates A and B.

Definition 3.2 *Let $I \subseteq \binom{X}{k-1}$ be a family of $(k-1)$-sets.*

1. *Define $X(I) = \{x \in X \mid x \in U \text{ for some } U \in I\}$. In words, $X(I)$ is the actual set of elements (variables) that appear in the subsets of I.*
2. *We say that I is* **"scattered"** *if for every $x \in X(I)$, $\frac{|\{U \in I| \quad x \in U\}|}{|I|} < \frac{1}{2}$*

Definition 3.3 *Let $I_0 \subseteq \binom{X}{k-1}$ and $I_1 \subseteq \binom{X}{k}$ be two families of subsets. We say that the pair (I_0, I_1) is* **"increasing"** *if $\forall U \in I_0, \forall x \in X(I_0) \setminus U,\ \ U \cup \{x\} \in I_1$. In words, (I_0, I_1) is increasing if for every subset U in I_0, and every variable x not in U, $U \cup \{x\}$ is a member of I_1.*

Fact 3.1 *If (I_0, I_1) is increasing and $|I_0| \geq 2$ then $I_1 \neq \emptyset$.*

We state and prove now our main Theorem.

Theorem 3.2 *Let $8 \leq 4k \leq |X|$, and let $I_0 \subseteq \binom{X}{k-1}$,$I_1 \subseteq \binom{X}{k}$ be two families such that (I_0, I_1) is increasing, then $DT_{(\wedge,\vee)}(I_0, I_1) \geq \log_4 \frac{|I_0|^2}{\binom{|X|}{k-1}}$.*

Proof: The proof is by induction on k. The base is for $k = 2$ and it will follow as a special case of the general case. Assume that the theorem is true for $k' < k$ and that T is a $(\wedge, \vee)$-decision tree that separates I_0, I_1.

Each step, T can either "ask" the AND or the OR of a set of variables S. The adversary will choose the answer (the out-going edge in T) that maximizes the number of inputs of I_0 that are still consistent with it. We will have to make sure that as long as there are consistent inputs from I_0 then there is at least a consistent input from I_1 too. As long as we will take care of this fact the current node cannot be the end of the path.

There are two main cases:

Assume first that I_0 is scattered. The adversary will follow along the path of T (that is, she will provide answers) so that the set of consistent inputs from I_0 will be maximized. Let I_0^t, I_1^t be the families of consistent inputs after the t-th query. Let $X_t = X(I_0^t)$ be the set of variables at that step ($X_0 = X$). The adversary will keep on choosing the answer to the next query that maximizes the size of the consistent subset of I_0. She will do so as long as I_0^t is scattered while keeping the 'increasing' property. By Fact 3.1 this guarantees that as long as $(I_0^t \geq 2)$ then $I_1^t \neq \emptyset$. We start with (I_0, I_1) that is increasing and with I_0 that is scattered, we will assume inductively (on t) that (I_0^{t-1}, I_1^{t-1}) is increasing and that I_0^{t-1} is scattered and $|I_0^{t-1}| \geq 3$.

1. Assume that the t-th query is OR of a subset, $S \subseteq X_{t-1}$. Assume moreover, that the answer '0' maximizes the number of inputs of I_0^{t-1} that are consistent with it. Thus the new families are $I_0^t = \left\{U \in I_0^{t-1} | U \cap S = \emptyset\right\}$ $I_1^t = \left\{U \in I_1^{t-1} | U \cap S = \emptyset\right\}$ and $X_t = X_{t-1} \setminus S$.
   It is easy to see that as long as $|I_0^t| \geq 2$, then (I_0^t, I_1^t) is increasing (by the assumption that (I_0^{t-1}, I_1^{t-1}) is). We also have that $|I_0^t| \geq \frac{|I_0^{t-1}|}{2}$.
   In the case that the answer '1' maximizes the consistent input set, the two new families are $I_0^t = \left\{U \in I_0^{t-1} | U \cap S \neq \emptyset\right\}$ and $I_1^t = \left\{U \in I_1^{t-1} | U \cap S \neq \emptyset\right\}$. It can be verified that in this case too, $|I_0^t| \geq \frac{|I_0^{t-1}|}{2}$ and the pair of families is increasing.
2. Assume now that the t-th query is $\wedge$ of the variables of a subset $S \subseteq X_{t-1}$. In this case the adversary chooses an arbitrary $x \in S$, set it to 0 and gives a '0' answer. In addition she also gives the additional information (for free) that $x = 0$. Thus the consistent families are $I_0^t = \left\{U \in I_0^{t-1} | x \notin U\right\}$ and $I_1^t = \left\{U \in I_1^{t-1} | x \notin U\right\}$. As I_0^{t-1} is scattered then $|I_0^t| \geq \frac{|I_0^{t-1}|}{2}$. It is also easy to see that (I_0^t, I_1^t) is increasing.

What we have shown above is that if we start from two families, $I_0 \subseteq \binom{X}{k-1}$ and $I_1 \subseteq \binom{X}{k}$ that are: (a) separated by T, (b) increasing, (c) I_0 is scattered and (d) $|I_0| \geq 3$, then the adversary can keep choosing an answer to the next question so that: (i) The new families $I_0^t \subseteq \binom{X(I_0^t)}{k-1}$, $I_1^t \subseteq \binom{X(I_0^t)}{k}$ are consistent with the current node of T, (ii) (I_0^t, I_1^t) is increasing, and (iii) $|I_0^t| \geq |I_0^{t-1}|/2$. Thus, as noted before, the current node cannot be the end of the path as long as $|I_0^t| \geq 2$. Hence, upon termination either $t \geq \log_2 |I_0| = \log_4 |I_0|^2 \geq \log_4 \frac{|I_0|^2}{\binom{n}{k-1}}$. Or I_0^t becomes un-scattered.

In the later case we have that $|I_0^t| \geq 2^{-t}|I_0|$. Furthermore, we have seen before that in this case the subtree rooted at the current node must separate I_0^t, I_1^t. As I_0^t is not scattered, there is $x \in X(I_0^t)$ such that $|I_0(x)| \geq |I_0^t|/2$, where $I_0(x) = \{U \in I_0^t| \ \ x \in U\}$. Let $I_1(x) = \{V \in I_1^t| \ \ x \in V\}$.

Clearly the decision tree T separates $I_0(x), I_1(x)$ as partial families of I_0, I_1.

Let $I_0' = \{U \setminus \{x\}| \ \ U \in I_0(x)\}$, $I_1' = \{V \setminus \{x\}| \ \ V \in I_1\}$ and let $X' = X \setminus \{x\}$. We claim that there is a tree T' of depth (and size) at most the depth (size) of the tree rooted at the current node of T that separates I_0' and I_1'. Indeed the decision whether an input set A is in I_0' or I_1' is equivalent to the decision whether $(A \cup \{x\})$ is in $I_0(x)$ or $I_1(x)$. Thus in fact T' can be taken as the tree rooted at the current node of T where $\vee$ queries on sets that contain x have the '1' answer.

It can be easily checked that the families $I_0' \subseteq \binom{X'}{k-2}$ and $I_1' \subseteq \binom{X'}{k-1}$ are increasing. This follows directly from that fact that (I_0^t, I_1^t) is increasing. Thus, as $k' = k-1$ and $|X'| = |X|-1 \geq 4k'$ we can apply the induction hypothesis and deduce that the depth of T', $d(T')$ satisfies $d(T') \geq \ \log_4 \frac{|I_0'|^2}{\binom{|X'|}{k-2}}$

Let $n = |X|$, using the fact that $|I_0'| = |I_0(x)| \ \geq \ \frac{|I_0^t|}{2}$ and $n \ \geq \ 4k$ we have:

$$\frac{|I_0'|^2}{\binom{|X'|}{k-2}} \geq \frac{\frac{1}{4}|I_0^t|^2}{\binom{n-1}{k-2}} = \frac{|I_0^t|^2\binom{n}{k-1}}{4\binom{n}{k-1}\binom{n-1}{k-2}} = \frac{|I_0^t|^2 n}{\binom{n}{k-1}\cdot 4(k-1)} \geq \frac{|I_0^t|^2}{\binom{n}{k-1}}$$

Thus the total number of queries is at least

$$t + \log_4 \frac{|I_0'|^2}{\binom{|X'|}{k-2}} \geq t + \log_4 \frac{|I_0^t|^2}{\binom{n}{k-1}} \geq t + \log_4 \frac{4^{-t}|I_0|^2}{\binom{n}{k-1}} \geq \log_4 \frac{|I_0|^2}{\binom{n}{k-1}}$$

This completes the general case of the induction argument. To complete the proof we should proof the theorem for $k = 2$. Note however, that if $k = 2$ then as long as $|I_0| \geq 3$, I_0 is always scattered. Thus the the case $k = 2$ follows directly from the general case without relying on the induction hypothesis (as the induction hypothesis was only used in the case that I_0 becomes un-scattered), and the whole proof is completed $\Box$.

We can deduce now Theorem 3.1.

Proof of Theorem 3.1 We might assume, by symmetry that $k \leq |X|/2 \ = n/2$. We note that a straightforward lower bound of $k-1$ is trivial as an adversary can commit itself to set a variable to '1' for each OR query and to '0' for each AND query. For $k > \frac{n}{4}$ this is good enough. For $n \geq 4k$ one can apply Theorem 3.2 to the families $I_0 = \binom{X}{k-1}$ and $I_1 = \binom{X}{k}$ $\Box$.

Corollary 3.1 *Let* $I_0 \subseteq \binom{n}{k-1}$, $I_1 \subseteq \binom{n}{k}$ *and assume that* (I_0, I_1) *is increasing. Let* $g : \{0,1\}^n \to \{0,1\}$ *be a Boolean function and assume that for every* $x \in I_1$, $g(x) = 1$ *and that for every* $x \in I_0$, $g(x) = 0$, *then* $DT_{\vee,\wedge}(g) = \Omega(k \log \frac{n}{k})$. $\Box$

In particular, for every function that is fixed to 0 (1) on its $(k-1)$-slice, and that is fixed to 1 (0) on its k-slice, the corollary is applicable with the above bound. Thus, one may note that this corollary implies a bound on every symmetric function, e.g. parity etc. (although for parity this bound is trivial.

3.3 A lower bound on the 'direct sum' of T_2^n

For $j = 1, ..., k$ let $X_j = \{x_{i,j} \mid 1 \leq i \leq n\}$ be a set of Boolean variables and let $f_j, \;\; j = 1, ..., k$ be the threshold-2 function on the set $\{x_{i,j} \mid \;\; 1 \leq i \leq n\}$. We want to determine the complexity of an $(\wedge, \vee)$-decision tree that decides simultaneously the value of all the k Boolean functions. In such a decision tree every leaf would be labeled by a k-string that corresponds to the output of the k functions. We have proved in the previous subsection that $DT_{(\wedge,\vee)}(T_2^n) = \Omega(\log n)$. However, there is no reason why a decision tree would not query subsets containing variables of more then one of the functions. Thus one cannot deduce a lower bound of $\Omega(k \log n)$ from the $\Omega(\log n)$ lower bound for T_2^n. However, such a result can be obtained using Theorem 3.2.

Theorem 3.3 *The complexity of a $(\vee, \wedge)$-decision tree that simultaneously decides the value of k copies of T_2^n is $\Omega(k \log n)$.*

proof: Let $X = \bigcup_{j=1}^{k} X_j$. In our case, a (global) input to the problem is an assignment to kn Boolean variables and thus corresponds to a subset of X. Let T be a $(\vee, \wedge)$-decision tree for the above problem. Consider the following families of inputs: $I_0 = \{S \subseteq X \mid \; |S \cap X_j| = 1, \; 1 \leq j \leq k\}$, and $I_1 = \{S \subseteq X \mid \; \exists j |S \cap X_j| = 2, \;\; \textit{and } r \neq j \rightarrow |S \cap X_r| = 1\}$.

Clearly every input of I_0 is a '0' input for all of the functions while each input of I_1 is '1' for one of the functions. Hence, two inputs $U \in I_0$ and $V \in I_1$ cannot arrive to the same leaf of T. Thus, we see that by relabeling the leaves of T, T separates the families I_0, I_1. Moreover, one can easily check that (I_0, I_1) is increasing, thus by applying the result of Theorem 3.2 we get that $d(T) \geq \log_4 \frac{n^{2k}}{\binom{nk}{k}} = \Omega(k \log n) \;\; \Box$.

4 Lower bounds on the size

So far we have considered the depth of the tree as our complexity measure. We turn now to the size complexity. The size of a decision tree is the number of leaves it has. Obviously a lower bound on the size of a decision tree implies a (logarithmic) lower bound on its depth. Thus, a question that arises in view of the previous sections, is weather the depth bounds do not follow from bounds on the size, and if not what bounds can one prove on the size.

We first start with a general discussion.

A Σ_3 Boolean circuit is a depth-three unbounded fan-in circuit with $\vee$ as the output gate. In other words, a Σ_3 circuit is $\vee_{i=1}^{p} \wedge_{j=1}^{q} \vee_{\ell=1}^{r} t_{i,j,\ell}$ where $t_{i,j,\ell} \in \{x_1, ..., x_n\} \cup \{\bar{x}_1, ..., \bar{x}_n\}$. A Π_3 Boolean circuit is defined similarly except that its output gate is $\wedge$. The size of a circuit is the total number of gates in it. (For exact Boolean circuits terminology see [9]).

For the classical decision tree model, it is known that a function whose decision tree size is s has a DNF and a CNF of size bounded by $O(s)$ (a DNF is a depth two unbounded fan-in Boolean circuit with OR as the top gate, a CNF is such a circuit with an AND gate at the top). What is the counterpart here? It is easy to see that for the $(\wedge, \vee)$-decision tree, a decision tree of size s and depth d implies that the function has simultaneous Σ_3 and Π_3 circuits of size $O(ds)$ and 'small' fan-in in the second layer (bounded by $O(n)$). This is just by writing the function as $\vee$ over all the '1' leaves in the decision tree, where each leaf is defined by the path to it which is a conjunction of at most d ANDs or ORs (or their negation). This results in a Σ_3 circuit. For Π_3 circuit a similar construction is done on the '0' leaves. Hence lower bounds on depth three circuits lead to lower bounds on the size of $(\wedge, \vee)$-decision trees. In particular, lower bounds of $2^{\Omega(\sqrt{n})}$ can be obtained directly for parity, majority etc. [10, 11]. In fact, by taking into account the limitation on the fanin in the second level, one can adapt the proof of [11] to obtain an $2^{\Omega(n^{1-\epsilon})}$, for any $\epsilon > 0$. We note that parity does have depth-three circuits (both Σ_3 and Π_3) of size $2^{O(\frac{n}{\log n})}$, and with fan-in bounded by n in the second level. Majority has such circuits of size $2^{O(\frac{n \log\log n}{\log n})}$. These constructions are best understood via the connection to a communication complexity game that was introduced by Karchmer and Wigderson [15], see also [10, 11] for more details on depth three circuits (though the full details are left to the reader).

The results above can give a sublinear lower bound for the depth of $(\vee, \wedge)$-decision trees for both parity and majority. We note however, that the true lower bound for parity and majority is at least n and $n/2 - 1$ respectively, due to a trivial adversary argument.

The situation is different in the case of T_k^n. Namely, we don't know of suitable lower bounds that would imply the lower bounds of section 3, or anything close to that. In fact, T_k^n does have relatively small Σ_3 and Π_3 circuits. In particular it has circuits of size $2^{O(k \log k + \log n)}$ and fan-in of the second level bounded by k for Σ_3, and by n^2 for Π_3 (note that for Π_3 the fan-in is not small enough to meet our requirements but comes quite to close to it). Such circuits can be constructed using perfect hashing families and will not be described here (for further details on the connection of Boolean formulae to communication complexity and to perfect hashing see [9, 12]).

The best lower bounds for general depth-three circuits for T_k^n that we are aware of are $e^{\Omega(\delta(k))} n \log n$ where $\delta(k) = \sqrt{\frac{k}{\log k}}$, and $k \leq \log\log n$, due to J. Radhakrishnan [12]. For $k \leq n/2$ a lower bound of $e^{\Omega(\sqrt{k})} n$ can also be proven [12]. These bounds imply the same bounds for the size of $(\vee, \wedge)$-decision trees.

We present here a direct proof of a (weak) lower bound on the $(\vee, \wedge)$-decision tree namely:

Theorem 4.1 *The size of any $(\vee, \wedge)$-decision tree for T_k^n is $\Omega(n^{2-\frac{2}{k+1}}/k^3)$.*

Note that for $k = o(\log^2 n)$ this is better from what is implied by the bounds in [12].

We conjecture that the right lower bound on the size of a $(\vee, \wedge)$-decision tree for T_k^n should be $\Omega(\binom{n}{k-1}^{\epsilon})$ for some $\epsilon \leq 1$.

Proof: (of Theorem 4.1) Let T be an $(\vee, \wedge)$-tree for T_k^n. Assume that $A, B \in \binom{[n]}{k-1}$ are two '0'-inputs that arrive both to the same leaf v of T. If there are only $\vee$ queries along the path leading to v, then it can be easily seen that $A \cup B$ would also reach v which would be a contradiction. Thus there must be a $\wedge$ query along the way. Moreover, in order for this query to separate between $\{A, B\}$ on one hand and $A \cup B$ on the other hand, it should be to a set $S \subseteq A \cup B$ so that it would get a '0' answer for both A, B and a '1' answer to $A \cup B$. In particular $2 \leq |S| \leq 2k - 2$.

Assume, for the contrary that T has at most $s < (n^{2-\frac{2}{k+1}})/8k^3$ AND queries. Let $G = ([n], E)$

be the following graph on n vertices; $(i,j) \in E$ iff there is a $\wedge$ query to a set U so that $|U| \leq 2k-2$ and $i,j \in U$. As any such set U give rise to at most $2k^2$ edges, $|E| \leq (n^{2-\frac{2}{k+1}})/4k$. Thus by a theorem of Turan [3] there is an independent set $I \subset [n]$ of size $|I| \geq n^{\frac{2}{k+1}}k$. Now look at any two $(k-1)$ subsets $A, B \in \binom{I}{k-1}$, A, B cannot reach the same leaf in T as otherwise, as explained before, there should have been a $\wedge$ query on a set $U \subseteq A \cup B$. However if there was such a set then any $i,j \in U$ would be connected by an edge in G in contradiction to the fact that I is independent.

We conclude that each $A \in \binom{I}{k-1}$ arrives at a different leaf and thus the number of leaves is at least $\binom{|I|}{k-1} \geq \binom{n^{\frac{2}{k+1}}k}{k-1} \geq n^{2-\frac{2}{k+1}}$ $\square$.

We note here that our method, as it is, cannot give a bound better then $\Omega(n^2)$ however, suitable independence definition along with a Turan-like theorem for hypergraphs might do. Note also that in the proof above, $A \cup B$ can be replaced by any set $C \subseteq A \cup B$ and so that $C \cap (A \triangle B) \neq \emptyset$. In particular C could be of size k. Thus we get the following corollary.

Corollary 4.1 *Let $g : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function that is fixed to 0 (1) on the $(k-1)$-slice and 1 (0) on the k-slice, then the $(\vee, \wedge)$-decision tree for g is of size $\Omega(n^{2-\frac{2}{k+1}}/k^3)$.*

5 Randomized and Non-deterministic Complexity

So far we treated only the deterministic complexity. As in the classical model, one may (and should..) define randomized and non-deterministic complexity, and determine their relations.

5.1 Randomized Complexity

Intuitively, a randomized Boolean threshold decision tree for a Boolean function $g : \{0,1\}^n \rightarrow \{0,1\}$ is a decision tree where the next query might be random. Two different version are usually defined. In the 'Las-Vegas' version (or 0-error), The tree is not allowed to make any mistakes. The other version is the 'Monte-Carlo' version where the tree is allowed to have an error probability of a predetermined size ϵ. In both cases, the number of queries (the path length) becomes a random variable. The complexity of the decision tree is the expected number of queries (for the worst case input). For formal definition (of randomized trees for the standard decision tree model) one is referred to [18].

In a randomized algorithm, it is helpful to think of all random choices as being made in advance. Thus the algorithm along with a particular set of choices becomes a deterministic algorithm. In this way, a randomized algorithm is no more then a collection of deterministic algorithms with a probability distribution on it. Formally this can be defined,

Definition 5.1 *A randomized, 0-error, Boolean threshold decision tree for a Boolean function $g : \{0,1\}^n \rightarrow \{0,1\}$ is a collection of deterministic threshold decision trees for g along with a probability measure on them. The complexity is defined as the expected depth (according to the given probability).*

As usual the **randomized complexity** of g is defined to be the complexity of the optimal randomized decision tree for g and is denoted by $R_0(g)$ (The subscript 0, is to denote the fact that this is the complexity of a 0-error tree). Similarly a randomized $\wedge$, and $(\wedge, \vee)$-decision tree is defined. We omit the query types from the notation.

A 'Monte-Carlo' decision tree is defined similarly except that each deterministic tree in the collection may fail to give the right answer. The only requirement is that for each input, the probability of error is bounded by ϵ. In this case the complexity of g is denoted by $R_\epsilon(g)$.

In the classical model it was shown by [18] that $R_0(g)$ and $R_\epsilon(g)$ are polynomially related even for constant ϵ. Here the situation is quite different; $R_0(g)$ and $R_\epsilon(g)$ can be arbitrary apart, as demonstrated by the following example:

Along the sequel we consider $\vee$-decision trees. The results that we prove are true for $(\wedge, \vee)$-decision tree as well. However, the proofs are more involved and will not be presented here.

Claim 5.1 $R_0(T_2^n) = \Omega(\log n)$

Claim 5.2 $R_\epsilon(T_2^n) = O(\log \frac{1}{\epsilon})$. *In particular for* $\epsilon = O(1)$, $\quad R_\epsilon(T_2^n) = O(1)$

Proof (of claim 5.1): Following Yao, [23], to prove a lower bound on $R_0(g)$ it is enough to set a probability distribution on inputs and to bound the expected number of queries of any deterministic tree (expectation here is according to the probability on the inputs).

Consider the following probability on inputs: $1/(2n)$ for each input that contains exactly a single '1' and the rest $(1/2)$ is split arbitrarily between the '1' inputs of T_2^n. Let T be an optimal tree for that distribution.

We have shown (Fact 3 in section 3.1), that for any $\vee$-tree for T_k^n, no two $k-1$ sets (inputs) may arrive to the same leaf. Thus for $k = 2$, each of the n inputs that have exactly a single '1' arrives to a different leaf. Hence any tree for T_2^n has a set leaves L, with size at least n, to where the inputs with exactly a '1' must arrive. For our distribution, each leaf in L is obtained with probability $1/(2n)$. Let T' be the tree that is obtained by deleting all nodes form T except these that lead to a leaf in L. Clearly the expected leaf depth in T is at least half the expected depth in T' (as we deleted leaves of total probability $1/2$). But T' is a binary tree with n leaves and uniform distribution, thus its average leaf depth is $\Omega(\log n)$ $\quad\Box$.

Proof (of claim 5.2) Consider the following procedure repeated independently for r times: Pick a random subset $S \subseteq [n]$ uniformly. Query the OR of the variables in S and in $[n] \setminus S$.

If in one of the r trials the answer in both queries is '1' then decide on output '1'. Otherwise decide '0'.

Clearly if the decision is '1' then this is the true answer (no error). If the decision is '0', an error may occur if the input set (of size of at least two) is contained in S or in $[n] \setminus S$ for each of the rounds. However, that occurs with probability of at most $1/2$ in each of the r repetitions. Hence the error probability is bounded by 2^{-r}. Thus only $O(\log \frac{1}{\epsilon})$ queries are required to bring the error probability bellow ϵ $\Box$.

5.2 Non-Deterministic Complexity

Intuitively, a non-deterministic decision tree is the minimum set of queries that is required to convince a third party that the value of the function is determined when one is allowed to 'guess' (for the worst case input).

Formally, a non-deterministic threshold decision tree is a collection $\mathcal{N} = \{(C_1, v_1), (C_2, v_2)...\}$. Each C_i is a set of threshold queries and corresponding answers, each v_i is the value of the function for that 'guess' that is, $C_i = \{(S_{i1}, k_{i1}, b_{i1}), (S_{i2}, k_{i2}, b_{i2})...\}$ where $S_{ij} \subseteq [n]$ is the set being queried, k_{ij} is the type of query (e.g AND, OR ..) and $b_{ij} \in \{0,1\}$ is the expected answer, and $v_i \in \{0,1\}$ is the proposed value of the function. This collection must satisfy the following two conditions: Every input $x \in \{0,1\}^n$, x is consistent with at least one of the query sets C_i. (b) If $x \in \{0,1\}^n$ is consistent with a query C_i then $f(x) = v_i$. The complexity of the non-deterministic tree is $max|C_i|$ that is, the maximum number of queries among all sets in the collection. The complexity of a Boolean function $g : \{0,1\}^n \to \{0,1\}$ is the complexity of the optimal tree and is denoted by $N_T(g)$.

We define similarly, the complexity for $\wedge, \vee$ queries and denote it by $N_{\wedge,\vee}(g)$.

In the classical decision tree model it is folklore that the deterministic complexity is polynomially related to the non-deterministic complexity (see also [8, 21, 18]). Here, again we have a complete separation as will follow from the next claim.

Claim 5.3 *For every Boolean function $g : \{0,1\}^n \to \{0,1\}$, $N_{\wedge,\vee}(g) \leq 2$.*

Proof: The proof follows directly form the fact that for '1' inputs of g it is enough to examine a satisfied term in its DNF representation. This examination can be done by two queries (an AND query for the positive variables and an OR query of the negated variables). A dual argument is true for a '0'-input $\square$.

We note that both AND and OR queries are needed in the general case.

Comparing the claim above with the fact that for $(\wedge, \vee)$-queries $DT_{(\wedge,\vee)}(T_k^n) = \Omega(k \log(n/k))$ shows that $N(g)$ and $DT(g)$ may be unrelated.

6 Concluding Remarks

We want to mention here several open problems.

1. We don't know how to prove stronger lower bounds on the size of an $(\wedge, \vee)$-decision tree for T_k^n for general k. In particular we don't know the asymptotical size complexity for this problem. The related circuit complexity might be interesting on its own. Namely to improve the bounds of Radhakrishnan for the case of restricted fan-in on the second level, both for T_k^n and in general for other Boolean functions.
2. We don't know if the right lower bound for computing T_k^n with queries that contain T_2^n (in addition to $\wedge, \vee$) is also $\Omega(k \log(n/k))$. A trivial lower bound of $\Omega(k)$ is true and an

upper bound is given by the procedure in section 3.1. As indicated in the introduction this question is also related to a PRAM model that has a certain 'Ethernet' channel as its mean of communication.

3. We proved a lower bound on a certain direct-sum problem in section 3.3. Is this a general phenomenon in this model ? In other words, is it true for any Boolean function ? What about the direct-sum problem for the randomized version ? We currently, have no clue on these questions in this full generality.

References

[1] Y. Afek, G. Landau, B. Schieber, and M. Yung. The power of multimedia: Combining point-to-point and multiaccess network. *Information and Computation*, 84(1), Jan 1990.

[2] A. Aggarwal. Optimal bounds for finding maximum on array of processors with k global buses. *IEEE Trans. on Comp.*, 35:62–64, Jan 1986.

[3] N. Alon, J. H. Spencer, and P. Erdős. *The probabilistic method.* John Wiley, 1991.

[4] F. Meyer auf der Heide and H. Thien Pham. On the performance of networks with multiple busses. In *STACS92: Springer-Verlag*, pages 97–108, 1992.

[5] Y. Ben-Asher and I. Newman. Routing on meshes with buses. manuscript, 1994.

[6] M. Ben-Or. Lower bounds for algebraic computation trees. In *15th Annual ACM Symp. on the Theory of Computing*, pages 80–86, 1983.

[7] A. Bjorner, L. Lovasz, and A. Yao. Linear decision trees: volume estimates and topological bounds. In *24th Annual ACM Symp. on the Theory of Computing*, pages 170–177, 1992.

[8] M. Blum and R. Impagliazzo. Generic oracles and oracles classes. In *28th Annual IEEE Symp. on Foundations Of Computer Science*, pages 118–126, 1987.

[9] R.B. Boppana and M. Sipser. *The complexity of finite functions*, volume A , Algorithms and complexity, chapter 14, pages 757–803. Elsevier, 1990.

[10] J. Håstad. *Computational limitation for small depth circuits.* MIT Press, Cambridge, MA, 1986.

[11] J. Håstad and P. Pudlak S. Jukna. Top-down lower bounds for depth 3 circuits. In *34th Annual IEEE Symp. on Foundations of Computer Science*, pages 124–129, 1993.

[12] J. Radhakrishnan jaikumar. $\sigma\pi\sigma$ threshold formulas. *To appear in Combinatorica.*

[13] J. Kahn, M. Saks, and D. Sturtevant. A topological approach to evasiveness. *Combinatorica*, 4:297–306, 1984.

[14] M. Karchmer, R. Raz, and A. Wigderson. On proving super logarithmic depth lower bounds via the direct sum in communication complexity. In *6th IEEE Symp. on Structures in Complexity Theory*, pages 299–304, 1991.

[15] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super- logarithmic depth. In *20th Annual ACM Symp. on the Theory of Computing*, pages 539–550, 1988.

[16] V. King. Lower bounds on the complexity of graph properties. In *20th Annual ACM Symp. on the Theory of Computing*, pages 468–476, 1988.

[17] S. Moran, M. Snir, and U. Manber. Application of Ramsey's theorem to decision tree complexity. *J. ACM*, 32:938–949, 1985.

[18] N. Nisan. CREW PRAMS and decision trees. In *21th Annual ACM Symp. on the Theory of Computing*, pages 327–335, 1989.

[19] P. Ragde, I. Newman, and A. Wigderson. Perfect hashing and graph-entropy as applied to circuit and formulae complexity. In *6-th IEEE Symp on Structures in Complexity Theory*, pages 91–99, 1990.

[20] R. Rivest and S. Viullemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3:371–384, 1978.

[21] G. Tardos. Query complexity, or why is it difficult to separate $\mathrm{NP}^{\mathrm{A}} \cap \mathrm{coNP}^{\mathrm{A}}$ from P^{A} by a random oracle? *Combinatorica*, 9:385–392, 1989.

[22] U. Vishkin and A. Wigderson. Trade-offs between depth and width in parallel computation. *SIAM J. Comput.*, 14(2):303–314, 1985.

[23] A. C. Yao. Probabilistic computation, towards a unified measure of complexity. In *18th Annual IEEE Symp. on Foundations Of Computer Science*, pages 222–227, 1977.

[24] A. C. Yao. Lower bounds to randomized algorithms for graph properties. In *28th Annual IEEE Symp. on Foundations Of Computer Science*, pages 393–400, 1987.