Randomized Decision Tree Complexity for Read-Once Boolean Functions

Thesis submitted for the degree of Doctor of Philosophy by

Rafi Heiman
The Weizmann Institute of Science, June 1991

Advisors:
Prof. David Harel, The Weizmann Institute of Science
Prof. Avi Wigderson, The Hebrew University

Abstract

A decision tree for a Boolean function f is an algorithm that has to determine the value of f on an unknown input setting. It probes input variables in an adaptive manner until it has sufficient information to determine f's value. The randomized decision tree complexity of f, denoted $RC(f)$, is the expected number of probes performed for the worst case input by the best randomized algorithm. In 1986 Saks and Wigderson proved a lower bound on $RC(f)$ for every *read-once* function f. A function is called read-once if it is definable by a formula in which each variable appears exactly once. This lower bound is implicit in that it depends on the structure of the formula, and can be computed only in very special cases.

Our main results are three new lower bounds on $RC(f)$ for read-once functions f. First, we generalize the Saks-Wigderson lower bound to functions whose read-once formulae contain arbitrary fan-in AND/OR gates. Second, we prove a (non-recursive) $n^{0.51}$ lower bound where n is the number of input variables the function depends on. This proves that randomness helps less than nondeterminism as it is known that for the latter $n^{0.5}$ can be achieved. The third lower bound may be considered as a step towards the circuit complexity problem of separating TC^0 and NC^1. It is an $\frac{n}{2^d}$ lower bound for functions definable by read once formulae with *threshold* gates, where d is the formula's depth. The first two results appear in [HW91] and the third result appears in [HNW90].

In memory of Tsachi Tal

Acknowledgements

It was wonderful to have David Harel and Avi Wigderson as advisors and friends. I thank David in particular for his encouragement and continuous help while working in a field that was new for me, and after many changes of direction. Avi showed me how to think and be excited and obstinate, let me always be optimistic and calm, and lead me to the illusion we are equally collaborating. I thank him for these, as well as for his combination of an extremely nice and friendly personality on one hand, and fruitful thoughts and ideas on the other, due to which I enjoyed so much and succeeded, if indeed, during my PhD research.

Special thanks are to my other friends 'responsible' for the pleasant time I have had at school, Jonathan Dym, Uri Feige, Hugo Krawczyk, Oded Maler, Ilan Newman, Muli Safra, Avital Schrift, Danny Szoke, Moshe Tennenholtz and Irad Yavne, to name only some of them. I thank my teachers and the flexible staff at the Weizmann Institute for the lovely years there. I thank Faith Fich, Nati Linial, Mike Saks, Mario Szegedy, Eli Upfal and Edna Wigderson for sharing their time with me, working on 'my' problems. I thank Andy Yao who invited me to DIMACS, and many others at Princeton university, students and staff, for the pleasant last year.

Finally, I thank Yona, Eyal, Naama, Tirza and Jehoshua for their being, and for being as they are.

Contents

Chapter 1

Introduction

1.1 Motivation

1.1.1 Evaluating Games

Consider a program that plays some two-person zero-sum game, such as chess. In each step it faces some current position of the game and has to decide how to move next. It has in its 'mind' a tree representing some of the possible future scenarios of the game. Each future scenario may or may not end in a final position of the game. The root of the tree corresponds to the current position. The children of some node correspond to the game positions which are the results of the possible moves of the next player. Thus, a path in the tree describes a possible scenario of the game's development ended in some 'future' position that need not be a final position of the game, but is represented in the path's leaf. The tree can be quite general: some nodes may have more sons than others and some paths may be longer than others.

The program has to know its winning chances in the game positions represented by the leaves of the tree. Once it knows these values it can easily compute its winning chances in the current position - the root, and its best next move - the root's son with the highest winning chances. For each node that corresponds to the program's turn to move the winning chance is computed by taking the maximal winning chance among the node's sons, and for the opponent's nodes the minimum is computed.

We assume that the structure of the tree is known to the program or is easily computed. The min-max calculations are also very easy and are ignored. Only the values of the leaves, the winning chances, are difficult to compute. So the program's task is to compute, or to probe, as few leaves values as possible, provided that the leaves probed determine the result, the root's value, no matter what the values of the unprobed leaves are. To do so, the program is allowed to choose the order of leaves it probes adaptively: The choice of the 'next' leaf to probe may depend on the values revealed so far (as well as on the specific structure of the tree).

Is this possible? Can the program terminate before probing ALL leaves? It can be easily shown that for any such tree and any algorithm, there exists an 'input', a list of leaves values, in which the 'output', the root value, is not determined before all leaves are probed. This is true even if we assume that the tree is Boolean, that is, winning chances are estimated by just '1' or '0' values ('win' or 'loose'), and hence the MAX/MIN operations are replaced by the Boolean AND/OR ones, and the algorithm has only to decide whether the output is '0' or '1'.

On the other hand, if the algorithm can make RANDOM choices, it can sometimes save a lot. There are several known examples for which randomization enables probing in average only a negligible fraction of the leaves, while the output is guaranteed to be correctly computed. The best known example for this is quoted below.

The outline of this thesis is the study of the power of such randomized algorithms, following Pearl [Pea82], Tarsi [Tar83] and Saks and Wigderson [SW86]. Namely, we prove lower bounds on the number of leaves that are probed by any such randomized algorithm.

1.1.2 Simple Computational Model

Being interested in lower bounds and following Yao [Yao77] and Saks and Wigderson [SW86], we assume Boolean values. This makes the bounds obtained valid both for arbitrary real valued games and for Boolean valued functions. In other words, the model we study is that of (randomized) Boolean decision trees.

The model of Boolean decision trees is of theoretical interest in its own right. It is perhaps the simplest model of computation and is information theoretic in flavor: The algorithm is assumed to have perfect knowledge on the Boolean function that is computed and infinite power of computation. Only the input is not known and probing its bits values is charged.

The randomized complexity was studied by Yao [Yao77] in general, and by Yao [Yao87], King [Kin88] and Hajnal [Haj90] in the context of graph properties.

1.1.3 Read Once Functions

The set of functions we are interested in is that of *read once* Boolean functions, functions definable by formulae in which each input variable appears exactly once. [1]

Some combinatorial properties of this class were studied by Gurevich [Gur77], [Gur82]. As mentioned, this class was also studied in the context of evaluating games. Broder et. al. [BKRU91] studied parallel algorithms for evaluating games. Valiant [Val84] and Boppana [Bop89] studied this class in the context of amplifying approximating circuits. Finally, this class was studied by Angluin et. al. [AHK89] as subject to learning algorithms.

[1] We use the term 'read once functions' rather than 'game trees' or 'tree functions' in order to distinguish between this notion and the 'tree' in the notion of 'decision trees'.

1.1.4 Relations to Other Computational Models

The model of decision trees is central in the field of 'concrete complexity'. It simplifies the model of branching programs and captures their depth. Therefore, lower bounds in this model also bound the length of branching programs.

Decision trees were shown by Nisan [Nis89] to be fundamental for studying the complexity of Boolean functions in the CREW PRAM model as well. They were also used in characterizing and learning constant-depth polynomial-size circuits (AC^0) by Linial, Mansour and Nisan [LMN89].

Finally, our original motivation in studying randomized decision trees was a circuit complexity problem. As observed by Saks [Sak86] and explained below in detail, a lower bound in the randomized decision tree model implies a lower bound in circuit complexity. Namely, a lower bound on the randomized decision tree complexity of TC^0 functions, functions computable by constant depth polynomial size circuits with threshold gates, implies the separation of TC^0 from NC^1, the class of functions computable by polynomial size logarithmic depth circuits.

1.2 Definitions and Preliminaries

1.2.1 Boolean Decision Trees

A *deterministic decision tree* T is a binary tree labeled as follows. Each non-leaf node is labeled by some input-variable x_i. The two outgoing edges of such nodes are labeled, one by '1' and the other by '0'. Each leaf is labeled by an output value which is either '1' or '0'.

The *path* of T on the input-setting $\varepsilon = \varepsilon_1, \ldots, \varepsilon_n \in \{0,1\}^n$, termed $\mathrm{Path}_T(\varepsilon)$, is that (unique) path in the tree which starts at the root, and at each node, labeled x_i, follows the edge labeled ε_i. $\mathrm{Var}_T(\varepsilon)$ denotes the set of variables labeling the nodes of $\mathrm{Path}_T(\varepsilon)$. The *output* of T given ε, termed $\mathrm{Output}_T(\varepsilon)$, is the bit labeling the leaf of $\mathrm{Path}_T(\varepsilon)$. T *computes* the Boolean function f if $\mathrm{Output}_T(\varepsilon) = f(\varepsilon)$ for every ε.

The *time* consumed by T on input ε, termed $\mathrm{Time}_T(\varepsilon)$, is simply $|\mathrm{Var}_T(\varepsilon)|$. (Every variable is probed at most once in a path.) The *complexity* of T is the time consumed for a worst case input. The *deterministic decision tree complexity* of f, termed $DC(f)$, is the complexity of the best deterministic decision tree that computes f,

$$DC(f) = \min_T \max_\varepsilon \mathrm{Time}_T(\varepsilon). \qquad (1.1)$$

Clearly, $DC(f) \leq n$. f is called *evasive* if $DC(f) = n$, the maximum possible.

A *random decision tree* for f, RT, is a distribution over the deterministic decision trees for f. Given ε, a deterministic decision tree is chosen according to this distribution and is 'executed'. This makes the path and the time consumed random variables.

(Note that the output is always correct, though). The complexity of RT is the expected time (i.e., the expected number of variables it probes in order to determine the output) for a worst case input. The *randomized decision tree complexity* of f, termed $RC(f)$, is the complexity of the best randomized decision tree that computes f,

$$RC(f) = \min_{RT} \max_{\varepsilon} \mathrm{E}_{T \in RT}[\mathrm{Time}_T(\varepsilon)]. \tag{1.2}$$

Here E stands for expectation and $T \in RT$ stands for a random T chosen according to the distribution RT.

We can view a random decision tree also as a single decision tree that is allowed to choose the variables it probes at random, and is required to compute f with no error. Such a decision tree with random choices defines a distribution on the deterministic decision trees in the obvious way, and vice versa.

The following lemma of Yao [Yao77], which is based on the minimax theorem, expresses $RC(f)$ in terms of what he calls *the distributional complexity* of f.

Lemma 1.2.1 *(Yao)*

$$RC(f) = \max_{D} \min_{T} E_{\varepsilon \in D}[Time_T(\varepsilon)].$$

Here D ranges over all distributions on input settings of f, T ranges over all deterministic decision trees for f, and $\varepsilon \in D$ stands for a random input-setting ε chosen according to the distribution D.

The distributional complexity is a useful tool for proving lower bounds. One can guess some D and then prove a lower bound on $\min_T \mathrm{E}_{\varepsilon \in D}[\mathrm{Time}_T(\varepsilon)]$, which involves deterministic decision trees only.

The third notion we define is the *nondeterministic complexity* of f, $NC(f)$. For every input setting, ε, there is a *minimal proof*: a minimal subset of input variables whose values determine $f(\varepsilon)$, no matter what the values of the rest of the variables are. The size of such a subset is denoted $m(\varepsilon)$. The subset is called a *minterm* or *maxterm*, depending on whether it forces a 1-value of f or a 0-value. The *nondeterministic complexity* of f is defined as

$$NC(f) = \max_{\varepsilon} m(\varepsilon) \tag{1.3}$$

We also distinguish between 1-proofs and 0-proofs, and define the nondeterministic complexity for proving 1's of f, denoted $N_1(f)$, and the analogue for the 0's of f, denoted $N_0(f)$. Formally,

$$N_\nu(f) = \max_{\{\varepsilon : f(\varepsilon) = \nu\}} m(\varepsilon), \quad \text{for } \nu = 0, 1.$$

The following lemma states that the gap between any two of those complexity measures (deterministic, randomized or nondeterministic) is at most quadratic.

Lemma 1.2.2 *(Folklore) Every function f depending on n variables satisfies*

$$n \geq DC(f) \geq RC(f) \geq NC(f) = \max\{N_1(f), N_0(f)\} \geq \sqrt{DC(f)}$$

Proof: The first inequality is clear since a decision tree may never probe an input variable more than once. The second is clear because a deterministic decision tree is a special case of a randomized decision tree in which the distribution is concentrated on one decision tree . The third follows from the fact that randomized decision trees never err: Consider an input whose minimal proof contains $NC(f)$ input variables. The path of any deterministic decision tree for this input contains some proof for that input and so its length is at least $NC(f)$. The average path length for any distribution over deterministic trees is therefore also at least $NC(f)$. The equality is clear by definition. For the last inequality it is sufficient to show $DC(f) \leq N_1(f) \cdot N_0(f)$. This is done by constructing a deterministic decision tree for f. The tree starts by probing the variables of a minimal minterm of f, hence at most $\leq N_1(f)$ variables. If all of them are consistent with the variables' values of the minterm f is determined, so this branch of the decision tree is short. Otherwise, in the restricted function N_0 decreases by at least 1 (since every minterm intersects all maxterms) and the proof follows by induction. $\square$

The class of monotone graph properties has been extensively studied with respect to decision tree complexity. Rivest and Viullemin [RV78] proved a linear lower bound on the deterministic complexity for this class. The Aanderaa-Karp-Rosenberg conjecture is that all monotone graph properties are evasive. This was proven for graphs of prime power order (number of nodes) by Kahn, Saks and Sturtevant [KSS84]. On the other hand, there are several monotone graph properties with square-root nondeterministic complexity.

As for the randomized complexity, a conjecture that a linear lower bound applies to this class even if we allow randomization, is attributed to Karp. This has been proven for a few special monotone graph properties, but the best general lower bound is $\Omega(n^{2/3})$ of Hajnal [Haj90] (improving on Yao [Yao87] and King [Kin88]).

1.2.2 Read-Once Functions

Example 1.2.3 *An example of a read-once function is the alternating AND/OR function, g. $g = g^{(d)}$ is defined for every depth $d = 1, 2, \ldots$ as the function computed by the full binary balanced formula (tree) on $n = 2^d$ input-variables with alternating levels of AND and OR gates. Formally,*

$$g^{(0)}(x_1) = x_1; \quad \textit{and}$$
$$g^{(d+1)}(x_1, \ldots, x_{2^{d+1}}) = g^{(d)}(x_1, \ldots, x_{2^d}) \diamondsuit g^{(d)}(x_{2^d+1}, \ldots, x_{2^{d+1}}).$$

where $\diamondsuit$ is AND for even d and is OR for odd d.

A *read once formula* is a formula in which each input variable appears exactly once. A Boolean function is called *read once* if it can be represented by a read once formula. The class of read once functions depends on the type of gates (the basis) we allow in the formula's nodes. We will be considering two bases. The first is the AND,OR,NOT basis (called the standard basis), and the second is the THRESHOLD,NOT basis (called the threshold basis). Here a *threshold gate* is denoted T_l^k for some $k > 1$, $1 \leq l \leq k$, and is the Boolean gate with k inputs which outputs '1' iff at least l of its inputs are '1'. These two bases determine the two families of read once Boolean functions which are called *read once AND/OR functions* and *read once threshold functions.* Every read-once AND/OR function is a read once threshold function since the threshold gates T_1^k and T_k^k are, respectively, the OR and AND gates of fan-in k.

The following lemma allows ignoring the presence of negation gates in all the contexts we are interested in here.

Lemma 1.2.4 *(Ignoring Negations) Let F be a read once formula over the standard or the threshold bases computing some Boolean function f. Then there exists another read once formula G over the same basis, computing a monotone Boolean function g, such that (1) G contains no negations, (2) The size (respectively, height) of G is at most the size (respectively, height) of F, and (3) $DC(g) = DC(f)$, $RC(g) = RC(f)$.*

Proof: First, the negation gates can be 'pushed' to be applied directly only to the variables. This doesn't change the function being computed, can only decrease the formula's depth and doesn't change the number of gates that are not negations. Next, every negated variable and the NOT gate applied to it can be replaced by a new input variable. This renaming introduces a new function, but it allows back and forth simulations of deterministic decision trees computing these functions in the obvious way. As a result, the deterministic and the randomized complexities do not change. $\square$

Another property of a read once function is that there exists a unique read-once formula representing the function in some sense. Precise definition and statement follow. This was proven for AND/OR functions by Gurevich [Gur77], [Gur82] (see also [KLN+] and [Mun89]), and for threshold functions by Newman (see [HNW90], Theorem 2).

Definition 1.2.5 *A read-once threshold formula is non-degenerate if no input of some T_1^k-gate (OR) is the output of some other $T_1^{k'}$-gate, and similarly, no input of a T_k^k-gate (AND) is the output of a $T_{k'}^{k'}$-gate.*

Theorem 1.2.6 *(uniqueness) Two non-degenerate read-once threshold formulae that compute the same Boolean function are identical.*

Consequently, we sometimes do not distinguish between a read once function and the formula that computes it.

Finally, it is easy to see that every read once threshold function is evasive.

be evaluated. This corresponds to $u_0(f) = u_0(g)+u_0(h)$. To find a 1 of f, a directional randomized decision tree would start by evaluating g with some probability p, or start by evaluating h with probability $1-p$. If $g=1$ and $h=0$ the time consumed would be $pu_1(g)+(1-p)(u_0(h)+u_1(g)) = u_1(g)+(1-p)u_0(h)$. Similarly, if $g=0$ and $h=1$ the time consumed would be $p(u_0(g)+u_1(h))+(1-p)u_1(h) = pu_0(g)+u_1(h)$. The term for $g=h=1$, $pu_1(g)+(1-p)u_1(h)$ is smaller. Therefore the time consumed for the worst case input is

$$\max\{u_1(g)+(1-p)u_0(h),\ pu_0(g)+u_1(h)\}.$$

Minimizing this maximum subject to the constraint $0 \le p \le 1$ yields the term $\lambda(u_1(g), u_0(g), u_1(h), u_0(h))$.

The deterministic decision tree complexity of the alternating AND/OR function is maximal, $DC(g^{(d)}) = n$, by the evasiveness lemma (Lemma 1.2.7). However the randomized decision tree complexity of this function is low, $RC(g^{(d)}) = O(n^\alpha)$ for $\alpha = \log_2(\frac{1+\sqrt{33}}{4}) = 0.753...$, as was calculated using this upper bound.

The Saks-Wigderson lower bound is the following. Naturally, the terms $R_0(f)$ and $R_1(f)$ are lower bounds for the randomized decision tree complexity of finding, respectively 0's and 1's of f.

Theorem 1.2.9 *[SW86] Let f be a read-once Boolean formula with $\wedge$-$\vee$ gates of fan-in 2. Then $RC(f) \ge R_*(f)$ where $R_*(f) = \max\{R_0(f), R_1(f)\}$, and $R_0(f)$ and $R_1(f)$ are given by the following recursion.*

$$\begin{array}{ll}
R_0(f) = R_1(f) = 1 & \textit{if } f \textit{ is a single variable;} \\
R_0(f) = R_0(g) + R_0(h), \ \textit{and} & \\
R_1(f) = \chi(R_1(g), R_0(g), R_1(h), R_0(h)) & \textit{if } f = g \vee h; \\
R_1(f) = R_1(g) + R_1(h), \ \textit{and} & \\
R_0(f) = \chi(R_0(g), R_1(g), R_0(h), R_1(h)) & \textit{if } f = g \wedge h.
\end{array}$$

Here $\chi(a,b,c,d) = \min\{a+d, b+c, \frac{ab+cd+bd}{b+d}\}$.

Using this lower-bound, it was shown in [SW86] that the alternating AND/OR function g satisfies also $RC(g) = \Omega(n^{0.753...})$. This improves the results of Snir [Sni85] and meets the upper bound mentioned above. Actually, the Saks-Wigderson bounds meet also for many other functions (see [SW86]).

1.3 Main Results

Our first result is the generalization of the Saks-Wigderson lower bound (Theorem 1.2.9) for read once AND/OR formulae with arbitrary fan-in gates ('wide' gates). One way to do this is by replacing each wide gate, e.g., $\wedge(x,y,z)$ by an equivalent binary formula, $((x \wedge y) \wedge z)$ and applying Theorem 1.2.9 to the resulting binary formula. The following theorem, to which Chapter 2 is devoted, gives a better lower bound.

Theorem 1.3.1 *(wide gates) Let f be a read-once Boolean formula with $\wedge$-$\vee$ gates of arbitrary fan-in. Then $RC(f) \geq R_*(f)$ where $R_*(f) = \max\{R_0(f), R_1(f)\}$, and $R_0(f)$ and $R_1(f)$ are given by the following recursion.*

$$\begin{array}{ll} R_0(f) = R_1(f) = 1 & \textit{if } f \textit{ is a single variable;} \\ R_0(f) = \sum_{i=1}^k R_0(g_i), \textit{ and} & \\ R_1(f) = \chi(R_1(g_1), R_0(g_1), R_1(g_2), R_0(g_2), ..., R_1(g_k), R_0(g_k)) & \textit{if } f = \vee_{i=1}^k g_i; \\ R_1(f) = \sum_{i=1}^k R_1(g_i), \textit{ and} & \\ R_0(f) = \chi(R_0(g_1), R_1(g_1), R_0(g_2), R_1(g_2), ..., R_0(g_k), R_1(g_k)) & \textit{if } f = \wedge_{i=1}^k g_i. \end{array}$$

Here $\chi(a_1, b_1, a_2, b_2, ..., a_k, b_k) =$

$$\min_{\emptyset \neq T = \{i_1,...,i_t\} \subseteq \{1,2,...,k\}} [\sum_{i \notin T} b_i + \phi(a_{i_1}, b_{i_1}, a_{i_2}, b_{i_2}, ..., a_{i_t}, b_{i_t})]$$

and $\phi(a_1, b_1, a_2, b_2, ..., a_t, b_t) = (\sum_{j=1}^t a_j b_j + \sum_{1 \leq j < h \leq t} b_j b_h) / \sum_{j=1}^t b_j$.

This bound is given in a recursive form that depends on the structure of the formula. We would like to have a non-recursive lower bound — a lower bound in terms of the number of variables the function depends on, n (which is the deterministic decision tree complexity of the function, recalling Lemma 1.2.7). By Lemma 1.2.2, $RC(f) \geq [DC(f)]^{0.5}$ for every Boolean function f (called an 0.5-exponent). On the other hand, the AND/OR function f is known to have $RC(f) = \Theta(DC(f)^{0.753...})$ (or an 0.753...-exponent). No better bounds on the exponents are known for arbitrary Boolean functions.

Dealing with the special class of Graph properties, the sequence of results by Yao, King and Hajnal led to $RC(f) \geq DC(f)^{2/3}$ (for every non-trivial monotone graph property, f). In Chapter 3 we prove the following 'super square-root' lower bound for the class of read once AND/OR functions.

Theorem 1.3.2 *(non-recursive l.b.) There exists a real number $\theta > 0.5$ s.t. for every read-once AND/OR function f, $RC(f) \geq n^\theta$, where n is the number of input variables f depends on. In particular, $\theta = 0.51$ satisfies this inequality.*

Interestingly, Broder and Upfal [BU91] proved an $\Omega(n^{0.66...})$ lower bound for a certain subclass of read-once fan-in two functions. This subclass contains exactly those functions for which the Saks-Wigderson upper and lower bounds match.

The result proved in Chapter 4 was motivated by a circuit complexity problem. It is whether TC^0, the class of Boolean functions computable by polynomial-size constant-depth circuits with threshold gates, is strictly contained in NC^1, the class of functions having polynomial-size logarithmic-depth circuits with fan-in 2 gates. This question naturally surfaced after $AC^0 \neq TC^0$ was resolved ([FSS84], [Ajt83] and their improvements [Yao85] [Has88]), and after the results about constant depth circuits with prime modulo gates were proved ([Raz87], [Smo87]). It has been under attack in the last few years. Two important steps were made in the direction of separating these classes. The first, by Hajnal et al. [HMP+87], separated depth-2 from depth-3 polynomial-size threshold circuits. The second, by Yao [Yao89], separated the monotone analogues of the classes TC^0 and NC^1.

In 1986 Saks suggested a bold approach for separating TC^0 from NC^1: Show that every function in TC^0 has high (say linear) randomized decision tree complexity (in terms of its deterministic complexity). This would suffice, as there are several examples of evasive functions in NC^1 with randomized complexity n^α for $\alpha < 1$. One of them is the AND/OR function, which has the largest known exponent α. The intuition behind this approach is that all these examples have large (logarithmic) depth. There is no function in TC^0 which is even believed to have $o(n)$ randomized decision tree complexity. And so the only known reason for having a large gap between the deterministic and the randomized decision tree complexities is due to large depth, which allows iterated savings when using randomization. In fact it is sufficient to show that every **evasive** TC^0 function has randomized decision tree complexity higher than that of the alternating AND/OR function. This approach reduces a lower bound in the circuit model to a lower bound in the information theoretical model of randomized decision trees. It is particularly original and intriguing, since the separation will be proved by showing that functions in the smaller class are harder (in the second model).

The following lower bound, to which Chapter 4 is devoted, can be considered as a first step towards implementing Saks' approach. It proves the desired lower bound for read-once TC^0 functions. It is naive to be optimistic just because every TC^0 function is a simple projection of a read-once TC^0 function; it is not clear what happens to decision tree complexity under projections. However, the proof of the lower bound reveals that, from the point of view of randomized decision trees, threshold gates are no more powerful than ANDs and ORs, which hints that this may be the right direction to pursue. It perhaps hints also that a proof for AC^0 (if found) might generalize to a proof for TC^0 by similar techniques.

Theorem 1.3.3 *(threshold gates) Let f be a Boolean function computed by a read-once threshold formula of depth d over n input-variables. Then $RC(f) \geq \frac{n}{2^d}$*

In Chapter 5 the notion of *minterms oriented decision tree* is defined. It is proved that $DC(f)$ can be achieved by a minterms oriented decision tree for every function f whose minterms are of size at most 2. For the proof we show some generalizations of Lemma 1.2.7 stating that the deterministic decision tree complexity measure is additive under "read-once compositions". An example of a non monotone function given by Fich [Fic91] shows that this property is not true for all Boolean functions. Whether this strong and surprising property is true for all *monotone* functions remains open. However, an example is given to show that the attempt of a straightforward inductive proof for this property must fail.

Another example given in Chapter 5 shows that the Saks-Wigderson bounds are not tight: A a simple read once function simultaneously shows the following.

(i) No directional randomized algorithm (in the sense defined above) is optimal for this function, and hence the Saks-Wigderson upper bound is higher than the actual randomized decision tree complexity of the function.
(ii) The Saks-Wigderson lower bound is lower than the actual randomized decision tree complexity of the function. In fact, it shows the non optimality of the Shrinking lemma, which is the basic lemma in the Saks-Wigderson lower-bound, and whose generalizations are the basic steps in our Theorems 1.3.1 and 1.3.3 and in the new lower bound of Santha [San91] for Monte Carlo decision trees.

Chapter 6 discusses some open problems and concludes this thesis.

Chapter 2

Lower Bound for Wide Gates

In this chapter we prove Theorem 1.3.1,
Theorem *(wide gates) Let f be a read-once Boolean formula with $\wedge$-$\vee$ gates of arbitrary fan-in. Then $RC(f) \geq R_*(f)$ where $R_*(f) = \max\{R_0(f), R_1(f)\}$ and $R_0(f)$, $R_1(f)$ are given by the following recursion.*

$$
\begin{array}{ll}
R_0(f) = R_1(f) = 1 & \textit{if } f \textit{ is a single variable;} \\
R_0(f) = \sum_{i=1}^k R_0(g_i), \textit{ and} & \\
R_1(f) = \chi(R_1(g_1), R_0(g_1), R_1(g_2), R_0(g_2), ..., R_1(g_k), R_0(g_k)) & \textit{if } f = \vee_{i=1}^k g_i; \\
R_1(f) = \sum_{i=1}^k R_1(g_i), \textit{ and} & \\
R_0(f) = \chi(R_0(g_1), R_1(g_1), R_0(g_2), R_1(g_2), ..., R_0(g_k), R_1(g_k)) & \textit{if } f = \wedge_{i=1}^k g_i.
\end{array}
$$

Here $\chi(a_1, b_1, a_2, b_2, ..., a_k, b_k) =$

$$\min_{\emptyset \neq T = \{i_1,...,i_t\} \subseteq \{1,2,...,k\}} [\sum_{i \notin T} b_i \; + \; \phi(a_{i_1}, b_{i_1}, a_{i_2}, b_{i_2}, ..., a_{i_t}, b_{i_t})]$$

and $\phi(a_1, b_1, a_2, b_2, ..., a_t, b_t) = (\sum_{j=1}^t a_j b_j + \sum_{1 \leq j < h \leq t} b_j b_h) / \sum_{j=1}^t b_j$.

2.1 Reducing the Theorem to the Shrinking Lemma

The proof mimics the proof of Theorem 1.2.9 [SW86] , and consists of a bottom-up induction, whose single step is called the shrinking lemma. The calculations in the proof of the shrinking lemma are done slightly differently, and this is what enables the generalization to wide gates. A top down induction for a similar problem, which generalizes Theorem 1.2.9 to Monte Carlo algorithms, was given by Santha [San91].

In the definitions of the time consumed by a randomized decision tree and the randomized decision tree complexity we assumed a unit cost for probing a variable. In order to carry out an induction argument, these notions have to be generalized, and defined relative to a *variables cost function*, $c = (c_0, c_1)$. The function c_ν : $\{x_1, ..., x_n\} \rightarrow \mathbb{R}$ assigns to each input variable x_i the cost (or the time consumed) for probing this variable in case its value is ν, $\nu \in \{0,1\}$. Given such c the time consumed by a deterministic decision tree T on a given input setting ε is

$$\mathrm{Time}_{c,T}(\varepsilon) = \sum_{x_i \in \mathrm{Path}_T(\varepsilon),\ \varepsilon(x_i)=0} c_0(x_i) + \sum_{x_i \in \mathrm{Path}_T(\varepsilon),\ \varepsilon(x_i)=1} c_1(x_i).$$

$DC(f,c)$ and $RC(f,c)$ denote the complexities relative to c and are defined similarly to the definitions of $DC(f)$ and $RC(f)$. Yao's lemma (Lemma 1.2.1) then becomes

$$RC(f,c) = \max_D \min_T \ \mathrm{E}_{\varepsilon \in D}[\ \mathrm{Time}_{c,T}(\varepsilon)]. \tag{2.1}$$

The wide gates theorem follows by applying the following ('shrinking') lemma inductively. We begin with the non degenerate formula for the read once function f given in the theorem and with unit variables cost, $c_\nu(x_i) = 1$ for every variable x_i and $\nu \in \{0,1\}$. The last application of the lemma yields the trivial formula consisting of a single variable, v, whose costs are exactly $c_0(v) = R_0(f)$ and $c_1(v) = R_1(f)$, and therefore the greater of them bounds $RC(f)$ from below. $\square$

2.2 The Shrinking Lemma

The shrinking lemma bounds the randomized decision tree complexity of a read-once formula by that of a smaller (shrunk) formula. The smaller formula is obtained by shrinking one gate of the original formula, say it is an OR gate, whose inputs are all input-variables, into a single variable denoted v. The cost of v depends on the costs of the variables entering the gate in the original formula. Roughly, the cost to probe v when its value is 0 is the sum of the costs to probe those variables since they all must be probed in order to find that $v = 0$. Similarly, the cost to probe v when its value is 1, is about half of this sum. The reason for this is that the worst case which causes v to be 1 is when all the variables are 0 except for a single variable which is 1, thus in average about half of the variables are probed before this single 1 is found, at which point the value of v is found.

Lemma 2.2.1 *(shrinking) Let F be a read-once threshold formula of depth $d > 0$ that computes a Boolean function f. Consider a gate $\diamondsuit$, $\diamondsuit \in \{\wedge, \vee\}$ whose children are all input variables. Denote these variables by $Y = \{y_1, ..., y_k\}$. Denote the rest of the variables by $X = \{x_1, ..., x_m\}$. (See Figure 2.1.) Let F' be the formula obtained from F by replacing the sub-formula $\diamondsuit(y_1, ..., y_k)$ by a single variable v (see Figure 2.2), and let f' be the function computed by F'. Let $c = (c_0, c_1)$ be a cost function for the $m+k$ variables of f, that is, $c_\nu : X \cup Y \to I\!R$ for $\nu = 0, 1$. Define a new pair of cost functions $c' = (c'_0, c'_1)$ for the $m+1$ variables of f' by*

$$\begin{array}{ll}
c'_\nu(x_i) = c_\nu(x_i) & \forall 1 \le i \le m,\ \nu = 0,1 \\
c'_0(v) = \sum_{i=1}^k c_0(y_i),\ \textit{and} & \\
c'_1(v) = \chi(c_1(y_1), c_0(y_1), c_1(y_2), c_0(y_2), ..., c_1(y_k), c_0(y_k)) & \textit{if } \diamondsuit = \vee; \\
c'_1(v) = \sum_{i=1}^k c_1(y_i),\ \ \textit{and} & \\
c'_0(v) = \chi(c_0(y_1), c_1(y_1), c_0(y_2), c_1(y_2), ..., c_0(y_k), c_1(y_k)) & \textit{if } \diamondsuit = \wedge.
\end{array}$$

where χ is defined in Theorem 1.3.1.
Then $RC(f',c') \leq RC(f,c)$.

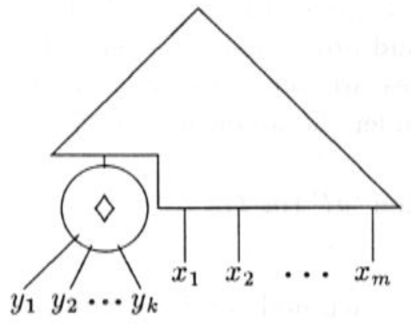


Figure 2.1: The given F

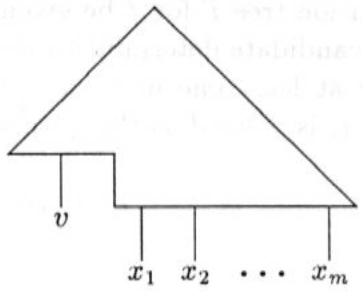


Figure 2.2: The shrunk F'

Proof: We prove the shrinking lemma for the case $\Diamond = \wedge$. The $\vee$-case is dual. First, we introduce some necessary notations.

Notations:

$[k]$ denotes the set $\{1, ..., k\}$.

$\{^A_a\}$ denotes the set of all subsets of a set A which have cardinality a.

For a (partial) setting $\theta : X \to \{0,1\}$,
θ^1 denotes the setting on $\{v\} \cup X$ which extends θ by assigning 1 to v, $\theta^1(v) = 1$;
θ^0 denotes the setting on $\{v\} \cup X$ which extends θ by assigning 0 to v, $\theta^0(v) = 0$;
$\theta_{\bar{1}}$ denotes the setting on $Y \cup X$ which extends θ by assigning 1 to y_j, $\forall j \in [k]$; and
$\theta_{\bar{0}_i}$ for some $i \in [k]$ denotes the setting on $Y \cup X$ which extends θ by assigning 0 to y_i and 1 to y_j, $\forall j \in [k]$, $j \neq i$.

$\Pr_D(E)$ denotes the probability of E, given a distribution D.

$c_\nu(U)$ denotes the total ν-cost of a subset U of input variables, $c_\nu(U) = \sum_{u \in U} c_\nu(u)$.

$c(U,\varepsilon)$ denotes the total cost of U given input-setting ε, $c(U,\varepsilon) = \sum_{u\in U} c_{\varepsilon(u)}(u)$.

U^T_ε denotes those variables in some subset U of input variables that are probed by T given (complete) input-setting ε, $U^T_\varepsilon = U \cap \text{Var}_T(\varepsilon)$.

$c(U^T_\varepsilon)$ denotes the probing cost of these variables, $c(U^T_\varepsilon) = c(U^T_\varepsilon, \varepsilon)$.

To show the lemma's conclusion, $RC(f',c') \leq RC(f,c)$, we use (2.1) and show that

$$(\forall D')\ (\exists D)\ (\forall T)\ (\exists T')\ \ \mathrm{E}_{\varepsilon' \in D'}[\ \mathrm{Time}_{c',T'}(\varepsilon')] \leq \ \mathrm{E}_{\varepsilon\in D}[\ \mathrm{Time}_{c,T}(\varepsilon)], \qquad (2.2)$$

where D (respectively D') is a distribution on the input-settings to f (respectively f'), and T (respectively T') is a deterministic decision tree for f (respectively f').

Let D' be given. Define a distribution D by assigning non-zero probability only to those inputs in which all the Y-variables are 1, except perhaps one of them as follows. For every X-setting $\theta : X \to \{0,1\}$ define

$$\Pr_D(\theta_{\bar{1}}) = \Pr_{D'}(\theta^1) \text{ and}$$

$$\Pr_D(\theta_{\bar{0}_i}) = p_i \cdot \Pr_{D'}(\theta^0)$$

where $p_i = c_1(y_i)/\sum_{j\in[k]} c_1(y_j)$. Note that all other extensions of θ to $Y \cup X$ have probability 0 under D.

Now, let a decision tree T for f be given. We do not define T' explicitly. Rather, we define a set of candidate deterministic decision trees and prove that the inequality in (2.2) holds for at least one of them. The candidates are the k decision trees, T_i, $i \in [k]$, where T_i is defined as the 'projection' of T under the following actions:

1. Each question 'y_i?' (in T) is replaced by the question 'v?' (in T_i).

2. For each $j \neq i$, T_i assumes that $y_j = 1$. Namely, for each node of T containing the question 'y_j?', T_i passes down this question to the 1 direction while deleting that node and the entire sub-tree under the 0 direction. (See Figure 2.3.)

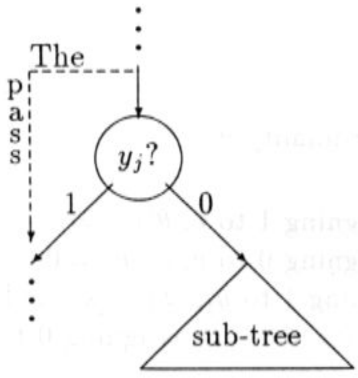


Figure 2.3: Assuming $y_j = 1$

We now show that the inequality in (2.2) holds for some candidate T_i. It suffices to show that a convex combination of the left hand sides of this inequality with the T_i's replacing T' is bounded by (the fixed) right hand side of the inequality. Namely, it suffices to show that

$$\sum_{i\in[k]} p_i \; \mathrm{E}_{\varepsilon'\in D'}[\, \mathrm{Time}_{c',T_i}(\varepsilon')] \leq \; \mathrm{E}_{\varepsilon\in D}[\, \mathrm{Time}_{c,T}(\varepsilon)]. \qquad (2.3)$$

First, we write the explicit terms for the two expectations above:

$$\mathrm{E}_{\varepsilon'\in D'}[\, \mathrm{Time}_{c',T_i}(\varepsilon')] \overset{\mathrm{def}}{=} \sum_{\theta:X\to\{0,1\}} [\Pr_{D'}(\theta^1)\cdot \mathrm{Time}_{c',T_i}(\theta^1) + \Pr_{D'}(\theta^0)\cdot \mathrm{Time}_{c',T_i}(\theta^0)]$$

$$\mathrm{E}_{\varepsilon\in D}[\, \mathrm{Time}_{c,T}(\varepsilon)] \overset{\mathrm{def}}{=} \sum_{\theta:X\to\{0,1\}} [\Pr_{D'}(\theta^1)\cdot \mathrm{Time}_{c,T}(\theta_{\bar{1}}) + \Pr_{D'}(\theta^0)\cdot \sum_{i\in[k]} p_i \cdot \mathrm{Time}_{c,T}(\theta_{\bar{0}_i})]$$

Inserting these terms to (2.3), we note that it is sufficient to show for each $\theta : X \to \{0,1\}$ that the following inequalities hold:

$$\sum_{i\in[k]} p_i \text{ Time}_{c',T_i}(\theta^1) \leq \text{ Time}_{c,T}(\theta_{\bar{1}}) \tag{2.4}$$

$$\sum_{i\in[k]} p_i \text{ Time}_{c',T_i}(\theta^0) \leq \sum_{i\in[k]} \text{ Time}_{c,T}(\theta_{\bar{0}_i}) \tag{2.5}$$

Next, we divide 'Time' to the costs of X, Y and v, and use the notation above. Inequalities (2.4) and (2.5) become (2.6) and (2.7), respectively.

$$\sum_{i\in[k]} p_i[c'(X^{T_i}_{\theta^1}) + c'_1(v)\cdot 1_{v\in \text{ Var}_{T_i}(\theta^1)}] \leq c(X^T_{\theta_{\bar{1}}}) + c(Y^T_{\theta_{\bar{1}}}) \tag{2.6}$$

$$\sum_{i\in[k]} p_i[c'(X^{T_i}_{\theta^0}) + c'_0(v)\cdot 1_{v\in \text{ Var}_{T_i}(\theta^0)}] \leq \sum_{i\in[k]} p_i[c(X^T_{\theta_{\bar{0}_i}}) + c(Y^T_{\theta_{\bar{0}_i}})] \tag{2.7}$$

By the definition of T_i, $\text{Path}_{T_i}(\theta^1)$ is the 'projection' of $\text{Path}_T(\theta_{\bar{1}})$. In particular, $X^{T_i}_{\theta^1} = X^T_{\theta_{\bar{1}}}$ and $v \in \text{ Var}_{T_i}(\theta^1)$ iff $y_i \in Y^T_{\theta_{\bar{1}}}$. Denote $Y^T_{\theta_{\bar{1}}}$ by $Z = \{z_1, z_2, ..., z_s\}$ and assume that the order these variables are probed by T given $\theta_{\bar{1}}$ is $(z_1, z_2, ..., z_s)$. Since c' and c are identical on X, and θ fixes the X-values, the X-costs cancel out and (2.6) becomes

$$\sum_{i\in[k]} [p_i c'_1(v)\cdot 1_{y_i\in Z}] \leq c(Z, \theta_{\bar{1}}) = c_1(Z).$$

This is true by the definitions of $c'_1(v)$ and the p_i-s.

Similarly, the X-costs in (2.7) also cancel out and that inequality reduces to

$$\sum_{t=1}^{s} p_{i(t)} c'_0(v) \leq \sum_{i\in[k]} p_i c(Y^T_{\theta_{\bar{0}_i}}) = \sum_{t=1}^{s} \frac{c_1(z_t)}{c_1(Y)}[\sum_{j=1}^{t-1} c_1(z_j) + c_0(z_t)] + \sum_{y_i\in Y\setminus Z} \frac{c_1(y_i)}{c_1(Y)} c_1(Z)$$

where $i(t)$ denotes that index i for which $y_i = z_t$. The equality is due to the facts that for $y_i = z_t \in Z$ we have $Y^T_{\theta_{\bar{i}}} = \{z_1, z_2, ..., z_t\}$ and $c(Y^T_{\theta_{\bar{i}}}) = \sum_{j=1}^{t-1} c_1(z_j) + c_0(z_t)$, and for $y_i \in Y \setminus Z$ we have $Y^T_{\theta_{\bar{i}}} = \{z_1, z_2, ..., z_s\}$ and $c(Y^T_{\theta_{\bar{i}}}) = \sum_{j=1}^{s} c_1(z_j) = c_1(Z)$.

Finally, the last inequality vanishes if $s = 0$, that is, Z is empty. Otherwise, by $\sum_{t=1}^{s} p_{i(t)} = \frac{c_1(Z)}{c_1(Y)}$, the requirement reduces to

$$\begin{aligned} c'_0(v) &\leq \tfrac{1}{c_1(Z)}[\textstyle\sum_{j<t\leq s} c_1(z_j)c_1(z_t) + \sum_{t\leq s} c_1(z_t)c_0(z_t)] + \sum_{y_i\in Y\setminus Z} c_1(y_i) \\ &= \phi(c_0(z_1), c_1(z_1), c_0(z_2), c_1(z_2)...c_0(z_s), c_1(z_s)) + \textstyle\sum_{y_i\in Y\setminus Z} c_1(y_i) \end{aligned}$$

where ϕ refers to Theorem 1.3.1. The definition of $c'_0(v)$ makes sure that this holds for every nonempty Z. $\square$

Chapter 3

Non-recursive Lower Bound

3.1 Introduction

In this chapter we prove the $n^{0.51}$ lower bound on the randomized decision tree complexity for read once AND/OR functions (Theorem 1.3.2). We call it non-recursive since this bound is not given in terms of the structure of a formula representing the function, rather it is given in terms of the number of input variables the function depends on. The proof combines the generalized recursive lower bound (Theorem 1.3.1) with restrictions arguments.

In Section 3.2 we point out that the Saks-Wigderson lower bound (Theorem 1.2.9) cannot by itself imply the non-recursive theorem. We give there an example of an infinite family $\{F_n\}$ of read once Boolean functions, where F_n depends on n variables. Theorem 3.2.4 shows that this family satisfies $R_*(F_n) = o(n^\theta)$ for every $\theta > \frac{1}{2}$. However, a simple restriction consideration shows that F_n has high randomized decision tree complexity (Theorem 3.2.5). This suggests that the proof of Theorem 1.3.2 should involve something else in addition to the Saks-Wigderson recursive argument, perhaps restrictions.

It appears also that formulae with gates of fan-in greater than two should be considered in a more delicate way than simply replacing them by binary trees and then applying the Saks-Wigderson lower bound. In Section 3.3 we prove a more 'friendly' version of the generalized recursive lower bound, Theorem 3.3.1. This version is used in the proof of the non-recursive theorem.

Section 3.5 contains that part of the proof that involves restrictions and Section 3.6 contains the rest of the proof — some inequalities on real valued functions.

While it is clear that the proof can be pushed to give a lower bound better than 0.51, we remark in Section 3.7, that 0.58 is the best exponent one can hope for when using Theorem 3.3.1. The AND/OR tree function shows this. In fact, Broder and Upfal [BU91] have proved very recently that the randomized decision tree complexity of every *well balanced* read once AND/OR function is at least that of the AND/OR tree function, $1.5^{\log n} = n^{0.58\ldots}$. 'Well balanced' means that the Saks-Wigderson upper and lower bounds match for the function. The example in Section 3.2 and the cases where we need to use restrictions in Section 3.5 involve functions which are not well balanced in this sense.

Returning to one of the motivations mentioned in the introduction, we would like to state here an immediate corollary of the non-recursive theorem. It corresponds to the problem of evaluating two-person zero-sum game trees using α-β pruning [Pea82].

Imagine a chess program making a decision about the next move in a given board position. It develops a partial tree of possible moves alternating between it and its opponent. This results in new game positions at the leaves. Each such position is assigned a real value from which the value of the root can be computed (hence also the best move under this information). This is a generalization of read once function to real valued inputs and MIN/MAX gates.

The standard α-β pruning algorithm evaluates this tree by evaluating *some* of the leaf positions, and is charged according to the number of leaves evaluated. So its deterministic and randomized versions are analogous to deterministic and randomized decision tree. Theorem 1.3.2 has as immediate consequence the following lower bound on the number of positions that have to be evaluated in any two-person game tree. Non trivial lower bounds for this problem were previously known only for very special game trees [Tar83], [SW86].

Corollary 3.1.1 *Any randomized α-β pruning algorithm computing the value of any two-person game tree with n final positions evaluates at least $n^{0.51}$ positions for the worst case values of leaves.*

3.2 Why Restrictions?

We define a family $\{F_m\}$ of read once AND/OR formulae for which the Saks-Wigderson lower bound gives 'only' $n^{0.5+o(1)}$. The idea is that the formulae are very imbalanced: Each of their OR gates has only a single variable as a child while a 'heavy' formula as the other child. The definition is recursive. Given F_m we AND it with another copy of F_m and OR the result with a single new variable. Then we again AND the result with another copy of F_m and OR with another new variable. We repeat these 'ANDing' and 'ORing' m times, so as to have $m+1$ copies of F_m in the resulting formula, denoted F_{m+1} (See Figure 3.1).

The analysis deals with the terms R_0 and R_1 which are defined in the Saks-Wigderson lower bound, Theorem 1.2.9. Intuitively, the non-balancing of any AND gate causes its R_0 to be relatively small. Then 'ORing' this with a single variable causes the new R_0 and R_1 to be equal, and only slightly higher than the value of R_0 at the AND gate, no matter how high R_1 at that AND was.

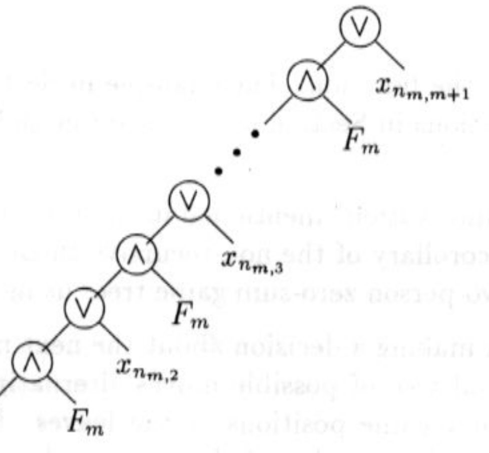


Figure 3.1: The formula F_{m+1}

Following is a formal definition of these formulae. First we define $F_{m,i}$ the prefix of 'length' i of F_{m+1}, which is used in the analysis.

Definition 3.2.1 *For $m = 1, 2, \ldots$ and $1 \le i \le m+1$ define a number $n_{m,i}$ and a Boolean formula $F_{m,i}(x_1, \ldots, x_{n_{m,i}})$ depending on $n_{m,i}$ input variables as follows.*

For $m = i = 1$ let

$$n_{1,1} = 1, \quad F_{1,1}(x_1) = x_1.$$

For $m \ge 2$ and $i = 1$ let

$$n_{m,1} = n_{m-1,m}, \quad F_{m,1}(x_1 \ldots x_{n_{m,1}}) = F_{m-1,m}(x_1 \ldots x_{n_{m-1,m}}),$$

For $m \ge 1$ and $2 \le i \le m+1$ let

$$n_{m,i} = n_{m,i-1} + n_{m,1} + 1, \quad F_{m,i}(x_1 \ldots x_{n_{m,i}}) =$$
$$(F_{m,i-1}(x_1 \ldots x_{n_{m,i-1}}) \wedge F_{m,1}(x_{n_{m,i-1}+1} \ldots x_{n_{m,i-1}+n_{m,1}})) \vee x_{n_{m,i}}.$$

The next definition which is just renaming of the full length prefix $F_{m,m+1}$ as F_{m+1} concentrates on the formulae described above.

Definition 3.2.2 *For $m = 1, 2, \ldots$ define a subsequence $F_m(x_1, \ldots, x_{n_m})$ by $n_m = n_{m,1}$ and $F_m = F_{m,1}$.*

For example, $F_1 = x_1$, $F_2 = x_1 \wedge x_2 \vee x_3$ and
$F_3 = [(x_1 \wedge x_2 \vee x_3) \wedge (x_4 \wedge x_5 \vee x_6) \vee x_7] \wedge (x_8 \wedge x_9 \vee x_{10}) \vee x_{11}$.

We first calculate n_m, the number of input variables in F_m. n_m is the sum of the number of inputs which are children of AND gates, denoted A_m, and the number of inputs which are children of OR gates, denoted O_m. For the trivial formula F_1 which consists of $n_1 = 1$ input variable, we make the convention $A_1 = 1$ and $O_1 = 0$. We then have:

Proposition 3.2.3 *$A_m = m!$, $O_m = m! - 1$ and hence $n_m = 2m! - 1$.*

Proof: A_m satisfies the recursion $A_{m+1} = (m+1)A_m$, hence $A_m = m!$. $A_1 - O_1 = 1$ and by induction, $A_{m+1} - O_{m+1} = (m+1)(A_m - O_m) - m = 1$. The proposition follows. $\square$

The following theorem states that for the functions $\{F_m(x_1 ... x_{n_m})\}_{m=1,2,...}$ the Saks-Wigderson lower bound gives 'only' $(n_m)^{0.5+o(1)}$. R_* refers to that of Theorem 1.2.9. We use the 'big-oh' and the 'small-oh' notations. In contrast, Theorem 3.2.5 states that the 'real' randomized decision tree complexity of F_m is 'high' — linear in the number of input variables.

Theorem 3.2.4 $R_*(F_m) = O(4^m (m!)^{0.5}) = (m!)^{0.5+o(1)} = (n_m)^{0.5+o(1)}$.

Theorem 3.2.5 *For every m, $RC(F_m) > \frac{n_m}{2}$.*

Proof of Theorem 3.2.5: By Lemma 1.2.2, we show that the nondeterministic complexity is at least $\frac{n_m}{2}$. Restrict to 0 all the input variables that enter OR-gates. The restricted resulting function is a simple AND. By Proposition 3.2.3 this AND depends on $A_m = \frac{n_m+1}{2}$ inputs. Hence F_m has a single maxterm of size greater than $\frac{n_m}{2}$. $\square$

Proof of Theorem 3.2.4: The second and last equalities are clear by definition and by Proposition 3.2.3. To show the first equality we show that $\frac{R_*(F_{m+1})}{R_*(F_m)} \leq 4\sqrt{m+1}$ for $m \geq 3$. The following sequences help bounding this ratio.

For each m define a sequence of natural numbers $\{b_{m,i}\}_{i=1}^{m+1}$ inductively: $b_{m,1} = 1$ and $b_{m,i+1} = b_{m,i} + \frac{1}{1+b_{m,i}} + \frac{1}{R_*(F_m)}$.

The following proposition asserts that $b_{m,m+1}$ is exactly the ratio we are interested in, $\frac{R_*(F_{m+1})}{R_*(F_m)}$. Part (i) is just to help carry the induction.

Proposition 3.2.6 *For $m \geq 3$ and every i,*
(i) $R_1(F_{m,i}) = R_0(F_{m,i})$, hence $R_*(F_{m,i})$ equals this value.
(ii) $\frac{R_*(F_{m,i})}{R_*(F_m)} = b_{m,i}$.

Proof: The proof is by induction on m and i (lexicographically ordered). For all m and $i = 1$ part (*ii*) is trivially true; its both sides are 1. To see that part (*i*) is true for $m = 3$, $i = 1$ we compute $R_*(F_3)$, using the definitions of R_0 and R_1 as given in Theorem 1.2.9. Hereafter $G_{m,i}$ denotes $F_{m,i} \wedge F_m$.

$$\begin{array}{llllll}
R_0(F_1) & = & 1, & R_1(F_1) & = & 1, \\
R_1(G_{1,1}) & = & 1+1=2, & R_0(G_{1,1}) & = & \min\{1+1, \frac{1+1+1}{2}\} = 1.5, \\
R_0(F_2) & = & 1+1.5=2.5, & R_1(F_2) & = & \min\{3, 2.5, \frac{3+1.5+1}{2.5}\} = 2.2, \\
R_1(G_{2,1}) & = & 2.2+2.2=4.4, & R_0(G_{2,1}) & = & \min\{2.5+2.2, \frac{2.5\cdot 2.2\cdot 2+2.2^2}{2\cdot 2.2}\} = 3.6, \\
R_0(F_{2,2}) & = & 3.6+1=4.6, & R_1(F_{2,2}) & = & \min\{3.6+1, 4.4+1, \frac{3.6\cdot 4.4+3.6+1}{3.6+1}\} = 4.443..., \\
R_1(G_{2,2}) & = & 6.643..., & R_0(G_{2,2}) & = & 5.376...,
\end{array}$$

and finally,

$$R_0(F_3) = R_1(F_3) = 5.376... + 1 = 6.376... \ (= R_*(F_3)).$$

For $i = 1$ and $m > 3$ part (i) is valid by induction since $F_{m,1} = F_{m-1,m}$. This concludes the base cases.

For $m \geq 3$ and any i we have by Theorem 1.2.9 and by induction, part (i) that

$$\begin{aligned} R_1(G_{m,i}) =& \ R_*(F_{m,i}) + R_*(F_m) \text{ and} \\ R_0(G_{m,i}) =& \ \min\{R_*(F_{m,i}) + R_*(F_m), \ \tfrac{R_*(F_{m,i})^2 + R_*(F_m)^2 + R_*(F_{m,i})R_*(F_m)}{R_*(F_{m,i}) + R_*(F_m)}\} \\ =& \ \tfrac{R_*(F_{m,i})^2 + R_*(F_m)^2 + R_*(F_{m,i})R_*(F_m)}{R_*(F_{m,i}) + R_*(F_m)}. \end{aligned}$$

Also, $R_0(F_{m,i+1}) = R_0(G_{m,i}) + 1$ and

$$R_1(F_{m,i+1}) = \min\{R_0(G_{m,i}) + 1, \ R_1(G_{m,i}) + 1, \ \frac{R_0(G_{m,i})R_1(G_{m,i}) + R_0(G_{m,i}) + 1}{R_0(G_{m,i}) + 1}\}$$

The first term is smallest if $R_0(G_{m,i}) + 1 \leq R_1(G_{m,i})$. This is indeed the case since $R_1(G_{m,i}) - R_0(G_{m,i}) = \frac{R_*(F_{m,i})R_*(F_m)}{R_*(F_{m,i}) + R_*(F_m)}$ and $R_*(F_{m,i}) \geq R_*(F_m) \geq R_*(F_3) > 2$. Therefore,

$$R_1(F_{m,i+1}) = R_0(G_{m,i}) + 1 = R_0(F_{m,i+1}).$$

This value is

$$\begin{aligned} R_*(F_{m,i+1}) =& \ \tfrac{R_*(F_{m,i})^2 + R_*(F_m)^2 + R_*(F_{m,i})R_*(F_m)}{R_*(F_{m,i}) + R_*(F_m)} + 1 \\ =& \ R_*(F_{m,i}) + \tfrac{R_*(F_m)^2}{R_*(F_{m,i}) + R_*(F_m)} + 1 \end{aligned}$$

and so

$$\frac{R_*(F_{m,i+1})}{R_*(F_m)} = \frac{R_*(F_{m,i})}{R_*(F_m)} + \frac{1}{\frac{R_*(F_{m,i})}{R_*(F_m)} + 1} + \frac{1}{R_*(F_m)} = b_{m,i} + \frac{1}{1 + b_{m,i}} + \frac{1}{R_*(F_m)} = b_{m,i+1}.$$

□

Corollary 3.2.7 $b_{m,m+1} = \frac{R_*(F_{m+1})}{R_*(F_m)}$. □

The next proposition estimates this ratio. For this we use another sequence $\{a_i\}$, defined by $a_1 = 1$ and $a_{i+1} = a_i + \frac{1}{1+a_i}$.

Proposition 3.2.8

$$\begin{array}{lrl} (i) & b_{m,i} & \geq a_i \\ (ii) & b_{m,i} & \leq a_i + \frac{i}{R_*(F_m)} \\ (iii) & a_i & \leq 2\sqrt{i} \\ (iiii) & a_i & \geq \sqrt{i} \end{array}$$

Proof: The proof is by induction on i. For $i = 1$, $b_{m,1} = a_1 = 1$ and the four parts follow.

Assume it is true for i. Consider the following three functions defined on the positive reals, $f(x) = x + \frac{1}{1+x}$, $g(x) = \sqrt{x^2+4}$ and $h(x) = \sqrt{x^2+1}$. It is easy to see that for $x \geq 0$, $h(x) \leq f(x) \leq g(x)$ and that $f(x)$ is monotone increasing. Using these properties and the induction hypothesis (i.h.) we have

$$\begin{array}{lll}
(i) & b_{m,i+1} & = b_{m,i} + \frac{1}{1+b_{m,i}} + \frac{1}{R_*(F_m)} = f(b_{m,i}) + \frac{1}{R_*(F_m)} > f(b_{m,i}) \overset{\text{i.h.}}{\geq} f(a_i) = a_{i+1}, \\
(ii) & b_{m,i+1} & = b_{m,i} + \frac{1}{1+b_{m,i}} + \frac{1}{R_*(F_m)} \leq a_i + \frac{i}{R_*(F_m)} + \frac{1}{1+b_{m,i}} + \frac{1}{R_*(F_m)} \\
 & & \overset{(i)}{\leq} a_i + \frac{1}{1+a_i} + \frac{i+1}{R_*(F_m)} = a_{i+1} + \frac{i+1}{R_*(F_m)}, \\
(iii) & a_{i+1} & = f(a_i) \overset{\text{i.h.}}{\leq} f(2\sqrt{i}) \leq g(2\sqrt{i}) = 2\sqrt{i+1}. \\
(iiii) & a_{i+1} & = f(a_i) \overset{\text{i.h.}}{\geq} f(\sqrt{i}) \leq h(\sqrt{i}) = \sqrt{i+1}.
\end{array}$$

□

From this proposition we get for $m \geq 3$ the (weak) lower bound,

$$R_*(F_m) \geq \frac{R_*(F_m)}{R_*(F_{m-1})} \overset{3.2.7}{=} b_{m-1,m} \overset{3.2.8(i)}{\geq} a_m \overset{3.2.8(iiii)}{\geq} \sqrt{m} \geq \frac{\sqrt{m+1}}{2}$$

and then the upper bound which completes the proof,

$$\frac{R_*(F_{m+1})}{R_*(F_m)} \overset{3.2.7}{=} b_{m,m+1} \overset{3.2.8(ii)}{\leq} a_{m+1} + \frac{m+1}{R_*(F_m)} \leq a_{m+1} + \frac{2(m+1)}{\sqrt{m+1}} \overset{3.2.8(iii)}{\leq} 4\sqrt{m+1}.$$

□

3.3 Simple Non-recursive Lower Bound

The following theorem is a simplified version of the generalized non-recursive lower bound, Theorem 1.3.1. The recursion it involves is quite simple. It has only a single R (replacing the two R_0 and R_1 of Theorems 1.2.9 and 1.3.1) and it contains no min operator. The function ψ it involves also has the nice properties stated thereafter. These are the reasons for using this version in the proof that appears in the next section. Luckily enough, this version, when used together with restrictions, suffices for the recursive theorem. We remark that $\psi(a_1, ..., a_k) = \frac{1}{2}(\sum a_i + \frac{\sum a_i^2}{\sum a_i})$, and that replacing it by $\frac{1}{2}\sum a_i$ would simplify the following theorem but this simplified version would not suffice for the non-recursive bound.

Theorem 3.3.1 *Let f be a read-once Boolean formula with $\wedge$-$\vee$ gates (of unbounded fan-in). Then $RC(f) \geq R(f)$ where $R(f)$ is given by the following recursion.*

$$\begin{array}{ll}
R(f) = 1 & \text{if } f \text{ is a single variable;} \\
R(f) = \psi(R(g_1), ..., R(g_k)) & \text{if } f = \vee(g_1 ... g_k) \text{ or } f = \wedge(g_1 ... g_k).
\end{array}$$

Here $\psi(a_1, ..., a_k) = (\sum_{1\leq i\leq j\leq k} a_i a_j)/(\sum_{i=1}^k a_i)$

Proof: We show by induction on the structure of f that $R_0(f) \geq R(f)$ and $R_1(f) \geq R(f)$. R_0 and R_1 are the recursive terms given in the generalized non-recursive lower bound, Theorem 1.3.1. The base case is trivial. By duality we prove it only for $f = \vee_{i=1}^{k}(g_i)$. In this case we have

$$R_0(f) = \textstyle\sum_{i=1}^{k} R_0(g_i), \text{ and}$$
$$R_1(f) = \chi(R_1(g_1), R_0(g_1), R_1(g_2), R_0(g_2), ..., R_1(g_k), R_0(g_k))$$

where $\chi(a_1, b_1, a_2, b_2, ..., a_k, b_k) =$

$$\min_{\emptyset \neq T=\{i_1,...,i_t\}\subseteq\{1,2,...,k\}} \sum_{i \notin T} b_i + \phi(a_{i_1}, b_{i_1}, a_{i_2}, b_{i_2}, ..., a_{i_t}, b_{i_t})]$$

and $\phi(a_1, b_1, a_2, b_2, ..., a_t, b_t) = (\sum_{j=1}^{t} a_j b_j + \sum_{1 \leq j < h \leq t} b_j b_h)/\sum_{j=1}^{t} b_j$.
Bounding R_0 is easy,

$$R_0(f) = \sum R_0(g_i) \geq \sum R(g_i) \geq \psi(R(g_1), ..., R(g_k)) = R(f).$$

To bound R_1 we use the following properties of ϕ and χ.

Proposition 3.3.2

(*i*) *For non negative a-s and b-s, χ is monotone non decreasing in each a_i.*
(*ii*) *For $b_i \geq a_i \geq 0$, $i = 1, ..., k$, χ is monotone non decreasing in each b_i.*
(*iii*) *For $a_i = b_i \geq 0$, $i = 1, ..., k$, $\phi(a_1, a_1, a_2, a_2, ..., a_k, a_k) = \psi(a_1, a_2, ..., a_k)$.*

Proof: (*i*) It is enough to prove that ϕ is such, since each of the 2^k terms under the min operator of χ is then non decreasing and so is χ itself. By symmetry we show monotonicity in a_1. Indeed,

$$\frac{\partial\phi}{\partial a_1}(a_1, b_1, ..., a_t, b_t) = \frac{b_1}{\sum_{i=1}^{t} a_i} \geq 0 \text{ for positive } b\text{-s.}$$

(*ii*) As in part (*i*) it is enough to prove monotonicity of ϕ in b_1. For this we calculate the numerator of $\frac{\partial\phi}{\partial b_1}(a_1, b_1, ..., a_t, b_t)$ and verify it to be non negative. Indeed,

$$(a_1 + \textstyle\sum_{h=2}^{t} b_h) \cdot (\sum_{j=1}^{t} b_j) - \sum_{j=1}^{t} a_j b_j - \sum_{1 \leq j < h \leq t} b_j b_h \geq$$
$$(\textstyle\sum_{h=2}^{t} b_h) \cdot (\sum_{j=1}^{t} b_j) - \sum_{j=2}^{t} b_j^2 - \sum_{1 \leq j < h \leq t} b_j b_h = \sum_{2 \leq h < j \leq t} b_j b_h \geq 0.$$

(*iii*) By the definitions of χ and ψ, we have to show that the minimal term in χ is when $T = \{1, ..., k\}$. By symmetry, it is enough to show that the term for $T = \{1, 2, ..., t\}$ where $0 < t < k$ is bounded from below by that for $\{1, ..., k\}$. Indeed,

$$\begin{aligned}
&\textstyle\sum_{i \notin T} a_i + \phi(a_1, a_1, a_2, a_2, ..., a_t, a_t)] = \\
&\textstyle\sum_{i=t+1}^{k} a_i + (\sum_{1 \leq j \leq h \leq t} a_j a_h)/\sum_{j=1}^{t} a_j = \\
&\textstyle\sum_{i=t+1}^{k} a_i + \sum_{i=1}^{t} a_i - \frac{1}{2} \cdot \sum_{i=1}^{t} a_i^2 \geq \\
&\textstyle\sum_{i=1}^{k} a_i - \frac{1}{2} \cdot \sum_{i=1}^{k} a_i^2 = \\
&(\textstyle\sum_{1 \leq j \leq h \leq k} a_j a_h)/\sum_{j=1}^{k} a_j = \\
&\phi(a_1, a_1, a_2, a_2, ..., a_m, a_k).
\end{aligned}$$

$\square$

We can now bound R_1 and complete the proof of Theorem 3.3.1.

$$\begin{aligned}
R_1(f) &= \chi(R_1(g_1), R_0(g_1), R_1(g_2), R_0(g_2), ..., R_1(g_k), R_0(g_k)) \\
&\overset{3.3.2(i)}{\geq} \chi(R(g_1), R_0(g_1), R(g_2), R_0(g_2), ..., R(g_k), R_0(g_k)) \\
&\overset{3.3.2(ii)}{\geq} \chi(R(g_1), R(g_1), R(g_2), R(g_2), ..., R(g_k), R(g_k)) \\
&\overset{3.3.2(iii)}{=} \phi(R(g_1), R(g_1), R(g_2), R(g_2), ..., R(g_k), R(g_k)) \\
&\overset{\text{def}}{=} \psi(R(g_1), R(g_2), ..., R(g_k)) = R(f).
\end{aligned}$$

$\square$

The following properties of ψ, the function used in the simple lower bound, Theorem 3.3.1 are used below.

Proposition 3.3.3
(i) ψ is homogeneous of order 1, $\psi(\alpha a_1, ..., \alpha a_k) = \alpha\psi(a_1, ..., a_k)$.
(ii) ψ is monotone non-decreasing in $I\!R_k^+$ in each of its variables.

Proof:
(i) Follows directly from the definition of ψ.
(ii) By symmetry we show monotonicity in the first argument of ψ.

$$\frac{\partial\psi}{\partial a_1}(a_1, ..., a_k) = \frac{(2a_1 + \sum_{i=2}^k a_i) \cdot (\sum_{i=1}^k a_i) - \sum_{1 \leq i \leq j \leq k} a_i a_j}{(\sum_{i=1}^k a_i)^2} \geq 0 \text{ for positive } a_i\text{'s.}$$

$\square$

3.4 Proving The Recursive Bound: Intuition

We want to prove Theorem 1.3.2,
Theorem *(recursive lower bound) There exists a real number $\theta > 0.5$ s.t. for every read-once AND/OR function f, $RC(f) \geq n^\theta$, where n is the number of input variables f depends on. In particular, $\theta = 0.51$ satisfies this inequality.*

As hinted above, we will actually prove that $R(f) \geq n^\theta$, where $R(f)$ is the lower bound for $RC(f)$ given in Theorem 3.3.1. A natural way to prove this is by induction on the structure of f. (Note we do not distinguish between the function and its read-once formula.) Let $f = g\Diamond h$, where $\Diamond \in \{\wedge, \vee\}$. Say h is of size αn, and g is of size $(1-\alpha)n$, where by size of a function we mean its number of input variables. We want to show that

$$R(f) \geq n^\theta.$$

By definition,

$$R(f) = \psi(R(g), R(h)).$$

By induction, we may assume that $R(h) \geq (\alpha n)^\theta$ and $R(g) \geq ((1-\alpha)n)^\theta$. Using the monotonicity of ψ, Proposition 3.3.3, we have

$$\psi(R(g), R(h)) \geq \psi((\alpha n)^\theta, ((1-\alpha)n)^\theta).$$

Therefore we are done IF

$$\psi((\alpha n)^\theta, ((1-\alpha)n)^\theta) \overset{??}{\geq} n^\theta,$$

or, equivalently, by the fact that ψ is homogeneous (Proposition 3.3.3) IF

$$\psi(\alpha^\theta, (1-\alpha)^\theta) \overset{??}{\geq} 1.$$

'Unfortunately', this is not true. For every $\theta > \frac{1}{2}$, if f is very unbalanced, that is, if α is very close to 0 or 1, $\psi(\alpha^\theta, (1-\alpha)^\theta) < 1$. Well, this is not a big surprise, as we know by the example given in Section 3.2 — we have not used restrictions so far. Yet, we cannot simply restrict f so to eliminate the smaller subfunction. This way we might eliminate eventually almost everything. So what we do in the proof is look deeper in the structure of f, analyze this structure, and carefully restrict small parts of it.

Let us describe this in slightly more detail. We look at the larger sub-function of f. Say it is g, that is, say α is small. If g is also very unbalanced we look at its larger sub-function too, and so on. We stop when the small sub-functions we have seen in this process have together 'sufficiently large' total size. Now, each such small sub-function is a child of some gate, either AND or OR. We count the total size of sub-functions under AND gates and compare it to the total size of those under OR gates. Say the latter is greater. In this case we restrict f so as to eliminate exactly those small sub-functions under OR gates, leaving all other parts of f as is. We view the restricted function as a single AND, possibly of large fan-in, whose children are all the sub-functions that remain alive. On this restricted function we want to evaluate ψ. However, this is still not enough. ψ may still be smaller than 1 when evaluated at the θ-powers of the remaining sub-functions' sizes. This happens if the two total sizes were approximately the same.

In order to have a valid argument we distinguish between three cases of how the total size of the small sub-functions is distributed among them. These are the three cases mentioned in the next section. We then restrict f in a specific way for each case, and use a corresponding and specific inequality involving ψ. These three inequalities are stated as Lemma 3.5.1 below, and are brutally proved in Section 3.6.

A formal proof follows next.

3.5 Proving The Recursive Bound: Restrictions

We show that $\theta = 0.51$ satisfies the requirement of Theorem 1.3.2. We use also the parameters $b = 0.001$, $c = 0.025$ and $\delta = 0.4$. The following technical lemma, proved in the next section, is used in the proof.

Lemma 3.5.1 *(technical lemma)*

$$\begin{array}{lll} (i) & \psi((1-\varepsilon)^\theta, (\varepsilon-\varepsilon')^\theta) \geq 1, & \textit{for } c \leq \varepsilon \leq \frac{1+b}{2} \textit{ and } 0 \leq \varepsilon' \leq b. \\ (ii) & \psi((1-\varepsilon)^\theta, \varepsilon'^\theta) \geq 1, & \textit{for } \frac{1-\delta}{2-\delta}\delta b \leq \varepsilon \leq c \textit{ and } \frac{1}{2-\delta}\varepsilon \leq \varepsilon' \leq \varepsilon. \\ (iii) & \psi((1-\varepsilon)^\theta, \varepsilon'^\theta, \varepsilon''^\theta) \geq 1, & \textit{for } b \leq \varepsilon \leq c,\ \frac{1}{2-\delta}\frac{\varepsilon}{2} \leq \varepsilon' \leq \varepsilon'' \textit{ and } \varepsilon' + \varepsilon'' \leq \frac{\varepsilon}{2}. \end{array}$$

We prove by induction on n, the number of f's input variables, that $R(f) \geq n^\theta$. Note that R, which is defined in Theorem 3.3.1, is by that theorem a lower bound for RC. The case $n = 1$ is trivial. Assume $n > 1$. By Lemma 1.2.4, we assume that the formula contains no negation gates. Assume also that all gates of f have fan-in two to begin with. Otherwise simply replace 'wide' gates by binary trees.

Furthermore, consider the formula as a binary tree in which the left subtree of each node is at least as large as its right subtree (with respect to their numbers of leaves). Denote this tree by T. Now focus at the 'leftmost' path of T which starts at the root and ends in a certain gate, specified in (2) below. This path is of the form of

$$T = (...(...(T_* \diamondsuit_t T_t)...\diamondsuit_i T_i)...\diamondsuit_2 T_2)\diamondsuit_1 T_1$$

where each $\diamondsuit_i$ is either AND or OR, and the T_i-s and T_* are subtrees of T. Let n_i (respectively n_*) be the number of leaves in the subtree T_i (respectively T_*). Define $\varepsilon_i = \frac{n_i}{n}$ and $\varepsilon_* = \frac{n_*}{n}$ $(= 1 - \sum \varepsilon_i)$. The following two conditions precisely define the path.
(1) For each i, $\sum_{j=i+1}^{t} \varepsilon_j + \varepsilon_* \geq \varepsilon_i$. This is the above mentioned requirement on the number of leaves in left- and right-subtrees.
(2) t is minimal with $\sum_{i=1}^{t} \varepsilon_i \geq b$. This specifies the last gate in the leftmost path we focus on.

For each i let $S_i = \sum_{j=1}^{i} \varepsilon_j$, let $I_i^\wedge = \{j : 1 \leq j \leq i,\ \diamondsuit_j = \wedge\}$ and let $S_i^\wedge = \sum_{j \in I_i^\wedge} \varepsilon_j$. Similarly, define $I_i^\vee = \{j : 1 \leq j \leq i,\ \diamondsuit_j = \vee\}$ and $S_i^\vee = \sum_{j \in I_i^\vee} \varepsilon_j$. Note that $S_i = S_i^\wedge + S_i^\vee$ and $S_t \geq b$.

We distinguish between the following three possible cases:
Case 1: $S_t \geq c$. (T_t is 'very' large with respect to the other T_i's.)
Case 2: $b \leq S_t < c$ and $\varepsilon_u > \delta S_t$ for some u, $1 \leq u \leq t$. (No T_i is 'very' large but one of them is 'quite' large.)
Case 3: $b \leq S_t < c$ and $\varepsilon_u \leq \delta S_t$ for all u, $1 \leq u \leq t$. (All T_i's are 'small'.)

In case 1 we argue that all T_i-s with $i < t$ can be ignored: Restrict T to $T_* \diamondsuit_t T_t$ having $R(T) \geq R(T_* \diamondsuit_t T_t)$. By the induction hypothesis

$$R(T_* \diamondsuit_t T_t) \geq \psi(R(T_*), R(T_t)) \geq \psi(n_*^\theta, n_t^\theta).$$

To show $\psi(n_*^\theta, n_t^\theta) \geq n^\theta$, we use the homogeneity of ψ (Proposition 3.3.3), and show that $\psi(\varepsilon_*^\theta, \varepsilon_t^\theta) \geq 1$, i.e.,

$$\psi((1 - S_t)^\theta, (S_t - S_{t-1})^\theta) \geq 1.$$

Recall that $S_{t-1} < b$ and that $\varepsilon_t \leq \frac{1}{2}(1 - S_{t-1})$ or $c \leq S_t \leq S_{t-1} + \frac{1-S_{t-1}}{2} = \frac{1+S_{t-1}}{2} \leq \frac{1+b}{2}$. Case 1 follows by Lemma 3.5.1, part (i) with $\varepsilon = S_t$ and $\varepsilon' = S_{t-1}$.

Consider now case 2. The following claim states that for some prefix of the leftmost path of non negligible size, the total size of subtrees entering its AND gates significantly differs from that of those entering its OR gates. In this case we argue that the subtrees of smaller total size can be ignored.

Claim 3.5.2 *$\exists r \in \{u-1, u\}$ s.t. $S_r^{\Diamond_r} \geq \frac{1}{2-\delta} S_r$ and $S_r \geq \frac{1-\delta}{2-\delta}\delta b$.*

Proof: Assume without loss of generality that $\Diamond_u = \wedge$. If $S_u^{\wedge} \geq \frac{1}{2-\delta} S_u$ then $r = u$ fulfills the claim, since $S_u \geq \varepsilon_u \geq \delta S_t \geq \delta b$. Otherwise, $S_u^{\vee} (= S_{u-1}^{\vee}) \geq \frac{1-\delta}{2-\delta} S_u$. In this case,

$$\frac{S_{u-1}^{\vee}}{S_{u-1}} = \frac{S_{u-1}^{\vee}}{S_u - \varepsilon_u} \geq \frac{S_{u-1}^{\vee}}{S_u - \delta S_t} \geq \frac{\frac{1-\delta}{2-\delta} S_u}{S_u - \delta S_u} = \frac{1}{2-\delta}.$$

Accordingly, we choose $\Diamond = \vee$, $r = u - 1$, and we are done, since

$$S_{u-1} \geq S_{u-1}^{\vee} \geq \frac{1-\delta}{2-\delta} S_u \geq \frac{1-\delta}{2-\delta}\varepsilon_u \geq \frac{1-\delta}{2-\delta}\delta S_t \geq \frac{1-\delta}{2-\delta}\delta b.$$

□

Suppose, for the purpose of simplifying notation, that $\Diamond_u = \wedge$. Restrict the given formula so that each T_i with $i \leq u$ and $\Diamond_i = \vee$ becomes '0'. Represent the remaining tree in the form

$$T_{\Box} \wedge T_{\triangle}$$

where $T_{\Box} = (...(T^* \Diamond_t T_t)...\Diamond_{r+1} T_{r+1})$ and $T_{\triangle} = \wedge_{i \in I_r^{\wedge}} T_i$. As in case 1, using the induction hypothesis and the homogeneity of ψ, it is sufficient to show

$$\psi((1 - S_r)^{\theta}, (S_r^{\wedge})^{\theta}) \geq 1.$$

This follows by Lemma 3.5.1, part (ii), using $\varepsilon = S_r$ and $\varepsilon' = S_r^{\wedge}$.

Finally, consider case 3. Assume without loss of generality that $S_t^{\wedge} \geq S_t^{\vee}$, i.e., $S_t^{\wedge} \geq \frac{S_t}{2}$. We argue that also in this case one can ignore the subtrees entering OR gates. However, this time we have to be more careful and use the fact that the subtrees entering AND gates are small and hence can be partitioned into two sets of similar total sizes. For a subset $I \subseteq \{1, ..., t\}$, denote $S_I = \sum_{i \in I} \varepsilon_i$. The following claim states that there exists a partition of $I_t^{\wedge}$ into two subsets that are 'fairly' balanced with respect to the total number of inputs in the subformulae each represents.

Claim 3.5.3 *There exists a subset $I \subset I_t^{\wedge}$ s.t. $S_{I_t^{\wedge} \setminus I} \geq S_I \geq (1 - 2\delta) \cdot \frac{S_t}{2}$.*

Proof: Use the following fact with $\beta = 2\delta = 0.8$ and scaling. □

Fact 3.5.4 *Let* $0 \le \beta \le 1$ *and* $a_1, ..., a_k$ *s.t.* $0 \le a_i \le \beta$ *and* $\sum_{i=1}^k a_i \ge 1$. *Then* $\exists I \subseteq \{1, ..., k\}$ *s.t.* $\min\{\sum_{i\in I} a_i, \sum_{i\notin I} a_i\} \ge \gamma(\beta)$ *where*

$$\gamma(\beta) = \begin{cases} \frac{1-\beta}{2} & \text{if } \beta \le \frac{1}{3} \\ \frac{1}{3} & \text{if } \frac{1}{3} \le \beta \le \frac{2}{3} \\ 1-\beta & \text{if } \beta \ge \frac{2}{3} \end{cases}$$

Proof: If $\beta \le \frac{1}{3}$, then any partition with sums that differ by more than β can be improved by moving some a_i from the larger part to the other. If there is some part larger than $\frac{1}{3}$ put that part in one subset and all the others in the other subset. This covers the two last cases. □

Now, consider the subset I given by Claim 3.5.3 and restrict T to

$$T_* \wedge T_\square \wedge T_\triangle$$

where $T_\square = \wedge_{i\in I} T_i$ and $T_\triangle = \wedge_{i\in I_t^\wedge \setminus I} T_i$. Part (iii) of Lemma 3.5.1 with $\varepsilon = S_t$, $\varepsilon' = S_I$ and $\varepsilon'' = S_{I_t^\wedge \setminus I}$ shows that $\psi(\varepsilon_*^\theta, S_I^\theta, S_{I_t^\wedge\setminus I}^\theta) \ge 1$. This completes case 3 and therefore completes the proof of the theorem, once the technical lemma is proved. □

3.6 Proving The Technical Lemma

We now prove the technical lemma, Lemma 3.5.1. Using the monotonicity of ψ in each of its arguments (Proposition 3.3.3), and using $\delta = 0.4$, we replace the three inequalities of the lemma by the following.

(i') $\psi((1-\varepsilon)^\theta, (\varepsilon - b)^\theta) \ge 1$ for $c \le \varepsilon \le \frac{1+b}{2}$.
(ii') $\psi((1-\varepsilon)^\theta, (\frac{5}{8}\varepsilon)^\theta) \ge 1$ for $0.15b \le \varepsilon \le c$.
(iii') $\psi((1-\varepsilon)^\theta, \varepsilon'^\theta, \varepsilon''^\theta) \ge 1$ for $b \le \varepsilon \le c$, $0.1\varepsilon \le \varepsilon' \le \varepsilon''$ and $\varepsilon' + \varepsilon'' = \frac{\varepsilon}{2}$.

As mentioned, we prove the inequalities for $\theta = 0.51$, $b = 0.001$ and $c = 0.025$. We start with (iii'). Denote $\varepsilon' = \rho\varepsilon$. Then $0.1 \le \rho \le 0.3$, $\varepsilon'' = (\frac{1}{2} - \rho)\varepsilon$ and

$$\begin{aligned} \psi((1-\varepsilon)^\theta, \varepsilon'^\theta, \varepsilon''^\theta) &= (1-\varepsilon)^\theta + \tfrac{(\varepsilon')^{2\theta} + (\varepsilon'\varepsilon'')^\theta + (\varepsilon'')^{2\theta}}{(1-\varepsilon)^\theta + (\varepsilon')^\theta + (\varepsilon'')^\theta} \\ &= (1-\varepsilon)^\theta + \tfrac{[\rho^{2\theta} + \rho^\theta(\frac{1}{2}-\rho)^\theta + (\frac{1}{2}-\rho)^{2\theta}]\varepsilon^{2\theta}}{(1-\varepsilon)^\theta + [\rho^\theta + (\frac{1}{2}-\rho)^\theta]\varepsilon^\theta}. \end{aligned}$$

Hence, we have to show that

$$(1-\varepsilon)^\theta + \frac{[\rho^{2\theta} + \rho^\theta(\frac{1}{2}-\rho)^\theta + (\frac{1}{2}-\rho)^{2\theta}]\varepsilon^{2\theta}}{(1-\varepsilon)^\theta + [\rho^\theta + (\frac{1}{2}-\rho)^\theta]\varepsilon^\theta} \ge 1 \quad \text{for } 0.1 \le \rho \le 0.3.$$

By Lemma 3.6.1 below, $\rho^{2\theta} + \rho^\theta(\frac{1}{2}-\rho)^\theta + (\frac{1}{2}-\rho)^{2\theta} \ge 0.1^{1.02} + 0.04^{0.51} + 0.4^{1.02} > 0.68$, and $\rho^\theta + (\frac{1}{2}-\rho)^\theta \le 2 \cdot 0.25^{0.51} < 1$. It is thus enough to show that

$$0.68\varepsilon^{2\theta} \ge [1 - (1-\varepsilon)^\theta] \cdot [(1-\varepsilon)^\theta + \varepsilon^\theta].$$

Again, by Lemma 3.6.1 $(1-\varepsilon)^\theta + \varepsilon^\theta \leq 0.975^{0.51} + 0.025^{0.51} < 1.14$, since $\varepsilon \leq c = 0.025$. Also, $1-(1-\varepsilon)^\theta \leq \varepsilon\theta(1-\varepsilon)^{\theta-1} \leq \varepsilon\theta \cdot 0.975^{-0.49} < \varepsilon\theta \cdot 1.013$. Hence, it is enough to show that

$$\varepsilon^{2\theta-1} \geq \frac{0.51 \cdot 1.013 \cdot 1.14}{0.68}.$$

Indeed, $0.51 \cdot 1.013 \cdot 1.14 < 0.59$, $2\theta - 1 = \frac{1}{50}$ and $\varepsilon \geq b = 0.001 > (\frac{0.59}{0.68})^{50}$.

We now prove (ii') which states

$$(1-\varepsilon)^\theta + \frac{(0.625 \cdot \varepsilon)^{2\theta}}{(1-\varepsilon)^\theta + (0.625 \cdot \varepsilon)^\theta} \geq 1.$$

We use Lemma 3.6.1 again, $1-(1-\varepsilon)^\theta \leq \varepsilon\theta(1-\varepsilon)^{\theta-1}$, and show that

$$\varepsilon^{2\theta-1} \geq \frac{\theta(1-\varepsilon)^{\theta-1}}{0.625^{2\theta}} \cdot [(1-\varepsilon)^\theta + (0.625 \cdot \varepsilon)^\theta].$$

The left hand side is monotone increasing in ε. By Lemma 3.6.1, the right hand side is also monotone increasing in ε. Hence, to prove this last inequality for $0.15b \leq \varepsilon \leq c$, we exhibit a sequence $0.15b = \varepsilon_0 < \varepsilon_1 < \ldots < \varepsilon_k = c$ for which for each i, $1 \leq i \leq k$,

$$\varepsilon_{i-1}^{2\theta-1} \geq \frac{\theta(1-\varepsilon_i)^{\theta-1}}{0.625^{2\theta}} \cdot [(1-\varepsilon_i)^\theta + (0.625 \cdot \varepsilon_i)^\theta].$$

Indeed, the sequence 0.00015, 0.0006, 0.0034, 0.012, 0.021, 0.025 satisfies this requirement.

Finally, we prove (i'), that is,

$$(1-\varepsilon)^{2\theta} + (1-\varepsilon)^\theta(\varepsilon - 0.001)^\theta + (\varepsilon - 0.001)^{2\theta} \geq (1-\varepsilon)^\theta + (\varepsilon - 0.001)^\theta,$$

for $c \leq \varepsilon \leq \frac{1+b}{2}$. By Lemma 3.6.1, both sides of the inequality are monotone, and so, as before, we exhibit a sequence $c = \varepsilon_0 < \varepsilon_1 < \ldots < \varepsilon_k = \frac{1+b}{2}$ for which for each i, $1 \leq i \leq k$, the left hand side with $\varepsilon = \varepsilon_{i-1}$ is greater than the right hand side with $\varepsilon = \varepsilon_i$. A sequence satisfying this is 0.025, 0.026, 0.027, ..., 0.068, 0.069, 0.07, 0.08, 0.09, ..., 0.18, 0.19, 0.2, 0.3, 0.5005, where the first ellipsis corresponds to 0.001-jumps and the second corresponds to 0.01-jumps. □

To complete the proof of the technical lemma we need the following.

Lemma 3.6.1
(i) Let $0 < \theta \leq 1$ and $a > 0$. Then $(1-\varepsilon)^\theta + (a\varepsilon)^\theta$ is convex in the segment $0 \leq \varepsilon \leq 1$ and maximal in $\frac{1}{1+a^{\frac{\theta}{\theta-1}}}$.
(ii) Let $0 < \theta \leq 1$ and $0 \leq b \leq 1$. Then $(1-\varepsilon)^\theta + (\varepsilon - b)^\theta$ is convex in the segment $b \leq \varepsilon \leq 1$ and maximal in $\varepsilon = \frac{1+b}{2}$.
(iii) Let $0 < \theta \leq 0.6$ and $0 \leq b \leq 1$. Then $(1-\varepsilon)^{2\theta} + (1-\varepsilon)^\theta(\varepsilon - b)^\theta + (\varepsilon - b)^{2\theta}$ is convex in the segment $b \leq \varepsilon \leq 1$ and maximal in $\varepsilon = \frac{1+b}{2}$.
(iiii) Let $0 < \theta < 1$ and $0 \leq \varepsilon \leq 1$. Then $1-(1-\varepsilon)^\theta \leq \varepsilon\theta(1-\varepsilon)^{\theta-1}$.

Proof: We prove (i) and (ii) together: Consider $f(\varepsilon) = (1-\varepsilon)^\theta + [a(\varepsilon - b)]^\theta$. Then $\frac{\partial f}{\partial \varepsilon} = -\theta(1-\varepsilon)^{\theta-1} + a^\theta \theta(\varepsilon - b)^{\theta-1}$. This equals 0 iff $1-\varepsilon = a^{(\frac{\theta}{\theta-1})}(\varepsilon - b)$, i.e., $\theta = \frac{1+ba^{\frac{\theta}{\theta-1}}}{1+a^{\frac{\theta}{\theta-1}}}$. Also, $\frac{\partial^2 f}{\partial \varepsilon^2} = \theta(\theta-1)(1-\varepsilon)^{\theta-2} + a^\theta\theta(\varepsilon-b)^{\theta-2} \leq 0$.
(iii): Consider $g(\varepsilon) = (1-\varepsilon)^{2\theta} + (\varepsilon-b)^{2\theta} + (1-\varepsilon)^\theta(\varepsilon-b)^\theta$. Then

$$\frac{\partial g}{\partial \varepsilon} = -2\theta(1-\varepsilon)^{2\theta-1} + 2\theta(\varepsilon-b)^{2\theta-1} - \theta(1-\varepsilon)^{\theta-1}(\varepsilon-b)^\theta + \theta(1-\varepsilon)^\theta(\varepsilon-b)^{\theta-1}.$$

First, $\frac{\partial g}{\partial \varepsilon} = 0$ iff $1-\varepsilon = \varepsilon - b$, that is, $\varepsilon = \frac{1+b}{2}$. Second,

$$\begin{aligned}
\frac{1}{\theta}\cdot\frac{\partial^2 g}{\partial \varepsilon^2} &= 2(2\theta-1)(1-\varepsilon)^{2\theta-2} + 2(2\theta-1)(\varepsilon-b)^{2\theta-2} \\
&\quad +(\theta-1)(1-\varepsilon)^{\theta-2}(\varepsilon-b)^\theta - 2\theta(1-\varepsilon)^{\theta-1}(\varepsilon-b)^{\theta-1} + (\theta-1)(1-\varepsilon)^\theta(\varepsilon-b)^{\theta-2} \\
&\leq 2(2\theta-1)[(1-\varepsilon)^{2\theta-2} + (\varepsilon-b)^{2\theta-2}] + (\theta-1)[(1-\varepsilon)^{\theta-2}(\varepsilon-b)^\theta \\
&\quad +(1-\varepsilon)^\theta(\varepsilon-b)^{\theta-2}].
\end{aligned}$$

Now, $2(2\theta-1) \leq -(\theta-1)$ for $\theta \leq 0.6$, and so to show $\frac{\partial^2 g}{\partial \varepsilon^2} < 0$ we show

$$(1-\varepsilon)^{2\theta-2} + (\varepsilon-b)^{2\theta-2} \leq (1-\varepsilon)^{\theta-2}(\varepsilon-b)^\theta + (1-\varepsilon)^\theta(\varepsilon-b)^{\theta-2}, \text{ i.e.,}$$

$$0 \leq [(\varepsilon-b)^{\theta-2} - (1-\varepsilon)^{\theta-2}]\cdot[(1-\varepsilon)^\theta - (\varepsilon-b)^\theta].$$

Indeed, if $1-\varepsilon \geq \varepsilon - b$ the two multiplied terms are non negative, and if $1-\varepsilon \leq \varepsilon - b$ they are both non positive.
$(iiii)$: Consider $h(\varepsilon) = -(1-\varepsilon)^\theta$. Then $1-(1-\varepsilon)^\theta = h(\varepsilon) - h(0) = \varepsilon\frac{\partial h}{\partial \varepsilon}(\varepsilon')$ for some $0 \leq \varepsilon' \leq \varepsilon$, and we have $\frac{\partial h}{\partial \varepsilon}(\varepsilon') = \theta(1-\varepsilon')^{\theta-1} \leq \theta(1-\varepsilon)^{\theta-1}$. □

3.7 On Improving the Recursive Bound

When focusing on the lower bound R of Theorem 3.3.1, one gains, indeed, the simplicity which helps proving Theorem 1.3.2. However, Theorem 3.3.1 cannot imply an optimal lower bound. For g, the complete AND/OR-tree function defined in Example 1.2.3, $R(g) = \frac{3}{2}^{\log n} = n^{0.58\ldots}$, and restrictions cannot help here. While it is clear from our proof that $\theta = 0.51$ is not the highest exponent this proof technique can provide, this example shows that 0.58... is an upper bound for any proof based on Theorem 3.3.1.

Chapter 4

Lower Bound for Threshold Gates

4.1 Introduction

This chapter is devoted to the proof of Theorem 1.3.3,
Theorem *(threshold gates) Let f be a Boolean function computed by a read-once threshold formula of depth d over n input-variables. Then $RC(f) \geq \frac{n}{2^d}$.*

In the next section we give a simple proof for a weaker lower bound of $\frac{n}{4^d}$. The proof is simple as it reduces this weaker result to a theorem we have already proved, Theorem 1.3.1.

Sections 4.3 to 4.5 contain the proof of Theorem 1.3.3. The proof is based on directly using the proof technique of the generalized non-recursive lower bound, Theorem 1.3.1, as well as on a claim involving the new concept of *partial* decision trees. The direct proof given here is perhaps better for the study of the randomized decision tree complexity in general, and of threshold circuits in particular. Comparing Theorem 1.3.3 with Observation 4.2.2 below suggests that threshold gates do not increase the gap between the randomized decision tree complexity and the deterministic decision tree complexity relative to AND/OR gates. Hence it might be that a lower bound for the randomized decision tree complexity of AC^0, if established, would generalize to a TC^0 lower bound, solving, perhaps, the TC^0 versus NC^1 problem.

The direct proof also has the advantage of remaining valid in a more powerful model. This model enables, in particular, gates that compute arbitrary symmetric functions:

Definition 4.1.1 *A Boolean function g, defined on k input-variables, is said to contain a flip if there exists some l, $1 \leq l \leq k$, such that g outputs the same value whenever exactly l of its inputs are valued 1, and it outputs the opposite value whenever exactly $l-1$ of its inputs are valued 1.*

Corollary 4.1.2 *Let f be a Boolean function computed by a read-once formula of depth d over n input-variables whose gates are functions that each contains a flip. Then $RC(f) \geq \frac{n}{2^d}$.*

Proof: The proof of Theorem 1.3.3 given below works for gates containing flips as well. □

Section 4.6 concludes this chapter. It contains a claim which is similar to that of 4.5, under a more general assumptions — it deals with partial decision tree that computes any function. It is one of the referees of this thesis who suggested this general result, a result which might be interesting in its own right.

4.2 Simple Proof for Weaker Lower Bound

In this section we give a simple proof for a weaker lower bound, $RC(f) \geq \frac{n}{4^d}$.

Theorem 4.2.1 *(weaker bound) Let f be a Boolean function computed by a read-once threshold formula of depth d over n input-variables. Then $RC(f) \geq \frac{n}{4^d}$*

The proof uses the following two observations.

Observation 4.2.2 *($\frac{n}{2^d}$ for AND/OR formulae) Let f be a Boolean function computed by a read-once AND/OR formula of depth d over n input-variables. Then $RC(f) \geq \frac{n}{2^d}$*

Proof: It is implied by the lower bound of Theorem 3.3.1. That lower bound, $R(f)$ is given by the recursion

$$\begin{array}{ll} R(f) = 1 & \text{if } f \text{ is a single variable;} \\ R(f) = \psi(R(g_1), ..., R(g_k)) & \text{if } f = \vee(g_1...g_k) \text{ or } f = \wedge(g_1...g_k), \end{array}$$

where $\psi(a_1, ..., a_k) = (\sum_{1\leq i\leq j\leq k} a_i a_j)/(\sum_{i=1}^{k} a_i)$.
To deduce the $\frac{n}{2^d}$ lower bound it suffices to show that

$$R(f) \geq \frac{\sum_{i=1}^{k} R(g_i)}{2} \text{ if } f = \vee(g_1...g_k) \text{ or } f = \wedge(g_1...g_k).$$

Indeed,

$$\psi(a_1, ..., a_k) = (\sum_{1\leq i\leq j\leq k} a_i a_j)/(\sum_{i=1}^{k} a_i) > \frac{1}{2} \cdot \sum_{i=1}^{k} a_i$$

□

Observation 4.2.3 *Let F be a read-once threshold formula of depth d over n input-variables. Then F can be restricted into a read-once AND/OR formula of depth d over $\frac{n}{2^d}$ input-variables.*

Proof: We prove it by induction on d. This is clear if $d = 0$, that is, F is just a single variable. Now let $F = T_l^k(F_1, F_2, ..., F_k)$, where F_i depends on n_i inputs for $i = 1, 2, ..., k$ and where $\sum_{i=1}^k n_i = n$. We also know that each F_i is of depth at most $d-1$. To simplify notation assume that $n_1 \geq n_2 \geq ... \geq n_k$. By duality assume that $l \geq \frac{k}{2}$. Restrict $F_{l+1}, ..., F_k$ to 1. This restricts F to $\wedge(F_1, ..., F_l)$. By induction, restrict each F_i for $i = 1, ..., l$ to an AND/OR formula F_i' of depth $d-1$ which depends on at least $\frac{n_i}{2^{d-1}}$ inputs. The effect is what we want, a depth-d AND/OR formula depending on at least

$$\sum_{i=1}^{l} \frac{n_i}{2^{d-1}} = \frac{\sum_{i=1}^{l} n_i}{2^{d-1}} \geq \frac{n/2}{2^{d-1}} = \frac{n}{2^d}$$

input variables. $\square$

Proof of Theorem 4.2.1 Let F be a depth-d n-inputs formula for f. By Observation 4.2.3 restrict it to a depth-d read-once formula, F' depending on $\frac{n}{2^d}$ input variables. Clearly the randomized decision tree complexity of f is at least that of the function computed by F', which in turn is by Proposition 4.2.2 at least $\frac{n}{2^d}/2^d = \frac{n}{4^d}$. $\square$

4.3 Proving the Threshold Gates Lower Bound

We now prove Theorem 1.3.3. As in Chapter 2, we use a variable cost function. Here a single cost function $c : \{x_1, ..., x_n\} \rightarrow \mathbb{R}$ suffices. The time relative to such c is defined in the natural way,

$$\text{Time}_{c,T}(\varepsilon) = \sum_{x_i \in \text{Path}_T(\varepsilon)} c(x_i).$$

$DC(f, c)$ and $RC(f, c)$ denote the complexities relative to c and are defined similarly to (1.1) and (1.2). Yao's lemma (Lemma 1.2.1) then becomes

$$RC(f, c) = \max_D \min_T \text{E}_{\varepsilon \in D}[\text{Time}_{c,T}(\varepsilon)]. \tag{4.1}$$

As in Chapter 2, we use a bottom-up induction, the threshold shrinking lemma, and remark that a top-down induction may possibly work as well, as suggested by the proof technique of Santha [San91]. In the lemma's proof, Section 4.4, we carefully define a distribution on inputs and a set of decision trees. These definitions generalize the analogous ones given in Chapter 2. They enable reducing the lemma's statement to a statement involving a simple threshold formula only, i.e., a single gate. The reduced statement is a claim on partial decision trees that compute a simple threshold function. Section 4.5 is devoted to this claim.

The threshold gates lower bound, Theorem 1.3.3 follows by applying the threshold shrinking lemma given below inductively. We begin with unit variables cost. The last shrinking yields the simple formula consisting of a single variable, v', whose cost bounds $RC(f)$ from below. As in Observation 4.2.2, this cost is easily computed:

$$c(v') = \sum_{i=1}^{n} 2^{-\text{Depth}(x_i)} \geq \frac{n}{2^d}.$$

Here $\text{Depth}(x_i)$ is the depth of a variable x_i in the read-once formula given in the theorem $\square$

4.4 The Threshold Shrinking Lemma

Lemma 4.4.1 *(threshold shrinking) Let F be a read-once threshold formula of depth $d > 0$ that computes a Boolean function f. Consider an internal gate T_l^k whose children are all variables. Denote these variables by $Y = \{y_1, ..., y_k\}$. Denote the rest of the variables by $X = \{x_1, ..., x_m\}$ (see Figure 4.1). Let $c : X \cup Y \rightarrow I\!R$ be a cost function for the $m + k$ variables of f. Let F' be the formula obtained from F by replacing the sub-formula $T_l^k(y_1, ..., y_k)$ by a single variable v (see Figure 4.2), and let f' be the function computed by F'. Define a new cost function c' by $c'(x_i) = c(x_i)$ for $1 \leq i \leq m$, and $c'(v) = \frac{c(Y)}{2}$, where $c(Y) = \sum_{i=1}^{k} c(y_i)$.*
Then $RC(f', c') \leq RC(f, c)$.

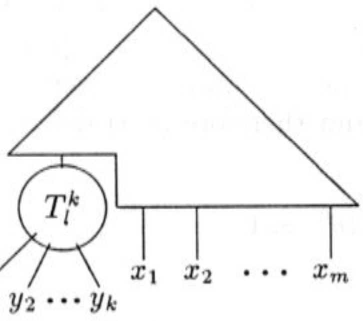


Figure 4.1: The given F

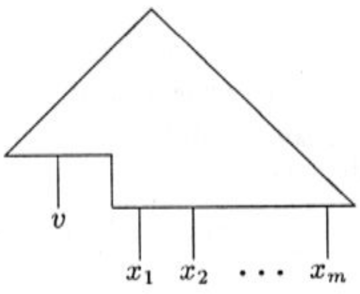


Figure 4.2: The shrunk F'

Proof:

We use the same notations as in Chapter 2,

Notations:

$[k]$ denotes the set $\{1, ..., k\}$.

$\{{A \atop a}\}$ denotes the set of all subsets of a set A which have cardinality a.

θ^1 (respectively, θ^0) denotes the extension of a (partial) setting $\theta : X \rightarrow \{0, 1\}$, on $X \cup \{v\}$ by $\theta^1(v) = 1$ (respectively, $\theta^0(v) = 0$).

θ_M where $M \subseteq [k]$ denotes the extension of $\theta : X \rightarrow \{0, 1\}$ to $X \cup Y$ by $\theta_M(y_i) = 1_{i \in M}$ (i.e., 1 if $i \in M$ and 0 otherwise).

$\Pr_D(E)$ denotes the probability of E, given a distribution D.
$c(U)$ denotes the total cost of a subset U of input variables, $c(U) = \sum_{u \in U} c(u)$.
1_U denotes the input setting that gives 1 exactly to those variables in U.
U_ε^T denotes the variables in some subset U that are probed by T given an input-setting ε (to all variables), $U_\varepsilon^T = U \cap \mathrm{Var}_T(\varepsilon)$.

We have to show that $RC(f', c') \leq RC(f, c)$. Using (4.1) we show that

$$(\forall D')\ (\exists D)\ (\forall T)\ (\exists T')\ \mathrm{E}_{\varepsilon' \in D'}[\mathrm{Time}_{c',T'}(\varepsilon')] \leq \mathrm{E}_{\varepsilon \in D}[\mathrm{Time}_{c,T}(\varepsilon)], \tag{4.2}$$

where D (respectively, D') is a distribution on the input-settings to f (respectively, f'), and T (respectively, T') is a deterministic decision tree for f (respectively, f').

Let D' be given. Define a distribution D as follows. For every X-setting $\theta : X \to \{0,1\}$ and subset $M \subseteq [k]$, define

$$\Pr_D(\theta_M) = \begin{cases} \Pr_{D'}(\theta^1) \cdot w(M) & \text{if } |M| = l \\ \Pr_{D'}(\theta^0) \cdot w([k] \setminus M) & \text{if } |M| = l - 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where

$$w(S) = \frac{\sum_{i \in S} c(y_i)}{\binom{k-1}{|S|-1} \cdot c(Y)} \tag{4.4}$$

for a non-empty set $S \subseteq [k]$. (The point here is to split $\Pr_{D'}(\theta^1)$ and $\Pr_{D'}(\theta^0)$ among the extensions of θ that are difficult to separate. These are the extensions θ_M for which $|M| = l$ or $|M| = l - 1$. The portion of probability that such an extension gets is proportional to its weight, that is, to the cost of the 'meaningful' Y-variables in it.) Note that $\sum_{|M|=l} w(M) = \sum_{|M|=l-1} w(M) = 1$ and therefore (4.3) defines a probability distribution,

$$\sum_{\theta, M} \Pr_D(\theta_M) = \sum_{\theta} \Pr_{D'}(\theta^1) + \sum_{\theta} \Pr_{D'}(\theta^0) = 1.$$

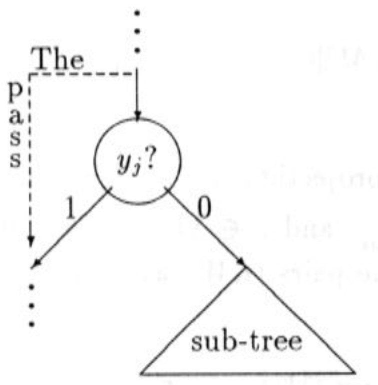
Now, let T be given. We do not define T' explicitly. Rather, we define a set of candidate deterministic decision trees and prove that the inequality in (4.2) holds for at least one of them. The candidates are the following $k \cdot \binom{k-1}{l-1}$ decision trees, $T_{(i,W)}$, indexed by pairs (i, W), where $i \in [k]$ and $W \in \{{[k] \setminus \{i\} \atop l-1}\}$.

$T_{(i,W)}$ is defined as the 'projection' of T under the following actions:

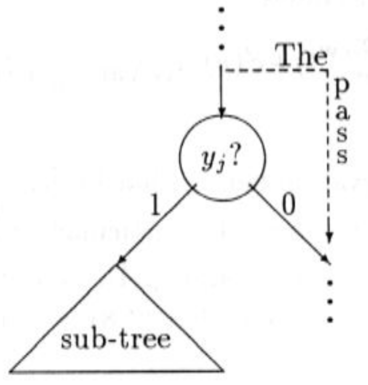
1. Each question 'y_i?' (in T) is replaced by the question 'v?' (in $T_{(i,W)}$).

2. For each $j \in W$, $T_{(i,W)}$ assumes that $y_j = 1$. Namely, for each node of T containing the question 'y_j?', $T_{(i,W)}$ passes down this question to the 1 direction while deleting that node and the whole sub-tree under the 0 direction (see Figure 4.3).

3. For all other j (i.e., $j \in [k] \setminus W$, $j \neq i$), $T_{(i,W)}$ assumes that $y_j = 0$: For each node of T containing 'y_j?', $T_{(i,W)}$ similarly passes down the question, this case to the 0 direction (see Figure 4.4).



Assuming $y_j = 1$ ($j \in W$)
Figure 4.3



Assuming $y_j = 0$ ($j \in W \setminus Y$, $j \neq i$)
Figure 4.4

It remains to show that the inequality of (4.2) holds for some $T_{(i,W)}$. We do this by proving that the following convex combination of these $k \cdot \binom{k-1}{l-1}$ inequalities holds.

$$\sum_{i\in[k], W\in\{\binom{[k]\setminus\{i\}}{l-1}\}} p_{(i,W)} \mathrm{E}_{\varepsilon'\in D'}[\mathrm{Time}_{c',T_{(i,W)}}(\varepsilon')] \leq \mathrm{E}_{\varepsilon\in D}[\mathrm{Time}_{c,T}(\varepsilon)], \qquad (4.5)$$

where the appropriate coefficients $\{p_{(i,W)}\}$ will be defined when used.

First, we write the explicit terms for the two expectations above:

$$\mathrm{E}_{\varepsilon'\in D'}[\mathrm{Time}_{c',T_{(i,W)}}(\varepsilon')] \overset{\mathrm{def}}{=} \sum_{\theta:X\to\{0,1\}} [\mathrm{Pr}_{D'}(\theta^1)\cdot \mathrm{Time}_{c',T_{(i,W)}}(\theta^1) + \mathrm{Pr}_{D'}(\theta^0)\cdot \mathrm{Time}_{c',T_{(i,W)}}(\theta^0)]$$

and

$$\mathrm{E}_{\varepsilon\in D}[\mathrm{Time}_{c,T}(\varepsilon)] \overset{\mathrm{def}}{=} \sum_{\theta:X\to\{0,1\}} \sum_{M\subseteq[k]} \mathrm{Pr}_D(\theta_M)\cdot \mathrm{Time}_{c,T}(\theta_M)] \overset{(4.3)}{=}$$

$$\sum_{\theta:X\to\{0,1\}} [\mathrm{Pr}_{D'}(\theta^1)\cdot \sum_{M\in\{\binom{[k]}{l}\}} w(M)\cdot \mathrm{Time}_{c,T}(\theta_M) + \mathrm{Pr}_{D'}(\theta^0)\cdot \sum_{M\in\{\binom{[k]}{l-1}\}} w([k]\setminus M)\cdot \mathrm{Time}_{c,T}(\theta_M)].$$

Inserting these terms into (4.5), and using the definition of D, we note that it is sufficient to show for each $\theta : X \to \{0,1\}$ that the following inequalities hold:

$$\sum_{i\in[k], W\in\{\binom{[k]\setminus\{i\}}{l-1}\}} p_{(i,W)} \mathrm{Time}_{c',T_{(i,W)}}(\theta^1) \leq \sum_{M\in\{\binom{[k]}{l}\}} w(M)\cdot \mathrm{Time}_{c,T}(\theta_M) \qquad (4.6)$$

and

$$\sum_{i\in[k], W\in\{\binom{[k]\setminus\{i\}}{l-1}\}} p_{(i,W)} \mathrm{Time}_{c',T_{(i,W)}}(\theta^0) \leq \sum_{M\in\{\binom{[k]}{l-1}\}} w([k]\setminus M)\cdot \mathrm{Time}_{c,T}(\theta_M) \qquad (4.7)$$

We prove (4.6), and (4.7) follows by duality: Change the roles of 1-s and 0-s in the Y variables and consider the threshold gate T^k_{k-l+1}.

Next, we divide 'Time' to the costs of X, Y and v, and use the notation above. Inequality (4.6) becomes

$$\sum_{(i,W)} p_{(i,W)}[c'(X^{T_{(i,W)}}_{\theta^1})+c'(v)\cdot 1_{v\in \mathrm{Var}_{T_{(i,W)}}(\theta^1)}] \leq \sum_{M\in\{{[k] \atop l}\}} w(M)[c(X^T_{\theta_M})+c(Y^T_{\theta_M})]. \quad (4.8)$$

The key observation here is that $\mathrm{Path}_{T_{(i,W)}}(\theta^1)$ is the 'projection' of $\mathrm{Path}_T(\theta_{\{i\}\cup W})$ under actions 1–3 above. In particular, $X^{T_{(i,W)}}_{\theta^1} = X^T_{\theta_{\{i\}\cup W}}$ and $v \in \mathrm{Var}_{T_{(i,W)}}(\theta^1)$ iff $y_i \in Y^T_{\theta_{\{i\}\cup W}}$. Using these and (4.4), and enumerating the pairs (i, W) as $\{(M,i) : M \in \{{[k] \atop l}\},\ i \in M\}$, we find that (4.8) is equivalent to

$$\sum_{M\in\{{[k] \atop l}\}} \sum_{i\in M} p_{(i,W)}\cdot[c'(X^T_{\theta_M})+c'(v)\cdot 1_{y_i\in Y^T_{\theta_M}}] \leq \frac{1}{\binom{k-1}{l-1}} \sum_{M\in\{{[k] \atop l}\}} \sum_{i\in M} \frac{c(y_i)}{c(Y)}\cdot[c(X^T_{\theta_M})+c(Y^T_{\theta_M})].$$

By definition, $c'(X^T_{\theta_M}) = c(X^T_{\theta_M})$. To cancel these terms out we now define $p_i = \frac{c(y_i)}{c(Y)}$ and $p_{(i,W)} = \frac{p_i}{\binom{k-1}{l-1}}$ for $i \in [k]$ and $W \in \{{[k]\setminus\{i\} \atop l-1}\}$. (Note that these coefficients are non-negative and their sum is 1.) Hence, canceling out and multiplying both sides by $\binom{k-1}{l-1}\cdot c(Y)$ reduces the last inequality to

$$c'(v)\cdot \sum_{M\in\{{[k] \atop l}\}} \sum_{i\in M} c(y_i)\cdot 1_{y_i\in Y^T_{\theta_M}} \leq \sum_{M\in\{{[k] \atop l}\}} \sum_{i\in M} c(y_i)\cdot c(Y^T_{\theta_M}),$$

which is equivalent to

$$c'(v)\cdot \sum_{M\in\{{Y \atop l}\}} c(M\cap Y^T_{\theta_M}) \leq \sum_{M\in\{{Y \atop l}\}} c(M)\cdot c(Y^T_{\theta_M}). \quad (4.9)$$

Note that we are now left with a problem involving the simple threshold sub-formula $F_{\mathrm{sub}} = T^k_l(y_1, ..., y_k)$. The only role θ plays in (4.9) is to determine some projection of T. This projection is derived from T by passing down each x_i?-question to the direction $\theta(x_i)$. Note that T may compute F without computing F_{sub}. This means that the projection is not a deterministic decision tree for F_{sub}. Rather, it may contain some leaves in which the value of the function being computed is not determined. We call such decision tree *partial.* The claim on partial decision trees given in the next section implies (4.9) and completes this proof. □

4.5 The Partial Decision Tree Claim

Definition 4.5.1 *A partial decision tree T for f is a generalization of a deterministic decision tree for f in that its leaves may contain '?'-s, not only 0-s and 1-s, but still, T is required to satisfy $Output_T(\varepsilon) = f(\varepsilon)$ for every ε with $Output_T(\varepsilon) \neq$ '?'.*

For example, the trivial decision tree, which contains a single node, a leaf, labeled by a '?', is a partial decision tree for every Boolean function. Figure 4.5 contains an example of partial decision tree computing the OR function of any variables set containing $\{z_1, ..., z_s\}$.

In the following claim T_0^k denotes the constant 1 function and T_{k+1}^k denotes the constant 0 function. Note that for the purpose of Lemma 4.4.1 and Theorem 1.3.3 we could state the following claim for $1 \le l \le k$ only, eliminating the somewhat artificial notations T_0^k and T_{k+1}^k. We do state it for $0 \le l \le k+1$ though, in order to simplify the proof.

Claim 4.5.2 *Let $0 \le l \le k+1$, let T be a partial decision tree for $T_l^k(y_1, ..., y_k)$ which does not probe a single variable more than once nor probes a variable that the function does not depend on, and let c be a leaf cost function on $Y = \{y_1, ..., y_k\}$. Then*

$$\frac{c(Y)}{2} \cdot \sum_{L \in \binom{Y}{l}} c(L \cap Var_T(1_L)) \le \sum_{L \in \binom{Y}{l}} c(L) \cdot c(Var_T(1_L)).$$

Before proving this claim we demonstrate its validity for the two intersting cases $l = k$ (AND gate) and $l = 1$ (OR gate). Note that these two cases could be the basis for an inductive proof of the weaker (but perhaps more intuitive) version of the claim in which $1 \le l \le k$.

Case 1: $l = k$ (AND gate).
The only $L \in \binom{Y}{k}$ is $L = Y$ and the inequality

$$\frac{c(Y)}{2} \cdot c(Y \cap \text{Var}_T(1_Y)) \le c(Y) \cdot c(\text{Var}_T(1_Y))$$

obviously holds.

Case 2: $l = 1$ (OR gate).
T does not probe a variable more than once. Hence, it is of the form of Figure 4.5, where $0 \le s \le k$ and $Z = \{z_1, ..., z_s\} \subseteq Y$.

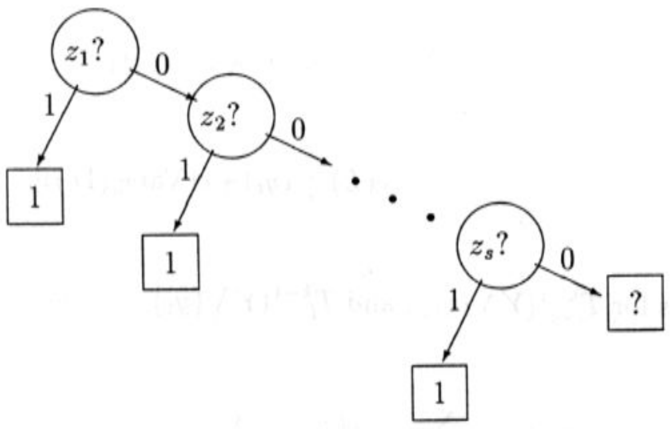


Figure 4.5: A partial decision-tree for OR

Denote $Y \setminus Z$ by $W = \{w_1, ..., w_{k-s}\}$. $L \in \{{Y \atop 1}\}$ consists of an element $y \in Y$. If y is some z_i then $\mathrm{Var}_T(1_L) = \{z_1, ..., z_i\}$, and if y is some w_i then $\mathrm{Var}_T(1_L) = Z$. We thus have to show:

$$\frac{c(Y)}{2} \cdot \sum_{i=1}^{s} c(z_i) \leq \sum_{i=1}^{s} [c(z_i) \cdot \sum_{j=1}^{i} c(z_j)] + [\sum_{i=1}^{k-s} c(w_i)] \cdot [\sum_{j=1}^{s} c(z_j)].$$

This indeed holds, since the quantity $\frac{1}{2}[\sum_{i=1}^{s} c(z_i)]^2 + [\sum_{i=1}^{k-s} c(w_i)] \cdot [\sum_{i=1}^{s} c(z_i)]$ is between the two values on both sides of the inequality, due to $\frac{1}{2}[\sum_{i=1}^{s} a_i]^2 \leq \sum_{i=1}^{s} [a_i \cdot \sum_{j=1}^{i} a_j]$.

Proof (of Claim 4.5.2): The proof is by induction on the structure of T. If T is trivial, i.e., it does not probe any variable, then for every L, $\mathrm{Var}_T(1_L)$ is empty, and the claim trivially holds.

Assume now that T is not trivial. In particular, by the assumption that T probes only variables that the function depends on, this means that the function is not a constant. Let 'y_t?' be the first probe in T, and let T_1 and T_0 be the subtrees under the directions $y_t = 1$ and $y_t = 0$, respectively (see Figure 4.6).

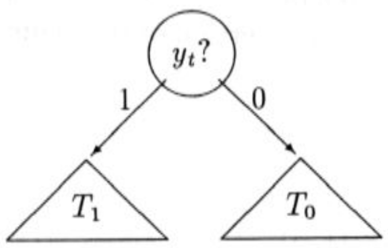


Figure 4.6: a non-trivial tree, T

For L containing y_t, say $L = \{y_t\} \cup L'$, we have $\mathrm{Var}_T(1_L) = \{y_t\} \cup \mathrm{Var}_{T_1}(1_{L'})$. For L not containing y_t, we have $\mathrm{Var}_T(1_L) = \{y_t\} \cup \mathrm{Var}_{T_0}(1_L)$. In these terms the claim states that

$$\frac{c(Y)}{2} \cdot \{ \sum_{L' \in \{{Y \setminus \{y_t\} \atop l-1}\}} [c(y_t) + c(L' \cap \mathrm{Var}_{T_1}(1_{L'}))] + \sum_{L \in \{{Y \setminus \{y_t\} \atop l}\}} c(L \cap \mathrm{Var}_{T_0}(1_L))\} \leq$$

$$\sum_{L' \in \{{Y \setminus \{y_t\} \atop l-1}\}} [c(y_t) + c(L')] \cdot [c(y_t) + c(\mathrm{Var}_{T_1}(1_{L'}))] + \sum_{L \in \{{Y \setminus \{y_t\} \atop l}\}} c(L) \cdot [c(y_t) + c(\mathrm{Var}_{T_0}(1_L))].$$

T_1 and T_0 are partial decision-trees for $T_{l-1}^{k-1}(Y \setminus \{y_t\})$ and $T_l^{k-1}(Y \setminus \{y_t\})$, respectively. Hence, by induction,

$$\frac{c(Y \setminus \{y_t\})}{2} \cdot \sum_{L' \in \{{Y \setminus \{y_t\} \atop l-1}\}} c(L' \cap \mathrm{Var}_{T_1}(1_{L'})) \leq \sum_{L' \in \{{Y \setminus \{y_t\} \atop l-1}\}} c(L') \cdot c(\mathrm{Var}_{T_1}(1_{L'})),$$

and

$$\frac{c(Y \setminus \{y_t\})}{2} \cdot \sum_{L \in \{{Y \setminus \{y_t\} \atop l}\}} c(L \cap \mathrm{Var}_{T_0}(1_L)) \leq \sum_{L \in \{{Y \setminus \{y_t\} \atop l}\}} c(L) \cdot c(\mathrm{Var}_{T_0}(1_L)).$$

Using these, and dividing by $c(y_t)$, the claim reduces to

$$\frac{c(Y)}{2} \cdot \binom{k-1}{l-1} + \frac{1}{2} \sum_{L'} c(L' \cap \mathrm{Var}_{T_1}(1_{L'})) + \frac{1}{2} \sum_{L} c(L \cap \mathrm{Var}_{T_0}(1_L))$$
$$\leq \binom{k-1}{l-1} \cdot c(y_t) + \sum_{L'} c(L') + \sum_{L'} c(\mathrm{Var}_{T_1}(1_{L'})) + \sum_{L} c(L),$$

and this holds due to

$$\frac{c(Y)}{2} \cdot \binom{k-1}{l-1} \quad = \quad \frac{1}{2} \cdot \sum_{L^* \in \{{Y \atop l}\}} c(L^*) \quad =$$

$$\frac{1}{2}\binom{k-1}{l-1} \cdot c(y_t) + \frac{1}{2} \cdot \sum_{L' \in \{{Y \setminus \{y_t\} \atop l-1}\}} c(L') + \frac{1}{2} \cdot \sum_{L \in \{{Y \setminus \{y_t\} \atop l}\}} c(L).$$

This completes the proofs of Claim 4.5, Lemma 4.4.1 and Theorem 1.3.3. □

4.6 Generalized Partial Decision Tree Claim

As pointed out by one of the referees of this thesis, the hypothesis of Claim 4.5.2 that the partial decision tree computes a threshold function is unnecessary, provided that in the conclusion the set L ranges over ALL subsets of the variable-set Y. This yields the following claim which might be interesting in its own right.

Claim 4.6.1 *Let T be a partial decision tree for any Boolean function depending on the variable-set Y which does not probe a single variable more than once, and let c be a leaf cost function on Y. Then*

$$\frac{c(Y)}{2} \cdot \sum_{L \subseteq Y} c(L \cap \mathit{Var}_T(1_L)) \leq \sum_{L \subseteq Y} c(L) \cdot c(\mathit{Var}_T(1_L)).$$

Proof: The proof follows the same lines of the proof of Claim 4.5.2. Formally, the proof for this claim is obtained from the previous one by
1. replacing $L' \in \{{Y \setminus \{y_t\} \atop l-1}\}$ by $L' \subseteq Y \setminus \{y_t\}$,
2. replacing $L \in \{{Y \setminus \{y_t\} \atop l}\}$ by $L \subseteq Y \setminus \{y_t\}$,
3. replacing $L^* \in \{{Y \atop l}\}$ by $L^* \subseteq Y$, and
4. replacing $\binom{k-1}{l-1}$ by $2^{|Y|-1}$. □

Chapter 5

Trials and Counter Examples

5.1 Deterministic Complexity for Read-Once Compositions

The following proposition and corollaries generalize the fact that read-once functions are evasive (Lemma 1.2.7). They state that the measure of deterministic decision tree complexity is additive if Boolean functions are composed in a read-once manner. Proposition 5.1.1 and Corollary 5.1.2 are used in Section 5.2. Analogue to Corollary 5.1.3 in the communication complexity model is a conjecture due to Karchmer et. al. [KRW91] whose affirmative answer would separate NC^1 from P.

Proposition 5.1.1 *Let f and g be two functions that depend on disjoint sets of input variables. Then $DC(f \vee g) = DC(f) + DC(g)$. In particular, an optimal deterministic decision tree for $f \vee g$ is obtained from an optimal tree for f by replacing each leaf that outputs 0 by an optimal tree for g.*

Proof: The construction described in the proposition's statement shows the $\leq$ direction. To show it is optimal interpret $DC(f)$ (or $DC(g)$) as a strategy against any algorithm. The strategy tells an answer for each probe in a way that only after $DC(f)$ probes the value of f is determined. The final answer can be either 0 or 1 leading to $f = 0$ or $f = 1$. Given such strategies for f and for g we construct that for $f \vee g$. Every sequence of probes is split into two sequences, one contains the variables of f and the other contains those of g. When a new variable x is probed, say it is a variable of f, we answer exactly like the f strategy would answer for this probe. If it determines f we answer in the way that causes $f = 0$. This new strategy forces $DC(f)$ probes of f-variables and $DC(g)$ probes of g-variables in order to determine the value of $f \vee g$. $\square$

The same proof idea yields Corollaries 5.1.2 and 5.1.3. It should be also clear how to construct optimal decision trees for the compound functions given those for their building blocks.

Corollary 5.1.2 *Let $f_1, f_2, ..., f_k$ be defined on disjoint sets of input variables. Then $DC(\vee(f_1, f_2, ..., f_k)) = DC(\wedge(f_1, f_2, ..., f_k)) = \sum_{i=1}^{k} DC(f_i)$.*

Corollary 5.1.3 *Consider $g : \{0,1\}^k \longrightarrow \{0,1\}$ and $f : \{0,1\}^l \longrightarrow \{0,1\}$. Define $g \circ f : \{0,1\}^{lk} \longrightarrow \{0,1\}$ on the variable set $\{x_{i,j}\}_{1\le i\le k, 1\le j\le l}$ by*

$$g \circ f(...x_{i,j}...) = g(f(x_{1,1}, ..., x_{1,l}),\ f(x_{2,1}, ..., x_{2,l}), ..., f(x_{k,1}, ..., x_{k,l})).$$

Then $DC(g \circ f) = DC(g) \cdot DC(f)$.

5.2 Minterms Oriented Decision Trees

Being interested in lower bounds in the decision tree model, we wish to know of properties that optimal decision trees satisfy. A candidate property is the following. We phrase it in terms of minterms of the function, but one may consider the dual — maxterms — as well.

Definition 5.2.1 *(minterms oriented decision tree) A deterministic decision tree T for a monotone function f is called minterms oriented if either*
(i) f is a constant $\nu \in \{0,1\}$ and T consists of a single leaf labeled ν, or
(ii) there is some minterm $\{x_1, x_2, ..., x_k\}$ of f with the following property. T probes first x_1, then, if $x_1 = 1$ is found, T probes x_2, then, if $x_2 = 1$ is found, T probes x_3 and so on. If x_k is eventually probed and found to be 1 too, T outputs 1. If 0 is found in some probe then the subtree under this branch is a minterms oriented decision tree for the corresponding restricted function.

For example, the decision tree constructed when proving that the nondeterministic decision tree complexity is at most quadratic in the deterministic decision tree complexity (Theorem 1.2.2) is minterms oriented. Following is another example.

Example 5.2.2 *(chain functions) Consider the function $f_t = \vee_{i=1}^{t} x_i x_{i+1}$ which depends on the $t+1$ variables $x_1, x_2 ... x_{t+1}$. Then f_t is evasive if $t \not\equiv 0 \pmod 3$, but if $t = 0 \pmod 3$, $DC(f_t) = t$ and this is achieved by a minterms oriented decision tree.*

Proof: The proof is by induction on t. f_1 and f_2 are clearly evasive. To see that $DC(f_3) \le 3$ probe x_3 first. If $x_3 = 0$ the restricted function is $x_1 \wedge x_2$. If $x_3 = 1$ the restricted function is $x_2 \vee x_4$. In any case only two additional probes are required.

That $DC(f_{t+1}) \ge DC(f_t)$ follows from the fact that f_t is the restriction of f_{t+1} under $x_{t+1} = 0$.

Now let $t = 3k$ where $k > 1$. Probe x_t first. If $x_t = 0$ the restricted function is f_{3k-2}. If $x_t = 1$ probe x_{t-1}, completing a minterm, and then probe x_{t+1}. The restricted function, if not a constant, is f_{3k-3} whose deterministic complexity is by induction $3k-4$. Hence, $DC(f_{3k} = \max\{1+3k-2,\ 3+3k-4\} = 3k-1$.

If $t = 1 \pmod 3$ and x_i is probed first, answer 0 iff $i = 0 \pmod 3$ and use induction and Proposition 5.1.1. If $t = 2 \pmod 3$ and x_i is probed first, answering 1 iff $i = 2 \pmod 3$ will do. $\square$

The notion of minterms oriented decision tree extends to non-monotone functions in the natural way. Fich [Fic91] gave the following example of a non-monotone Boolean function for which no minterms oriented decision tree is optimal. A v-tournament is a directed graph on v vertices in which for every pair of vertices there is exactly one directed edge connecting them. An input setting on $\binom{v}{2}$ variables can be viewed as a v-tournament; each variable value determines an edge direction.

Example 5.2.3 *(Fich) The function f of whether a v-tournament contains a sink or a source satisfies $CD(f) = \frac{7}{2}v + O(1)$ whereas every minterms oriented decision tree for this function has complexity $4v + \Omega(1)$.*

A natural question to ask here is whether every *monotone* function has an optimal decision tree which is minterms oriented:

Question 5.2.4 *Can the deterministic decision tree complexity be achieved by some minterms oriented decision tree for every monotone Boolean function?*

This, if correct, would be quite surprising and might help to design optimal decision trees. We can prove it for the special case of minterms size at most two, but do not know the answer in general.

Claim 5.2.5 *Suppose every minterm of f is of size at most two. Then there is a minterms oriented decision tree for f which achieves $DC(f)$.*

Proof: We prove it by induction on $DC(f)$. Let x be the first variable probed by some optimal decision tree for f. If $\{x\}$ is a minterm of f we are done by induction on $f|_{x=0}$, the restricted function of f under $x = 0$. Otherwise, let $\{x, y\}$ be a minterm of f. This means that $f|_{x=1}$ is $y \vee g$ where g depends on other variables. By Proposition 5.1.1, we can assume that the (optimal) subtree under $x = 1$ probes y first. By induction, we have optimal minterms oriented decision trees under the answers $x = 0$ and $y = 0$, hence the whole tree is minterms oriented and optimal. $\square$

As mentioned, we do not know of any monotone function for which no minterms oriented decision tree is optimal. There are many examples for which for *every* minterm there exists an optimal minterms oriented decision tree which starts by probing the variables of that minterm. However the following example shows that this stronger property is not true in general.

Example 5.2.6 *Let $f = x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1 + x_1x_3$. Then $DC(f) = 4$ whereas neither of the two variables in the minterm $\{x_4, x_5\}$ is the first probe in any optimal decision tree for f.*

Proof: Probe x_1 first (x_3 will also do). If it is 1 the restricted function is $\vee(x_2, x_3, x_5)$ whose complexity is 3. If it is 0 the restricted function is a chain of the type of Example 5.2.2 with $t = 3$ and its complexity is also 3. On the other hand, if x_2, x_4 or x_5 is probed first, say it is x_5, and its value is 1, then the restricted function is $\vee(x_1, x_2 \wedge x_3, x_4)$ whose complexity is by Proposition 5.1.2 1+2+1=4. □

A natural attempt for trying to answer Question 5.2.4 affirmatively is to use induction on the structure of an optimal decision tree for the function. Let T be an optimal decision tree for f and let x be the first variable it probes. Assume (by induction) that the subtree under $x = 1$ is minterms oriented with the minterm $m = \{x_1, x_2, ..., x_k\}$ probed first, and that the subtree under $x = 0$ is also minterms oriented. If $m \cup \{x\}$ happens to be a minterm of f we are done since T is already minterms oriented. Otherwise, we would like to slightly change T so that it becomes minterms oriented. We would like to delay the x-probe and claim that m might be the first minterm probed in another optimal decision tree for f. So we want to switch between x and the first variable in m probed by T. But we have to take care of the subtree of T under the $x = 0$ answer and make sure that its depth does not increase as a result of the switch. This raises the following question.

Question 5.2.7 *Let T be an optimal decision tree for f and let x be the first variable it probes. Let T_1 and T_0 be the subtrees of T under the answers $x = 1$ and $x = 0$ respectively. Suppose the first variable T_1 probes is y. Can T_0 be replaced by another decision tree for $F|_{x=0}$ whose first probed variable is also y and whose depth is at most the largest of that of T_1 and T_0?*

An affirmative answer would be sufficient for switching between x and y which is the first variable in m. But the answer is not true in general:

Example 5.2.8 *Consider the chain with $t = 6$ minterms, $f_6 = x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7$. There exists an optimal decision tree for f_6 whose first probe is x_5 and whose following probe under $x_5 = 1$ is x_1, but for every such decision tree the probe following $x_5 = 0$ is not x_1.*

Proof: The existence part follows from Example 5.2.2. If $x_5 = 0$ and $x_1 = 1$ the restricted function is $\vee(x_2, x_3 \wedge x_4, x_6 \wedge x_7)$, whose complexity is by Proposition 5.1.2, 1+2+2=5. □

5.3 The Non Optimality of The Saks-Wigderson Bounds

The example below shows that the Saks-Wigderson bounds are not tight: A simple read-once function f shows the following two gaps simultaneously.
(i) No directional randomized algorithm is optimal for this function, hence the Saks-Wigderson upper bound is higher than the randomized decision tree complexity of f.
(ii) The Saks-Wigderson lower bound is lower than the actual randomized decision tree complexity of f.

Actually, this example shows the non optimality of the Shirking lemma, which is the basic lemma in the Saks-Wigderson lower-bound, and whose generalizations are the basic steps in our Theorems 1.3.1 and 1.3.3 and in the new lower bound of Santha [San91] for Monte Carlo decision trees.

Example 5.3.1 *Consider the function $f = (x \wedge y) \wedge z$, and assume the following costs c_ν for probing each of its variables in case its value is $\nu \in \{0,1\}$,*

	x	y	z
c_0	*7*	*1*	*3*
c_1	*4*	*1*	*2*

Then f satisfies (i) and (ii) above.

Remark: One may claim that f is an artificial example in that it has non unit variable costs, and has AND gate as a child of other AND gate. However, this example is easily modified to a one without these 'weaknesses'.
To eliminate the non-unit costs, replace x by $(x_1 \vee x_2 \vee ... \vee x_7)$ and z by $(z_1 \vee z_2 \vee z_3)$ and argue about

$$f' = ((x_1 \vee x_2 \vee ... \vee x_7) \wedge y) \wedge (z_1 \vee z_2 \vee z_3)$$

in which all variables costs are 1.
To have alternating gates consider

$$f'' = (((g_1 \vee g_2 \vee ... \vee g_7) \wedge g_8) \vee w) \wedge (g_9 \vee g_{10} \vee g_{11})$$

where each g_i is an AND/OR-tree function with, say, 1000 input variables and where w is a single variable.

Proof of Example 5.3.1: We first show that $DC(f) = RC(f) = NC(f) = 7$. As for the nondeterministic decision tree complexity of f, the minterm $\{x, y, z\}$ and the maxterm $\{x\}$ show that $NC(f)$, $N_0(f)$, $N_1(f) \geq 7$. This value matches the deterministic decision tree complexity of f, $DC(f)$. Indeed, the *non-directional* algorithm which starts by probing x; then (if $x = 1$) probes z and then (if $z = 1$ too) probes y has cost 7 for every input setting. By Theorem 1.2.2, $DC(f) = RC(f) = NC(f) = 7$.

On the other hand the Saks-Wigderson upper and lower bounds for the randomized decision tree complexity of f differ from this value. To see this recall the recursive terms in these bounds (Theorems 1.2.8 and 1.2.9). In case of an AND-gate, $f = g \wedge h$ the upper bound for the 0-term is

$$u_0(f) = \max\{u_0(g), u_0(h), \frac{u_0(g)u_1(g) + u_0(h)u_1(h) + u_1(g)u_1(h)}{u_1(g) + u_1(h)}\}$$

and the lower bound for the 0-term is

$$R_0(f) = \min\{R_0(g)+R_1(h), R_1(g)+R_0(h), \frac{R_0(g)R_1(g) + R_0(h)R_1(h) + R_1(g)R_1(h)}{R_1(g) + R_1(h)}\}.$$

The 1-term for f is the same in both bounds. It is simply the sum of the 1-terms of g and of h. Note that in the two 0-terms there is also something in common. It is the third part subject to the maximization or minimization. Evaluating these on f yields

	x	y	z	$x \wedge y$	f
u_0	7	1	3	7	$\frac{51}{7} > 7$
u_1	4	1	2	5	7
R_0	7	1	3	5	$\frac{41}{7} < 7$
R_1	4	1	2	5	7

□

Finally, we would like to remark that if we ignore the maximization and the minimization in the Saks-Wigderson recurrent terms, and compute, instead, the third term only, then this simpler recursion gives exactly 7. Indeed, the following simpler recursion

$$R_0(f) = \frac{R_0(g)R_1(g) + R_0(h)R_1(h) + R_1(g)R_1(h)}{R_1(g) + R_1(h)}$$

(which simply ignores the first two terms in the minimization) evaluates on f exactly to 7.

We don't know of any example of read-once function for which this simpler recursion is not a lower bound for its randomized decision tree. In fact, we conjecture it is a lower bound. This better lower bound is worth knowing since it implies a non-recursive lower bound of $n^{0.66...}$ for read-once functions depending on n variables, as observed by Broder and Upfal [BU91].

The upper bound does not remain valid if its recursion is similarly simplified. This is clear by the example $(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \wedge y$. Evaluating the simplified recursion gives $\frac{5 \cdot 3 + 1 \cdot 1 + 3 \cdot 1}{3 \cdot 1} = 4.75$ for 0-inputs and 3+1=4 for 1-inputs while there is a maxterm of size 5.

Chapter 6

Open Problems

We conclude by mentioning some open problems related to the topics of this thesis. Many of them were suggested by others or mentioned above.

- Lower bounds for arbitrary functions.

  The most basic problem is perhaps to develop a lower bound technique for the randomized decision tree complexity of arbitrary Boolean functions. Is it true that the AND-OR tree function has the lowest randomized decision tree complexity in terms of its deterministic decision tree complexity among all Boolean functions? Can the lower bound of $RC \geq \sqrt{DC}$, which is known only through the nondeterministic argument (Theorem 1.2.2), be improved? Can the generalized claim on partial decision trees (Claim 4.6.1) be used for a lower bound involving arbitrary Boolean functions?

- Lower bounds in terms of circuit depth.

  Is it true that large gap between RC and DC cannot be achieved for TC^0 functions (and hence $NC^1 \neq TC^0$)? Is it true for AC^0 functions (which yields a new proof for $NC^1 \neq TC^0$)? What about the special case of depth two AC^0 functions?

- Improving the Read-once results.

  Can the Saks-Wigderson lower bound for read-once functions be improved? In particular, is it true that the simpler recursion mentioned in the end of Chapter 5 is a lower bound for the randomized decision tree complexity of read-once functions? Is it true that the AND-OR tree function has the lowest randomized decision tree complexity among all read-once Boolean functions? Can the $RC \geq n^{0.51}$ lower-bound for n-variable read once functions given in Chapter 3 be improved? Finally, concerning read-once threshold functions, is it true that the recursion given in Theorem 3.3.1 is a lower bound for the randomized decision tree complexity of this larger family?

- RC_1 versus RC_0.

  The randomized complexity for evaluating 1-s of the OR function and that for evaluating 0-s of the function differ by at most a factor of two. Is it true for every function? Is it true for every read-once function?

- Minterms oriented decision trees.

  Is the deterministic decision tree complexity achievable by a minterms oriented decision tree for every monotone function? What about the randomized decision tree complexity? Is it achievable by a distribution over minterms oriented decision trees only?

We have faced all these questions at some stage or another, found them interesting and attractive, and would be happy to hear about solutions of any of them.

Bibliography

[AHK89] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read once formulae with queries. Technical Report UCB CSD 89-528, U.C. Berkeley, 1989. To appear in JACM.

[Ajt83] M. Ajtai. $\sum_1^1$-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[BKRU91] A. Broder, A. Karlin, P. Raghavan, and E. Upfal. On the parallel complexity of evaluating game-trees. In *Proc. 2nd ACM-SIAM Symp. on Discrete Algorithms*, pages 404–413, 1991.

[Bop89] R. B. Boppana. *Amplification of probabilistic Boolean formulae*, volume 5 of *Advances in Computer Research.* JAI press, 1989.

[BU91] A. Broder and E. Upfal. Private communication, 1991.

[Fic91] F. Fich. Private communication, 1991.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[Gur77] V. A. Gurevich. On repetition-free Boolean functions. *Uspekhi Matematicheskikh Nauk*, 32(1):183–184, 1977. In Russian.

[Gur82] V. A. Gurevich. On the normal form of positional games. *Soviet Math. Dokl.*, 25(3):572–574, 1982.

[Haj90] P. Hajnal. On the power of randomness in the decision tree model. In *Proc. 5th Structure in Complexity Theory Conf.*, pages 66–77, 1990.

[Has88] J. Hastad. Almost optimal lower bounds for small depth circuits. *Advances in Computer Research*, 5 (Randomness and Computation), 1988.

[HMP$^+$87] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and Gy. Turán. Threshold circuits of bounded depth. In *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pages 99–110, 1987.

[HNW90] R. Heiman, I. Newman, and A. Wigderson. On read once threshold formulae and their randomized decision tree complexity. In *Proc. 5th Structure in Complexity Theory Conf.*, pages 78–87, 1990. To appear also in Theoretical Computer Science.

[HW91] R. Heiman and A. Wigderson. Randomized vs. deterministic decision tree complexity for read-once functions. In *Proc. 6th Structure in Complexity Theory Conf.*, pages 172–179, 1991. To appear also in J. Computational Complexity.

[Kin88] V. King. Lower bounds on the complexity of graph properties. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 468–476, 1988.

[KLN$^+$] M. Karchmer, N. Linial, I. Newman, M. Saks, and A. Wigderson. Combinatorial characterization of read-once formulae. To appear in Discrete Mathematics.

[KRW91] M. Karchmer, R. Raz, and A. Wigderson. On proving super-logarithmic depth lower bound via direct sum in communication complexity. In *Proc. 6th Structure in Complexity Theory Conf.*, 1991. To appear.

[KSS84] J. Kahn, M. Saks, and D. Sturtevant. A topological approach to evasiveness. *Combinatorica*, 4:297–306, 1984.

[LMN89] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 574–579, 1989.

[Mun89] D. Mundici. Functions computed by monotone Boolean formulas with no repeated variables. *J. Theoretical Computer Science*, 66:113–114, 1989.

[Nis89] N. Nisan. CREW PRAMs and decision trees. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 327–335, 1989.

[Pea82] J. Pearl. The solution for the branching factor of the alpha beta pruning algorithm and its optimality. *Communications of the ACM*, 25:559–564, 1982.

[Raz87] A. A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41:333–338, 1987.

[RV78] R. Rivest and S. Viullemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3:371–384, 1978.

[Sak86] M. Saks. Private communication, 1986.

[San91] M. Santha. On the Monte Carlo Boolean decision tree complexity of read-once formulae. In *Proc. 6th Structure in Complexity Theory Conf.*, pages 180–187, 1991.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. 19th ACM Symp. on Theory of Computing*, pages 77–82, 1987.

[Sni85] M. Snir. Lower bounds for probabilistic linear decision trees. *Theoretical Computer Science*, 38:69–82, 1985.

[SW86] M. Saks and A. Wigderson. Probabilistic Boolean decision trees and the complexity of evaluating game trees. In *Proc. 27th IEEE Symp. on Foundations of Computer Science*, pages 29–38, 1986.

[Tar83] M. Tarsi. Optimal search on some game trees. *Journal of the ACM*, 3:69–82, 1983.

[Val84] L. G. Valiant. Short monotone formulae for the majority function. *Journal of algorithms*, 5:363–366, 1984.

[Yao77] A. C. Yao. Probabilistic computations: towards a unified measure of complexity. In *Proc. 18th IEEE Symp. on Foundations of Computer Science*, pages 222–227, 1977.

[Yao85] A. C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 1–10, 1985.

[Yao87] A. C. Yao. Lower bounds to randomized algorithms for graph properties. In *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pages 393–400, 1987.

[Yao89] A. C. Yao. Circuits and Local Computation. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 186–196, 1989.