Electronic Colloquium on Computational Complexity, Report No. 9 (2002)

On determinism versus unambiquous nondeterminism for decision trees

Petr Savický *

Abstract

Let f be a Boolean function. Let $N(f) = \mathrm{dnf}(f) + \mathrm{dnf}(\neg f)$ be the sum of the minimum number of monomials in a disjunctive normal form for f and $\neg f$. Let $p(f)$ be the minimum size of a partition of the Boolean cube into disjoint subcubes such that f is constant on each of the subcubes. Let $\mathrm{dt}(f)$ be the minimum size (number of leaves) of a decision tree for f. Clearly, $\mathrm{dt}(f) \geq p(f) \geq N(f)$. It is known that $\mathrm{dt}(f)$ can be at most quasipolynomially larger than $N(f)$ and a quasipolynomial separation between $p(f)$ and $N(f)$ for a sequence of functions f is known. We present a quasipolynomial separation between $\mathrm{dt}(f)$ and $p(f)$ for another sequence of functions f.

1 Introduction

Let us consider binary decision trees, which test a single Boolean variable in each internal node. When the computation path reaches a leaf, the label assigned to the leaf is the value of the computed function. Since we consider Boolean functions, the leaves may be labeled by 0 or 1. The size of the tree is measured by the number of its leaves. Let $\mathrm{dt}(f)$ be the minimum size of a deterministic decision tree for f.

A nondeterministic decision tree is a decision tree, which can possibly contain nondeterministic nodes with an arbitrary number of successors. If a computation reaches such a node, it may continue into any of its successors. An input is accepted, if there is at least one accepting computation path for the input, i.e. a computation ending in a 1-leaf. By a co-nondeterministic tree for f, we understand a nondeterministic tree for $\neg f$.

The size of a nondeterministic tree is measured by the number of 1-leaves. Clearly, the minimal size of a nondeterministic tree for a Boolean function is equal to the minimum number of monomials in an equivalent disjunctive normal form formula. Let $\mathrm{dnf}(f)$ be the minimum number of monomials in a disjunctive normal form (dnf) for f and let $N(f) = \mathrm{dnf}(f) + \mathrm{dnf}(\neg f)$.

Clearly, the collection of all accepting resp. rejecting paths in a decision tree for a function f determines a dnf for f resp. $\neg f$. Hence, $N(f) \leq \mathrm{dt}(f)$.

*Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Praha 8, Czech Republic. The research was supported by the Ministry of Education of the Czech Republic - The research and development project LN00A056 "Institute of Theoretical Computer Science - ITI".

ISSN 1433-8092

The following theorem shows that the difference between $\mathrm{dt}(f)$ and $N(f)$ can be at most quasipolynomial.

Theorem 1.1 (Ehrenfeucht, Hausler, [1]) *Let f be a function of n variables. Then* $\mathrm{dt}(f) \leq 2^{O(\log n \log^2 N(f))}$.

The next theorem shows that a quasipolynomial separation may indeed be achieved.

Theorem 1.2 (Jukna et al., [2]) *There is a sequence of Boolean functions f_i with unbounded $N(f_i)$ such that* $\mathrm{dt}(f_i) \geq 2^{\Omega(\log^2 N(f_i))}$.

Let us denote, in the context of decision tree complexity, the classes of Boolean functions representable by polynomial size deterministic, nondeterministic and co-nondeterministic decision trees by P, NP and co-NP, respectively. Using this notation, Theorem 1.2 yields a separation between P and NP $\cap$ co-NP in the context of decision trees.

An unambiquous nondeterministic decision tree is a nondeterministic tree with at most one accepting path for every input. Such a tree corresponds to a dnf such that each input is accepted by at most one monomial. Let us call such a dnf an unambiquous dnf. Let $\mathrm{udnf}(f)$ be the minimum number of monomials in an unambiquous dnf for f.

Definition 1.3 For every Boolean function f, let $p(f) = \mathrm{udnf}(f) + \mathrm{udnf}(\neg f)$.

Note that $p(f)$ is also the minimum size of a partition of the cube $\{0,1\}^n$ into disjoint subcubes such that f is constant on each of these subcubes. Every decision tree for f determines such a partition of the cube into subcubes corresponding to the leaves of the tree. Hence, we have $\mathrm{dt}(f) \geq p(f) \geq N(f)$.

By inspecting the proof of Theorem 1.2 in [2], one can easily see that $\mathrm{dt}(f_i) \geq p(f_i) \geq 2^{\Omega(\log^2 N(f_i))}$ is proved there, although in a different notation. Hence, a quasipolynomial separation between $p(f_i)$ and $N(f_i)$ is possible for the sequence f_i used in [2]. The main result of the present paper is a quasipolynomial separation between $\mathrm{dt}(g_i)$ and $p(g_i)$ for some other sequence of Boolean functions g_i, see Theorem 3.7. Sections 2 and 4 contain additional related results.

2 A polynomial separation

In this section, we prove a polynomial separation between $\mathrm{dt}(f)$ and $p(f)$. By $\oplus$, we denote the addition modulo 2.

Lemma 2.1 *Let f_1, f_2 be functions of disjoint sets of variables. Then, we have* $\mathrm{dt}(f_1 \oplus f_2) \geq \mathrm{dt}(f_1)\mathrm{dt}(f_2)$.

Proof: The proof is done by induction on the number of variables. If one of the two functions is a constant, its dt size is 1 and the inequality is satisfied. Now, let both functions be nonconstant. Assume, we have a tree computing

$f_1 \oplus f_2$. W.l.o.g., assume that the variable x_i tested in the root belongs to f_1. By considering the tree size for the functions computed in the left and right subtrees separately, we obtain that the total size of the tree is at least $\mathrm{dt}(f_1|_{x_i=0} \oplus f_2) + \mathrm{dt}(f_1|_{x_i=1} \oplus f_2)$. By the induction hypothesis, this is at least $(\mathrm{dt}(f_1|_{x_i=0}) + \mathrm{dt}(f_1|_{x_i=1}))\mathrm{dt}(f_2) \geq \mathrm{dt}(f_1)\mathrm{dt}(f_2)$. □

This lemma can be used as follows. Find any function f, even on a small number of variables, such that $p(f) < \mathrm{dt}(f)$. Then, if $F_k = f^1 \oplus f^2 \oplus \ldots \oplus f^k$, where f^i are copies of f depending on disjoint sets of variables, we have $p(F_k) \leq p(f)^k$ and $\mathrm{dt}(F_k) \geq \mathrm{dt}(f)^k$. This implies

Theorem 2.2 *For every k, we have $\mathrm{dt}(F_k) \geq p(F_k)^\alpha$, where $\alpha = \frac{\ln \mathrm{dt}(f)}{\ln p(f)}$.*

For a concrete example, use e.g. $f(x,y,z) = xyz \vee \bar{x}\bar{y}\bar{z}$, for which we have $p(f) = 5$, $\mathrm{dt}(f) = 6$ and $\alpha = \frac{\ln 6}{\ln 5} \approx 1.113$.

3 A quasipolynomial separation

In order to construct functions G_i with a quasipolynomial difference between $p(G_i)$ and $\mathrm{dt}(G_i)$, let $g(x_1,x_2,x_3,x_4) = (x_1 \vee x_2 \vee x_3)x_4 \vee x_1x_2x_3$. For every i, we define a function G_i of 4^i variables as follows. Let G_0 be a single variable and for every $i \geq 0$, let $G_{i+1} = g(G_i^1, G_i^2, G_i^3, G_i^4)$, where the upper index distinguishes distinct copies of G_i on disjoint sets of variables. Clearly, G_i is a function of 4^i variables. Later, in Theorem 3.6, we prove $\mathrm{dt}(G_i) = \exp(\Omega(4^i))$. Together with the next theorem, this implies the required separation.

Theorem 3.1 *For every i, we have $p(G_i) \leq 2^{3^i}$.*

Proof: Consider the monomials

$$x_1\overline{x}_2x_4,\ x_2\overline{x}_3x_4,\ x_3\overline{x}_1x_4,\ x_1x_2x_3,$$

$$x_1\overline{x}_2\overline{x}_4,\ x_2\overline{x}_3\overline{x}_4,\ x_3\overline{x}_1\overline{x}_4,\ \overline{x}_1\overline{x}_2\overline{x}_3.$$

Note that the function g is constant on any subcube determined by any monomial in this list. This was used in [3] to prove that there is a partition of the cube of 4^i variables into disjoint subcubes determined by monomials of length 3^i such that G_i is constant on any of these subcubes. Clearly, the number of the monomials is 2^{3^i}. □

The proof of Theorem 3.7 consists of the next three lemmas. For any f, let $d(f)$ be the minimum depth of a decision tree for f.

Lemma 3.2 *If $h = f(g^1, g^2, \ldots, g^k)$, where $g^1, g^2, \ldots, g^k$ denote distinct copies of g depending on disjoint sets of variables, then $d(h) \geq d(f)d(g)$.*

Proof: The proof is done by an adversary strategy using the observation that for every i and g we have $d(g) \leq \max(d(g|_{x_i=0}), d(g|_{x_i=1})) + 1$. Assume, a variable is tested by the tree for h. This variable belongs to some of the occurrences of g. We choose the value of the variable that maximizes the depth of the obtained restriction of g. If both values lead to the same depth of the restriction and the restrictions are nonconstant, we can choose the value arbitrarily. If both restrictions are constant functions, we choose the value of the variable so that we maximize the decision tree depth for the corresponding restriction of f.

This strategy finds a path that determines the value of at least $d(f)$ occurrences of g and in each of them the strategy guarantees that we have to determine at least $d(g)$ variables. □

The strategy used in the proof of Lemma 3.2 finds a path leading to the last possible level in the tree. The proof of Lemma 3.5 shows that under some assumptions, we can find a lot of such paths.

Definition 3.3 For a nonconstant Boolean function f of n variables, let $r(f)$ be the minimum number of 1-leaves at the n-th level of a decision tree for f, i.e. after testing all variables. For a constant function f, let $r(f) = 0$.

Note that $r(f)$ is sensitive to adding formal variables. Namely, if f does not depend on all its variables then $r(f) = 0$. More generally, $r(f) = 0$ if and only if there is a tree for f of depth less than n. Note that every 1-leaf at the n-th level occurs in a pair with a 0-leaf. Hence, $r(f)$ is also the minimum number of such pairs.

Lemma 3.4 *Let g be a function of at least 2 variables. Then for every i, we have $r(g) \leq r(g|_{x_i=0}) + r(g|_{x_i=1})$.*

Proof: Since $g|_{x_i=a}$ depends on at least one variable, there exists a tree for $g|_{x_i=a}$ with at most $r(g|_{x_i=a})$ 1-leaves. By combining such trees for $a = 0$ and $a = 1$ using a root testing x_i, we obtain a tree proving the required upper bound on $r(g)$. □

Lemma 3.5 *Let f and $g_1, g_2, \ldots, g_k$ be Boolean functions and let $h = f(g_1, g_2, \ldots, g_k)$. Moreover, let $g_1, g_2, \ldots, g_k$ depend on disjoint sets of variables. Then $r(h) \geq r(f)r(g_1)r(g_2)\ldots r(g_k)$.*

Proof: The lemma will be proved by induction on the number of variables of h. W.l.o.g., we can assume that all functions $f, g_1, g_2, \ldots, g_k$ are functions of at least one variable, since otherwise $r(f)r(g_1)r(g_2)\ldots r(g_k) = 0$. If h is a function of one variable, then $k = 1$ and both f and g_1 are functions of one variable. If, moreover, $r(f)r(g_1) > 0$, then each of the functions f, g_1 and h is equal to its input variable or its negation. Hence, we have $r(h) = r(f)r(g_1) = 1$.

If h is a function of more than one variable, consider a tree for h with $r(h)$ 1-leaves at the last level. Assume, the root of the tree tests a variable from g_i. If g_i itself is a function of more than one variable, we use Lemma 3.4 to it. Let g_i' and g_i'' be the two restrictions of g_i by the root variable. By the induction hypothesis, the left and right subtree contribute to $r(h)$ at least $r(f)r(g_1)\dots r(g_{i-1})r(g_i')r(g_{i+1})\dots r(g_k)$ and $r(f)r(g_1)\dots r(g_{i-1})r(g_i'')r(g_{i+1})\dots r(g_k)$ respectively. The sum of these two contributions is at least $r(f)r(g_1)\dots r(g_k)$, since, we have $r(g_i') + r(g_i'') \geq r(g_i)$.

If g_i is a function of a single variable, then f is a function of at least two variables, since h is a function of at least two variables. We use Lemma 3.4 to the function f. The two restrictions of the root variable lead to the functions $f(g_1, \dots, g_{i-1}, 0, g_{i+1}, \dots, g_k)$ and $f(g_1, \dots, g_{i-1}, 1, g_{i+1}, \dots, g_k)$. Let f' and f'' be the corresponding restrictions of f. By the induction hypothesis, the contribution of the above two functions to $r(h)$ is at least $r(f')r(g_1)\dots r(g_{i-1})r(g_{i+1})\dots r(g_k)$ and $r(f'')r(g_1)\dots r(g_{i-1})r(g_{i+1})\dots r(g_k)$ respectively. Since $r(g_i) = 1$ and $r(f') + r(f'') \geq r(f)$, the two contributions together yield at least $r(f)r(g_1)\dots r(g_k)$ as required. □

Theorem 3.6 *For every i, we have* $\mathrm{dt}(G_i) \geq 2^{\frac{4^i+2}{3}}$.

Proof: By restricting any single variable in g to any constant, we obtain a read-once formula depending on three variables. Hence, in every decision tree for g, both subtrees connected to the root contain a 1-leaf. It follows that $r(g) = 2$. Using this and Lemma 3.5, we obtain by an easy induction on i that $r(G_i) \geq 2^{\frac{4^i-1}{3}}$. Since $\mathrm{dt}(G_i) \geq 2\,r(G_i)$, the theorem follows. □

Theorem 3.7 *For every i large enough, we have* $\mathrm{dt}(G_i) \geq 2^{\Omega(\log^\beta p(G_i))}$, *where* $\beta = \log_3 4 \approx 1.262$.

Proof: Combine Theorems 3.1 and 3.6. □

4 Iterated majority

Lemma 3.5 may also be used to prove a quasipolynomial separation between $p(f)$ and $N(f)$, although not as strong as needed for Theorem 1.2.

Let $\mathrm{MAJ}_3(x, y, z)$ be the majority on three variables. For every $i \geq 0$, let H_i be defined as follows. Let H_0 be a single variable and for every i let $H_{i+1} = \mathrm{MAJ}_3(H_i^1, H_i^2, H_i^3)$, where the upper index distinguishes distinct copies of H_i on disjoint sets of variables. Clearly, H_i is a function of 3^i variables.

Theorem 4.1 (see [2]) *For every i, we have* $\mathrm{dt}(H_i) \geq 2^{\frac{3^i+1}{2}}$.

This theorem together with an upper bound $N(H_i) \leq 2^{O(2^i)}$ was used in [2] to prove that $\mathrm{dt}(H_i) \geq 2^{\Omega(\log^\gamma N(H_i))}$, where $\gamma = \log_2 3 \approx 1.585$. We give an alternative proof of Theorem 4.1, using the technique developed in the current paper.

Proof: By setting any variable in MAJ_3 to a constant, we obtain a function depending on both the remaining variables. Hence, $r(\mathrm{MAJ}_3) = 2$. Using this and Lemma 3.5, we obtain $r(H_i) \geq 2^{\frac{3^i-1}{2}}$ by an induction on i. Since $\mathrm{dt}(H_i) \geq 2\,r(H_i)$, the theorem follows. $\square$

Acknowledgement. The author is indebted to R. Impagliazzo and P. Pudlák for encouraging discussions on the topic at early stages of the preparation of the paper.

References

[1] A. Ehrenfeucht, D. Haussler. Learning decision trees from random examples. *Information and Computation*, 82 (1989), pp. 231–246.

[2] S. Jukna, P. Savický, A. A. Razborov, I. Wegener. On P versus NP $\cap$ co-NP for decision trees and read-once branching programs. *Computational Complexity*, Vol. 8, No. 4 (1999), pp. 357–370.

[3] P. Savický, J. Sgall. On DNF tautologies with a limited number of occurrences of every variable. *Theoretical Computer Science*, Vol. 238, Issue 1-2 (2000), pp. 495–498.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'