

2010

Minimal Circuits for Very Incompletely Specified Boolean Functions

Richard Strong Bowen
Harvey Mudd College

Recommended Citation

Bowen, Richard Strong, "Minimal Circuits for Very Incompletely Specified Boolean Functions" (2010). *HMC Senior Theses*. 18.
https://scholarship.claremont.edu/hmc_theses/18

This Open Access Senior Thesis is brought to you for free and open access by the HMC Student Scholarship at Scholarship @ Claremont. It has been accepted for inclusion in HMC Senior Theses by an authorized administrator of Scholarship @ Claremont. For more information, please contact scholarship@cuc.claremont.edu.



Minimal Circuits for Very Incompletely Specified Boolean Functions

Richard Strong Bowen

Nicholas J. Pippenger, Advisor

Ran Libeskind-Hadas, Reader

May, 2010

HARVEY MUDD
COLLEGE

Department of Mathematics

Copyright © 2010 Richard Strong Bowen.

The author grants Harvey Mudd College the nonexclusive right to make this work available for noncommercial, educational purposes, provided that this copyright statement appears on the reproduced materials and notice is given that the copying is by permission of the author. To disseminate otherwise or to republish requires written permission from the author.

Abstract

In this report, asymptotic upper and lower bounds are given for the minimum number of gates required to compute a function which is only partially specified and for which we allow a certain amount of error. The upper and lower bounds match. Hence, the behavior of these minimum circuit sizes is completely (asymptotically) determined.

Contents

Abstract	iii
1 Introduction	1
1.1 Previous Work and Motivation	1
1.2 The Present Work	2
2 The Lower Bound	7
3 The Upper Bound	11
3.1 A Covering Lemma	11
3.2 Proof of the Upper Bound	12
4 Other Classes and Future Work	21
4.1 Counting	21
Bibliography	23

List of Figures

3.1	A block diagram describing the blocks of the circuit which proves the upper bound.	15
-----	--	----

List of Tables

- 2.1 A partially specified function which can be computed by the same circuit as Table 2.2. 7
- 2.2 A partially specified function which can be computed by the same circuit as Table 2.1. 8

- 3.1 Parameters of the blocks in the upper-bound construction. . . 16
- 3.2 Purpose of the blocks in the upper-bound construction. . . . 16

Chapter 1

Introduction

In this chapter, we will introduce the work, give definitions and general background, and state the theorem to be proved.

1.1 Previous Work and Motivation

One topic in circuit complexity theory is the minimum circuit (made up of some set of gates, such as AND, NOT, and OR gates) which computes a particular boolean function (i.e., a function from $\{0, 1\}^n$ to $\{0, 1\}$).

Livnat and Pippenger (2008) discuss some applications of circuit complexity theory to theoretical biology. Specifically, they consider theoretical model organisms with some *computational limitation*, which they model as some limit on the number of gates in a boolean circuit describing the organism's binary choices. They are interested in systematic mistakes—mistakes due to limitations in the circuit. They show that, for most functions, the best circuits (i.e., the smallest ones which are correct on all but a certain fixed fraction ε of input bitstrings) of a certain size depend on all their inputs, and conclude from this that most such circuits make systematic mistakes.

Sholomov (1969) considers incompletely defined boolean functions and bounds on their circuit sizes. A boolean function is said to be incompletely defined if it is only defined on a subset of $\{0, 1\}^n$, i.e., there are rows of its truth table which are marked by a “don't care” symbol. In the same paper, Sholomov also shows that as long as the number of specified rows (N_n) of the truth table is at least

$$n \log_2^{1+\delta} n,$$

for some positive δ , then as n grows the minimum number L of gates in a circuit for a function, for most circuits, grows as

$$L \sim \rho \frac{N_n}{\log_2 N_n},$$

where ρ is a constant depending on which gates are allowed in the circuit. In the sequel, for simplicity, we will consider only circuits composed of gates with at most two inputs, for which the constant ρ is always 1, and hence we will omit it. For more details on ρ , see for example Sholomov (1969).

Sholomov (1969) does not discuss the effects of allowable errors (the fraction ε in Livnat and Pippenger (2008)) on the complexity of circuit sizes. This is considered by Pippenger (1976), where the size of a circuit realizing a function having some fixed fraction of its inputs specified and allowed to make some fixed fraction of errors is asymptotically given. Unlike Sholomov (1969), Pippenger (1976) considers only circuits with a fixed fraction (i.e., a number growing exponentially) of specified rows in the truth table. In this thesis, we consider the more general combination of the two problems: we will allow a fixed fraction of errors (as considered by Pippenger) and a more general number of specified rows.

1.2 The Present Work

In this section, we present definitions and state the theorem to be proved.

1.2.1 Definitions

We begin with the definition of various kinds of boolean functions:

Definition 1.1. *A function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ is called a (n,m) -boolean function.*

Definition 1.2. *A function $f : \{0,1\}^n \rightarrow \{0,1,X\}^m$ is called a (n,m) -partially specified boolean function.*

Here the X in the range indicates an input for which the output is unspecified. In the case of both kinds of boolean functions, m may be omitted when it is one; both n and m may be omitted when their values are clear from context.

Definition 1.3. An (n, m) -boolean function f is a completion of an (n, m) -partially specified boolean function g if, for all input words $w \in \{0, 1\}^n$ where $g(w) \neq X$, $f(w) = g(w)$.

Next, we consider definitions for circuits. For more formal definitions, see for example Wegener (1987). We begin with the idea of a basis:

Definition 1.4. A basis for a circuit of size i is a set of $(n_i, 1)$ -ary functions.

Definition 1.5. A circuit over a particular basis is an acyclic network of gates and n inputs, where each gate computes some binary function in the basis; additionally, a list of m gates are specified as output gates.

Now we will give a notion of what it means for a function to compute a boolean function, and to compute a partially specified boolean function:

Definition 1.6. A circuit C computes an (n, m) -boolean function f if C has n inputs and m outputs, and for every word w in $\{0, 1\}^n$, $f(w)$ agrees with the output gates of the circuit when the inputs are set to agree with w .

Definition 1.7. A circuit C computes an (n, m) -partially specified function g if it computes some (n, m) -boolean function f and f is a completion of g .

Definition 1.8. A circuit computes an (n, m) - (perhaps partially specified) boolean function f with error E if it computes an (n, m) -function g which disagrees with f on at most a fraction E of all specified input words.

Finally, we give definitions which allow us to talk about complexity, by assigning a notion of resources (in this case, gates). We are concerned with circuits with minimally many gates, hence

Definition 1.9. If a circuit has n gates, then the size of the circuit is n .

Next we define the so-called Shannon function:

Definition 1.10. If f is a (perhaps partially specified) boolean function, then $L(f)$ is the size of the smallest circuit which computes it.

We can extend this to include a sense of computing with errors:

Definition 1.11. If f is a (perhaps partially specified) boolean function, then $L_E(f)$ is the size of the smallest circuit which computes it with error E .

There is a natural extension of the function L to sets of functions:

Definition 1.12. If B is a set of (perhaps partially defined) functions, then $L(B)$ (resp. $L_E(B)$) is the maximum of $L(f)$ (resp. $L_E(f)$) for $f \in B$.

Hence $L(B)$ is a tight upper bound for the number of gates needed to compute any function in B .

Finally, we will care about asymptotic behavior, so we need to define our asymptotic estimation:

Definition 1.13. We say $f(n) \gtrsim g(n)$ for functions f and g if

$$\liminf_{n \rightarrow \infty} \frac{f(n)}{g(n)} \geq 1.$$

Definition 1.14. We say $f(n) \lesssim g(n)$ for functions f and g if

$$\limsup_{n \rightarrow \infty} \frac{f(n)}{g(n)} \leq 1.$$

Definition 1.15. We say $f(n) \sim g(n)$ for functions f and g if

$$f \lesssim g \quad \text{and} \quad f \gtrsim g$$

1.2.2 Bounds for Circuit Classes

In this thesis, we are interested in how the complexity of computing a general partially specified circuit with allowed errors grows as the number of inputs grows. That is, if B_n is the set of all n -partially specified boolean functions with a particular number R_n of specified rows, we are interested in the asymptotic behavior of $L(B_n)$ or $L_E(B_n)$ for some fixed E . We may also be interested in the behavior of all functions with a particular property; for example, if M_n is the set of all n -boolean functions with at least half of their values equal to zero, then we might be interested in the asymptotic behavior of $L(M_n)$.

Lower Bounds

We are interested in asymptotic behavior, so by a lower bound, we mean a theorem like: as n goes to infinity, $L(B_n) \gtrsim q(n)$ for some function q .

In general, lower bounds in this field are proved by a counting argument (see for example Wegener (1987)). Two fully specified functions can't be computed by the same circuit, so to prove a lower bound we count the number of functions in the class, and the number of circuits with a number

n of elements. To get every function, we must at least increase n until the latter is greater than the former.

This is simply an upper bound for the most difficult function in the class. However, as long as the size of B_n is growing fast enough, in most cases almost all (that is, a fraction tending to one) of the functions in B_n will also need that many gates. This is known as the Shannon effect; see for example Wegener (1987).

Upper Bounds

We are interested in asymptotic behavior, so by a lower bound, we mean a theorem like: as n goes to infinity, $L(B_n) \lesssim q(n)$ for some function q

In general, upper bounds in this field are proved by explicit construction.

Main Theorem

The main theorem proved in this thesis is:

Theorem 1.1. *Let B_n be the set of all (n, R_n) -partially specified boolean functions, where $R_n \geq n \log_2^{1+\delta} n$ for some $\delta > 0$. Then*

$$L_E(B_n) \sim (1 - H(E)) \frac{R_n}{\log(R_n)}.$$

Here H is the binary entropy function

$$H(p) = -p \log p - (1 - p) \log(1 - p).$$

Here and in the sequel, \log means \log_2 .

Chapter 2

The Lower Bound

In this chapter, we prove the lower bound of the claimed theorem. We show this by means of a counting argument.

Theorem 2.1. *Let B_n be the set of partially specified boolean functions on n inputs. Then for large n , $L_E(B_n) \geq (1 - H(E)) \frac{R_n}{\log(R_n)}$.*

The key idea is that, to use the method of proving lower bounds described above, one needs only to consider those partially specified functions where the function is specified on the first R_n rows (lexicographically); this means we don't have to account for the fact that two functions can be computed by the same circuit—for example, the two partially specified functions given by the truth tables in Table 2.1 and Table 2.2 can be computed by the same circuit.

The lower bound follows from a counting argument. First we will estimate the number of partially specified functions with R_n specified rows (this will only be a lower bound, because we will only count those where the first R_n are specified). Second, we will use a theorem of Pippenger to

Input 1	Input 2	Output
0	0	0
0	1	1
1	1	X
1	1	X

Table 2.1: A partially specified function which can be computed by the same circuit as Table 2.2.

Input 1	Input 2	Output
0	0	X
0	1	X
1	1	1
1	1	0

Table 2.2: A partially specified function which can be computed by the same circuit as Table 2.1.

determine the smallest number of strings we could use and still be within some error of each of those. Finally, by using a standard argument, we will compute a lower bound on the size of the largest circuit for the class.

Let C_n be a minimal set of circuits for which, for every b in B_n , there is some circuit c in C_n which computes b to within the error E . We can view C_n as a set of strings in the usual lexicographic truth table encoding. Since these strings, restricted to the first R_n characters, E -approximate (in the sense of Pippenger (1976), §2.1.1) each of the 2^{R_n} possible strings of length R_n , but each E -approximates no more than $2^{H(E)R_n}$ of them (by Pippenger, 2.1.2-3, with $c = 0$), we have that the size of C_n is at least

$$2^{R_n} / 2^{H(E)R_n} = 2^{(1-H(E))R_n}.$$

Now we will prove a lower bound on $L_E(C_n)$ by a counting argument. This follows the arguments in Wegener (1987). Let $S(b, n)$ be the number of circuits using b or fewer gates and having n inputs. We have by Wegener (1987)

$$S(b, n) \leq (b + n + 1)^{2b} 16^b / b!.$$

Since we have shown that C_n has at least $2^{(1-H(E))R_n}$ different functions, we have that, if $b = L(C_n)$,

$$2^{(1-H(E))R_n} \leq |C_n| \leq S(b, n) \leq (b + n + 1)^{2b} 16^b / b!.$$

We can weaken this bound to make the algebra easier. For large n , we have that $b > n + 1$, and we can use $b! \geq (b/2)^b$ to get

$$S_{b,n} \leq (2b)^{2b} 4^{2b} / (b/2)^b.$$

Taking logs, we have (setting $q = (1 - H(E))$ for brevity)

$$\begin{aligned} qR_n &\leq 2b \log(2b) + 2b \log 4 - b \log b + b \log 2 \\ &= 2b(\log b + 1) + 8b - b \log b + b \\ &= b \log b + 11b \end{aligned}$$

Now we make the assumption that $b \leq \frac{(1-H(E))R_n}{\log((1-H(E))R_n)}$. The result of this assumption is

$$qR_n \leq \frac{qR_n}{\log qR_n} (\log qR_n - \log \log qR_n) + 11 \frac{qR_n}{\log qR_n},$$

which implies

$$0 \leq \frac{qR_n}{\log qR_n} (11 - \log \log qR_n),$$

which is clearly false. Hence, we have, for large n , $b \geq \frac{(1-H(E))R_n}{\log((1-H(E))R_n)}$, which proves the theorem.

Chapter 3

The Upper Bound

In this chapter, we will show the upper bound of the claimed theorem, that is that if f is a partially defined defined boolean function with R_n specified rows, having for large n

$$R_n \geq n \log_2^{1+\delta} n,$$

then for large n ,

$$L_E(f) \lesssim (1 - H(E)) \frac{R_n}{\log R_n}.$$

Together with the lower bound, this shows

$$L_E(f) \sim (1 - H(E)) \frac{R_n}{\log R_n}.$$

In order to show an upper bound, we give an explicit construction. This construction is similar to that in Sholomov (1969), and will follow his proof closely.

3.1 A Covering Lemma

First we show a covering lemma, similar to one proved in Sholomov (1969) but allowing for errors.

Lemma 3.1. *Let μ and v be integers, having $\mu < v$, and let $0 \leq E < 1$. Let S be the set of all words over $\{0, 1, X\}$ of size length v having exactly μ specified elements. Then there is a set T of words over $\{0, 1\}$ having the properties that:*

- *For every word s in S , we have a word t in T so that t disagrees with s on at most $E\mu$ of its specified position.*

- $|T| < 2^{\mu(1-H(E))}(1 + \mu + \nu)$.

where $H(E)$ is the binary entropy function.

Proof. We proceed by considering the words t over $\{0, 1\}^\nu$ and the words s over $\{0, 1, X\}^\mu$ as vertices of a bipartite graph where there is an edge between t and s if the first condition above is satisfied. We can see that each t will have degree

$$\binom{\nu}{\mu} \binom{\mu}{E\mu}.$$

By Pippenger (1976), Lemma 2.1.3-1, there is a cover T having

$$|T| \leq \frac{\binom{\nu}{\mu} 2^\mu}{\binom{\nu}{\mu} \binom{\mu}{E\mu}} \left(1 + \log \binom{\nu}{\mu} + \log \binom{\mu}{E\mu} \right).$$

By taking advantage of the fact that $\binom{a}{b} \leq 2^a$ we conclude

$$|T| \leq \frac{2^\mu}{\binom{\mu}{E\mu}} (1 + \nu + \mu),$$

and by Pippenger (1976), Lemma 2.1.2-1, we have $\binom{\mu}{E\mu} \geq 2^{H(E)\mu}$, hence

$$|T| \leq 2^{\mu(1-H(E))}(1 + \mu + \nu).$$

This cover gives exactly the set desired in the statement of the lemma. \square

3.2 Proof of the Upper Bound

In the paper Sholomov (1969), the author divides the proof of the upper bound into cases. We will follow those cases.

3.2.1 The First Case

The first case is when $\log R_n \sim n$.

Let f be an arbitrary (n, R_n) -partially specified boolean function. We must give a circuit which computes f . Given a parameter λ (the value of which is to be determined later), we can divide up the set $\{0, 1\}^n$ by

thinking of the function f as defined on a table A :

x_1	\cdots	$x_{\lambda-1}$	x_λ	0	0	0	$\sigma_{\lambda+1}$	\cdots	1	$x_{\lambda+1}$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
0	0	1	σ_{n-1}	\cdots	\cdots	\cdots	\cdots	\cdots	1	x_{n-1}
0	0	1	σ_n	\cdots	\cdots	\cdots	\cdots	\cdots	1	x_n
0	\cdots	0	0						\vdots	
0	\cdots	0	1						\vdots	
0	\cdots	1	0						\vdots	
\vdots	\vdots	\vdots	0						\vdots	
σ_1	\cdots	$\sigma_{\lambda-1}$	σ_λ	\cdots	\cdots	\cdots	\cdots	\cdots	$*$	
\vdots	\vdots	\vdots	\vdots							
1	1	1	1							

where at the position marked $*$ we have one of $\{0, 1, X\}$, that is, the value of $f(\sigma_1, \dots, \sigma_\lambda, \sigma_{\lambda+1}, \dots, \sigma_n)$. We subdivide this table in the following ways:

Columns A column A_i is one column of the table (the values of the function obtained by projection, that is, by fixing the last $n - \lambda$ variables).

Pieces We further subdivide the columns into pieces, which are contiguous and cover the entire column. Each piece in a column (excepting, perhaps, the last) has exactly μ specified elements, where μ is to be determined later, but subject to

$$\mu = o(\lambda).$$

Groups A piece has a starting position within a column, an ending position within a column, and a number of specified elements (which must be μ unless the piece is the last piece in the column). We place two pieces in the same *group* if they have all these things in common.

Next, we will define a completely specified boolean function g which approximates f to within an error E . The function g is most easily defined in view of a table B analogous to the table A for f . Each group in A is a set of strings of length at most 2^λ with at most μ specified elements. Hence by Lemma 3.1, we can fill in the table B with at most $2^{\mu(1-H(E))} (1 + 2^\lambda + \mu)$ different completions. The resulting table will agree with A on all but at most a fraction E of its specified entries.

The number of groups is bounded above by $2^{3\lambda}$, as there are no more than 2^λ choices for each of the defining characteristics (start position, end position, and number of specified elements). Hence the total number of completions a particular piece can have is at most

$$2^{3\lambda} 2^{\mu(1-H(E))} (1 + 2^\lambda + \mu) \leq 2^{\mu(1-H(E))+6\lambda},$$

and so can be encoded using $\mu(1 - H(E)) + 6\lambda$ bits. We will call the encoding of which completion a particular piece B_{ij} gets $\chi(B_{ij})$. We can encode a column by concatenating the encodings of its pieces:

$$\chi(B_i) = \chi(B_{i,1})\chi(B_{i,2}) \dots \chi(B_{i,m_i}),$$

and we can encode an entire function g by concatenating the encodings of its columns

$$\chi(g) = \chi(B_0)\chi(B_1) \dots \chi(B_{2^{n-\lambda}-1}).$$

We will call the length of this code h . There are at most $(R_n/\mu) + 2^{n-\lambda}$ pieces and the encoding of each piece has $(1 - H(E))\mu + 6\lambda$ bits, hence the length of this code is bounded by

$$h \leq ((R_n/\mu) + 2^{n-\lambda})((1 - H(E))\mu + 6\lambda)$$

bits. By virtue of $\lambda = o(\mu)$ and $(1 - H(E))$ being a constant, this gives us

$$h = O\left(R_n + 2^{n-\lambda}\mu\right)$$

Hence $\log h = O(n)$.

We will also be concerned with the length Q of the longest word $\chi(B_i)$. It's clear that in each column there are at most

$$W = \left\lceil \frac{2^\lambda}{\mu} \right\rceil$$

pieces, hence

$$Q \leq \left(\frac{2^\lambda}{\mu} + 1\right) (6\lambda + (1 - H(E))\mu).$$

We can see that $\log Q = O(n)$. We may now construct a circuit which computes g . We begin by describing the various blocks in the circuit briefly in Tables 3.1 and 3.2.

The pieces are assembled as indicated in Figure 3.1.

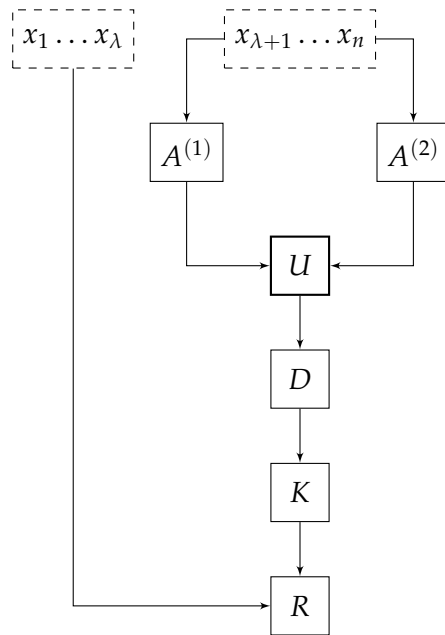


Figure 3.1: A block diagram describing the blocks of the circuit which proves the upper bound.

	Inputs	Outputs
$A^{(1)}$	$n - \lambda$	$\lceil \log h \rceil$
$A^{(2)}$	$n - \lambda$	$\lceil \log Q \rceil$
U	$\lceil \log h \rceil + \lceil \log Q \rceil$	$W(6\lambda + \mu(1 - H(E)))$
D	$W(6\lambda + \mu(1 - H(E)))$	$W2^\lambda$
K	$W2^\lambda$	2^λ
R	$2^\lambda + \lambda$	1

Table 3.1: Parameters of the blocks in the upper-bound construction.

	Purpose
$A^{(1)}$	Find the index of the start of $\chi(B_i)$
$A^{(2)}$	Find the length of $\chi(B_i)$
U	Compute the word $\chi(B_i)$.
D	Compute each $B_{i,j}$
K	Assemble the $B_{i,j}$ into B_i
R	Use the first λ inputs to select a particular output.

Table 3.2: Purpose of the blocks in the upper-bound construction.

The Blocks $A^{(1)}$ and $A^{(2)}$

These two blocks take the last $n - \lambda$ elements of the input word and compute the binary representations of the index of the start of the corresponding column in $\chi(g)$ and the length of the encoding of that column. It is a well known theorem of O.B. Lupanov (see, for example, Wegener (1987)) that an arbitrary function from a inputs to b outputs can be computed using g gates, where

$$g \lesssim \frac{b2^a}{a}$$

gates. Hence $A^{(1)}$ can be computed using

$$O\left(\frac{2^{n-\lambda}}{n-\lambda} \log h\right)$$

gates, and $A^{(2)}$ can be computed using

$$O\left(\frac{2^{n-\lambda}}{n-\lambda} \log Q\right)$$

gates.

We have both $\log h = O(n)$ and $\log Q = O(n)$, so the two A blocks together require $O\left(\frac{n2^{n-\lambda}}{n-\lambda}\right)$ gates.

The Block D

The block D consists of W sub blocks D_i , which decode the codes $\chi(B_{i,j})$ into a vector of length 2^λ , which is zero outside the piece $B_{i,j}$. Hence D_i can be computed using

$$O\left(\frac{2^{W(6\lambda+\mu(1-H(E)))}}{W(6\lambda+\mu(1-H(E)))}W2^\lambda\right)$$

gates, which by virtue of the fact that $W \leq 2^\lambda$ and $\lambda > \mu$ is

$$O\left(\frac{2^{7\lambda+\mu(1-H(E))}}{\mu}\right).$$

Hence the entire operator D requires no more than W times this many, or

$$O\left(\frac{2^{8\lambda+\mu(1-H(E))}}{\mu}\right)$$

gates.

The Block K

This block performs a large disjunction and clearly requires

$$O(W2^\lambda) = O(2^{2\lambda})$$

gates. The effect is to output the entire column B_i .

The Block R

This block is a selector—given λ bits (the selection) and 2^λ bits (the options), it uses the first λ bits (these being the first λ bits of the input) to choose one of the values from the options. We can see how many gates this takes by iteratively conditioning on the most significant bit. Let R_λ be the number of gates for a selector on $2^\lambda + \lambda$ inputs. It's clear that with a constant amount of circuitry and two copies of $R_{\lambda-1}$, we can condition on the most significant bit. Hence, the minimal size of this circuit is

$$L(R_\lambda) \leq c + 2L(R_{\lambda-1})$$

and so

$$L(R_\lambda) = O(2^\lambda).$$

The blocks K and R together require

$$O(2^{2\lambda})$$

gates.

The Block U

The block U does most of the actual work. We begin by dividing the code $\chi(g)$ into *chunks*, of size at most 2^λ . This chunk size is the same as the entropy of a completely unencoded column, so an encoded column spans at most two of them. The circuit for the operator U is composed of two parts: first, from the output of A_1 , which is the index of the column start in the code $\chi(g)$, we generate the code for chunk in which the code for the appropriate column B_i lies. Since this code might straddle chunks, we also generate the following chunk; by a construction due to Ulig, in Ulig (1974), this does not increase the complexity of the operator U . Second, using (small) selector circuits, we output just the column code $\chi(B_i)$. We refer to these two pieces as U_1 and U_2 , respectively. We will show that the complexity is, asymptotically,

$$L(U) \sim (1 - H(E)) \frac{R_n}{\log R_n}.$$

The number of chunks is at most $h/2^\lambda$; hence the number of inputs to U_1 is $\log \frac{h}{2^\lambda}$. The number of outputs is the maximum code length, which is Q . Hence the complexity of U_1 is bounded by

$$L(U_1) < \frac{\frac{R_n + 2^{n-\lambda}\mu}{2^\lambda} (2^\lambda + \mu) (1 - H(E))}{\log \frac{R_n + 2^{n-\lambda}\mu}{2^\lambda}}.$$

By virtue of $\mu < \lambda$, we have that $2^\lambda + \mu \sim 2^\lambda$. Hence, asymptotically,

$$L(U_1) \lesssim \frac{(R_n + 2^{n-\lambda}\mu) (1 - H(E))}{\log(R_n)}.$$

Now we show that

$$R_n + 2^{n-\lambda}\mu \sim R_n$$

by considering the fraction

$$\begin{aligned}
 \frac{R_n + 2^{n-\lambda}\mu}{R_n} &= 1 + \frac{\mu 2^n}{R_n 2^\lambda} \\
 &\rightarrow 1 + \frac{\mu 2^n}{2^\lambda 2^n} \quad (\text{Since } \log R_n \leq n) \\
 &\rightarrow 1 + \frac{\mu}{2^\lambda} \\
 &\rightarrow 1.
 \end{aligned}$$

Hence,

$$L(U_1) \lesssim (1 - H(E)) \frac{R_n}{\log R_n}.$$

Next we will consider the complexity of U_2 . It first computes the index of $\chi(B)$ in the chunk, which is to say it computes some $\log(h)$ -bit number mod a (fixed) 2^λ -bit number. Secondly, it shifts the pair of chunks to the left by this number so that $\chi(B_i)$ are the first bits of the shifter's output.

Computing the modulus can be done by a sort of binary search: compare the index to the starting index chunk halfway through the code; if appropriate, subtract off this index, then recurse. This requires a number of comparisons about equal to the log of the number of chunks; each comparison requires $O(\log h)$ gates. Hence this computation requires

$$\log(h) \log(h/2^\lambda) = O(n^2)$$

gates. Second, we must take the two chunks and shift them so that $\chi(B_i)$ is at the start. For this, we consider a gadget we call a *conditional shifter*. A conditional shifter for a permutation σ takes $k + 1$ inputs and has k outputs; if the last input (the condition) is high, then the σ_i th output is the i th input; otherwise it is the σ_i th input (that is, the shifter acts like a wire). This can be done in $O(k)$ gates. To shift by a number of bits described by a b -bit number, we need one shifter for each bit (taking advantage of the fact that a shift by $2^\alpha + 2^\beta$ is the same as a shift by 2^α followed by a shift by 2^β). Hence to shift the input pair of chunks to the left up to the length of one chunk, we have $b = k = \log 2^\lambda$, meaning we need λ^2 gates. Therefore the complexity of the operator U_2 is given by

$$L(U_2) = O(\lambda^2 + n^2).$$

The Total Complexity

The total complexity of the auxiliary operators (those which are not U) is

$$L' = O\left(\frac{n2^{n-\lambda}}{n-\lambda} + \frac{2^{8\lambda+(1-H(E))\mu}}{\mu} + 2^{2\lambda} + n^2 + \lambda^2\right).$$

Let $\Phi = \frac{2^n}{R_n}$. Then $\log(n\Phi) \sim n$. We can now choose values of μ and λ . We let $\lambda = \frac{\lceil 2\log(n\Phi) \rceil}{1-H(E)}$ and $\mu = \lceil \frac{n-17\log n\Phi}{1-H(E)} \rceil$. Then we have

$$\begin{aligned} L' &= O\left(\frac{n2^{n-2\log(n\Phi)}}{n-2\log(n\Phi)} + (1-H(E))\frac{2^{n-\log(n\Phi)}}{n-17\log(n\Phi)} + 2^{4\log(n\Phi)} + n^2\right) \\ &= O\left(2^{n-2\log n} + \frac{2^{n-\log n}}{n} + (n\Phi)^4 + n^2\right) \\ &= O\left(\frac{2^n}{n^2} + n^4\right) \end{aligned}$$

So, it is clear that

$$L' \frac{\log R_n}{R_n} \rightarrow 0.$$

3.2.2 Remaining Cases

In all the remaining cases, in which the conditions on R_n are changed, Sholomov's construction given in Sholomov (1969) produces a sub-block which recursively uses previous cases to compute new cases, whose complexity dominates the complexity of the entire circuit. Hence the above result carries through these results as well; for full detail see Sholomov (1969).

Chapter 4

Other Classes and Future Work

Other than incompletely specified functions, we considered some other classes of n -variable boolean functions. In the first class, we consider skewedness: the functions with at most $s2^n$ ones ($0 \leq s \leq 1$). The second class is the class of functions whose binary string representation (in lexicographic order of input strings) has at most $a2^n$ alternations ($0 \leq r \leq 1$). Finally we consider the class which has both these restrictions together.

The motivation for considering these classes is that, in some cases, the circuit complexity seems to exhibit a factoring behavior, that is: in some cases, almost all circuits having property P_1 will require $f_1 2^n / n$; almost all circuits having property P_2 will require $f_2 2^n / n$ gates, and almost all circuits having both properties P_1 and P_2 will require $f_1 f_2 2^n / n$ gates.

4.1 Counting

We begin by counting the number of functions over n inputs in each of the three classes. Of those functions which have $s2^n$ ones, there are $\binom{2^n}{s2^n}$. Of those functions with $a2^n$ alternations (here we assume the value $f(\vec{0}) = 0$, otherwise we count this as a position-0 alternation) there are $\binom{2^n}{a2^n}$. Finally, if we have both restrictions together, the behavior is a little more complicated. We will count all the strings with R runs and S ones. An *ordered k -partition* of n is an ordered list of k strictly positive integers which add to n . There are $\binom{n-1}{k-1}$ of these. In order to count the number of functions in the intersection class, we first fix the number of 1s in each run, followed by the number of 0s between them. This is a pair of ordered k -partitions - the runs uniquely determine an ordered R -partition of S , and the distribution of zeros determines an ordered $R + 1$ -partition of $2^n - S + 2$. Hence the

total number of such functions is

$$\binom{S-1}{R-1} \binom{2^n - S + 1}{R}.$$

The conversion between (S, R) and (s, a) is straightforward. These functions were chosen to study the interaction of two different circuit properties; that all three lower bounds are readily calculable is promising for their study for this purpose. The further study of the behavior of these classes is a prime target for future study.

Bibliography

Livnat, A., and N. Pippenger. 2008. Systematic mistakes are likely in bounded optimal decision-making systems. *Journal of Theoretical Biology* 250(3):410–423.

Pippenger, N. 1976. Information theory and the complexity of Boolean functions. *Mathematical System Theory* 10(1):129–167.

Sholomov, L.A. 1969. On the realization of incompletely-defined Boolean functions by circuits of functional elements. *Problemy Kibernetiki* 10:215–226. Trans: System Theory Research, 21 (1969) 211-223.

Ulig, D. 1974. On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements. *Mathematical Notes* 15(6):558–562.

Wegener, I. 1987. *The Complexity of Boolean Functions*. Teubner.