New Constructions with Quadratic Separation between Sensitivity and Block Sensitivity

Siddhesh Chaubal
University of Texas at Austin, USA
siddhesh@cs.utexas.edu

Anna Gál
University of Texas at Austin, USA
panni@cs.utexas.edu

Abstract

Nisan and Szegedy [14] conjectured that block sensitivity is at most polynomial in sensitivity for any Boolean function. There is a huge gap between the best known upper bound on block sensitivity in terms of sensitivity – which is exponential, and the best known separating examples – which give only a quadratic separation between block sensitivity and sensitivity.

In this paper we give various new constructions of families of Boolean functions that exhibit quadratic separation between sensitivity and block sensitivity. Our constructions have several novel aspects. For example, we give the first direct constructions of families of Boolean functions that have both 0-block sensitivity and 1-block sensitivity quadratically larger than sensitivity.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Sensitivity Conjecture, Boolean Functions, Complexity Measures

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.13

1 Introduction

The Sensitivity Conjecture posed by Nisan and Szegedy [14] is one of the most intriguing, yet elusive problems in computational complexity theory.

The *sensitivity* $s(f)$ of a Boolean function f is the maximum over all inputs x of the number of coordinate positions i such that changing the value of the i-th bit of x changes the value of the function. The *block sensitivity* $bs(f)$ of a Boolean function f is the maximum over all inputs x of the number of disjoint blocks of bits such that changing the value of all bits of x in any given block changes the value of the function. (See Section 2 for more formal definitions.) Sensitivity was introduced by Cook, Dwork and Reischuk [8] as a measure to prove lower bounds on the parallel complexity of Boolean functions in the CREW PRAM model. Nisan [13] defined the more general block sensitivity measure, and showed that the CREW PRAM complexity of any Boolean function f is characterized by its block sensitivity up to constant factors as $\Theta(\log bs(f))$. Nisan also showed that several other complexity measures, including certificate complexity and decision tree depth are polynomially related to block sensitivity. Nisan and Szegedy [14] showed that the degree of real polynomials representing a Boolean function f is also polynomially related to its block sensitivity. These relations extend to approximate representation by real polynomials and to randomized and quantum decision tree depth. Thus, a number of important complexity measures are polynomially related to block sensitivity. See [6, 11] for a survey.

However, it remains open to fully understand the relationship between sensitivity and block sensitivity. Of course for any Boolean function f, $s(f) \le bs(f)$. Nisan and Szegedy [14] conjectured that block sensitivity is at most polynomial in sensitivity for any Boolean

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 13; pp. 13:1–13:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

function f. They even raised the possibility that $bs(f) = O(s(f)^2)$. This possibility is still not ruled out - the best separation so far remains quadratic. The current best upper bound on block sensitivity in terms of sensitivity by Ambainis et al. [2, 4] is exponential: $bs(f) \leq s(f)2^{s(f)-1}$. (More precisely, $bs(f) \leq \max\{2^{s(f)-1}(s(f) - \frac{1}{3}), s(f)\}$ [4].) This improves the earlier upper bounds of Kenyon and Kutin and Simon [12, 16].

The first example of a function with quadratic separation between its sensitivity and block sensitivity was given by Rubinstein [15] who constructed a function f with $bs(f) = \frac{1}{2}s(f)^2$. Other constructions with quadratic separation were given in [19, 7, 10, 5]. The largest separation so far is achieved by the construction of Ambainis and Sun [5] who gave a function f with $bs(f) = \frac{2}{3}s(f)^2 - \frac{1}{3}s(f)$.

Improving the constant $\frac{2}{3}$ in the separation would be interesting, since a function f with $bs(f) > cs(f)^2$ for a constant $c > 1$ would imply a construction with superquadratic separation by iterated composition of the function f [3].

In order to better understand the relationship between sensitivity and block sensitivity, the one-sided versions of the measures 0-sensitivity $s_0(f)$, 1-sensitivity $s_1(f)$, 0-block sensitivity $bs_0(f)$ and 1-block sensitivity $bs_1(f)$ have also been extensively studied. These measures are obtained by restricting attention to inputs $x \in f^{-1}(0)$ for defining 0-sensitivity and 0-block sensitivity and to inputs $x \in f^{-1}(1)$ for defining 1-sensitivity and 1-block sensitivity, respectively. (See Section 2 for formal definitions.) Then $s(f) = \max\{s_0(f), s_1(f)\}$ and $bs(f) = \max\{bs_0(f), bs_1(f)\}$.

Ambainis and Prusis [3] (improving the constant of a statement in [12]) proved that $bs_0(f) \leq \frac{2}{3}s_0(f)C_1(f)$ where $C_1(f)$ denotes the 1-certificate complexity of f. See Section 2 for the definition of certificate complexity. On the other hand, [13] proved that $C_1(f) \leq bs_1(f)s_0(f)$. The analogous statements also hold for upper bounding bs_1 and C_0, respectively. Combining these results implies that in order to obtain much stronger separation between sensitivity and block sensitivity it is necessary to construct functions f such that both $bs_0(f)$ and $bs_1(f)$ are significantly larger than $s(f)$.

Avishay Tal [18] pointed out to us, that one can get such examples by the following trick. Let g be any function with $bs(g) = \Omega(s(g)^2)$, then taking $f(x, y) = g(x) \vee \neg g(y)$ will give $\min\{bs_0(f), bs_1(f)\} = \Omega(s(f)^2)$. Notice however that in this example the function f will not give an asymptotically larger separation between its block sensitivity and sensitivity than what was achieved by the function g unless $bs_1(g) = \theta(bs_0(g))$. Thus, limitations on the separation that follow from properties of the function g will be inherited by the function f. By direct constructions, the largest simultaneous separation has been $\min\{bs_0(f), bs_1(f)\} = \Omega(s(f)^{\log_2 3})$ in [1]. On the other hand, all previous direct constructions with quadratic separation between $bs(f)$ and $s(f)$ had $\min\{bs_0(f), bs_1(f)\} = O(s(f))$.

1.1 Our Results

In this paper we give various new constructions of families of Boolean functions that exhibit quadratic separation between sensitivity and block sensitivity. Our constructions have several novel aspects.

We provide the first direct constructions of families of Boolean functions f with $\min\{bs_0(f), bs_1(f)\} = \Omega(s(f)^2)$. Our simultaneous quadratic separation of both 0-block sensitivity and 1-block sensitivity from sensitivity is based on a more refined study of the effects of function composition on these measures. We also present sufficient conditions for achieving such simultaneous separations and give several examples of functions satisfying these conditions.

All previous constructions - with the exception of Chakraborty's functions [7] - were of the form $f = OR_m \circ g_k$ that is $f : \{0,1\}^{mk} \to \{0,1\}$ was obtained by composing the m-bit OR function with an appropriately chosen inner function g on k bits. Chakraborty [7] did not use function composition at all. As for the choice of the inner function, [10] defined the inner function g based on codewords of a Hamming code. All other constructions (including Chakraborty [7]) used the presence of certain patterns in the input x to set the function value $g(x)$ to 1.

We observe that other function compositions instead of OR-composition can also yield quadratic separations. We define new functions, that could be used as inner or outer functions, based on algebraic criterions related to multiplication in finite fields or polynomial multiplication.

We also give new examples of functions to match the current best constant of $\frac{2}{3}$ by Ambainis and Sun [5] among the known quadratic separations. We give a general condition for achieving quadratic separations with the $\frac{2}{3}$ constant for functions defined by families of certificates. The function by Ambainis and Sun [5] fits into this framework.

2 Preliminaries

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. For $x \in \{0,1\}^n$ and $i \in [n]$ we denote by x^i the input obtained by flipping the i-th bit of x. More generally, for $S \subseteq [n]$ we denote by x^S the input obtained by flipping the bits of x in all coordinates in the subset S.

▶ **Definition 1** (Sensitivity). The sensitivity $s(f,x)$ of a Boolean function f on input x is the number of coordinates $i \in [n]$ such that $f(x) \neq f(x^i)$. The 0-sensitivity and 1-sensitivity of f are defined as $s_0(f) = \max\{s(f,x) : f(x) = 0\}$ and $s_1(f) = \max\{s(f,x) : f(x) = 1\}$, respectively. The sensitivity of f is defined as $s(f) = \max\{s(f,x) : x \in \{0,1\}^n\} = \max\{s_0(f), s_1(f)\}$.

▶ **Definition 2** (Block Sensitivity). The block sensitivity $bs(f,x)$ of a Boolean function f on input x is the maximum number of pairwise disjoint subsets $S_1, \ldots, S_k$ of $[n]$ such that for each $i \in [k]$ $f(x) \neq f(x^{S_i})$. The 0-block sensitivity and 1-block sensitivity of f are defined as $bs_0(f) = \max\{bs(f,x) : f(x) = 0\}$ and $bs_1(f) = \max\{bs(f,x) : f(x) = 1\}$, respectively. The block sensitivity of f is defined as $bs(f) = \max\{bs(f,x) : x \in \{0,1\}^n\} = \max\{bs_0(f), bs_1(f)\}$.

It is convenient to refer to coordinates $i \in [n]$ such that $f(x) \neq f(x^i)$ as *sensitive bits* for f on x. Similarly, a subset $S \subseteq [n]$ is called a *sensitive block* for f on x if $f(x) \neq f(x^S)$.

▶ **Definition 3** (Partial assignment). Given an integer $n > 0$, a partial assignment α is a function $\alpha\colon [n] \to \{0,1,\star\}$. A partial assignment α corresponds naturally to a setting of n variables $(x_1, x_2, \ldots x_n)$ to $\{0,1,\star\}$ where x_i is set to $\alpha(i)$. The variables set to $\star$ are called unassigned or free, and we say that the variables set to 0 or 1 are fixed.
We say that $x \in \{0,1\}^n$ agrees with α if $x_i = \alpha(i)$ for all i such that $\alpha(i) \neq \star$.
The size of a partial assignment α is defined as the number of fixed variables of α.

▶ **Definition 4** (Certificate). For a function $f\colon \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$ a partial assignment α is a certificate of f on x if x agrees with α and any input y agreeing with α satisfies $f(y) = f(x)$.
The size of a certificate α is defined as the size of the partial assignment α.

▶ **Definition 5** (Certificate Complexity). The certificate complexity $C(f,x)$ of a Boolean function f on input x is the size of the smallest certificate of f on x. The 0-certificate complexity and 1-certificate complexity of f are defined as $C_0(f) = \max\{C(f,x) : f(x) = 0\}$ and $C_1(f) = \max\{C(f,x) : f(x) = 1\}$, respectively. The certificate complexity of f is defined as $C(f) = \max\{C(f,x) : x \in \{0,1\}^n\} = \max\{C_0(f), C_1(f)\}$.

▶ **Definition 6** (Function defined by a set of partial assignments). Let $C = \{c_1, c_2, \dots c_t\}$ be a set of partial assignments $c_1, c_2, \dots c_t \colon [n] \to \{0, 1, \star\}$.
Then C naturally defines a function $g_C \colon \{0,1\}^n \to \{0,1\}$ as:
$g_C(x) = 1$ iff x agrees with some partial assignment $c_i \in C$.

▶ **Definition 7** (Distances). The distance between two inputs $x, y \in \{0,1\}^n$ is defined as the number of bits in which they differ.
The distance between an input $x \in \{0,1\}^n$ and a partial assignment $\alpha \colon [n] \to \{0,1,\star\}$ is defined as the minimum distance between x and any input y agreeing with α.
The distance between two partial assignments $\alpha, \beta \colon [n] \to \{0,1,\star\}$ is defined as the minimum distance between any input x agreeing with α and any input y agreeing with β.

▶ **Definition 8** (Function Composition). For Boolean functions $f : \{0,1\}^m \to \{0,1\}$ and $g : \{0,1\}^k \to \{0,1\}$ the function $f \circ g : \{0,1\}^{mk} \to \{0,1\}$ is defined on $z \in \{0,1\}^{mk}$ as

$$f \circ g(z) = f(g(z_1, \dots z_k), g(z_{k+1}, \dots, z_{2k}), \dots, g(z_{(m-1)k+1}, \dots, z_{mk}))$$

Properties of function composition were formally studied with respect to sensitivity and block sensitivity (as well as other related measures) in [17, 9]. We note the following two properties, relevant for us.

▶ **Lemma 9.** *[17, 9] For any Boolean functions f and g we have $s(f \circ g) \le s(f)s(g)$.*

▶ **Definition 10.** [17] For $z \in \{0,1\}$ we say that $f : \{0,1\}^n \to \{0,1\}$ is in z-good form, if (1) $f(z^n) = z$ and (2) $bs(f) = bs(f, z^n)$

▶ **Lemma 11.** *[17] If both f and g are in 0-good form, or if both f and g are in 1-good form, then $bs(f \circ g) \ge bs(f)bs(g)$.*

2.1 Previous Constructions with Quadratic Separation

All previous constructions that achieve quadratic separation between sensitivity and block sensitivity - with the exception of Chakraborty's functions [7] - were based on the following “OR-composition Lemma” first used by Rubinstein [15].

▶ **Lemma 12.** *[15] For any function $g \colon \{0,1\}^m \to \{0,1\}$, we have:*

- $s_0(OR_n \circ g) = ns_0(g)$
- $bs_0(OR_n \circ g) = nbs_0(g)$
- $s_1(OR_n \circ g) = s_1(g)$
- $bs_1(OR_n \circ g) = bs_1(g)$

The quadratic separations of [15, 19, 5, 10] are based on using this lemma and considering functions of the form $f = OR_n \circ g$ for appropriately chosen inner functions g.

Next we briefly describe the previous constructions of functions with quadratic separation.

1. Rubinstein's function [15] Define $g \colon \{0,1\}^{2m} \to \{0,1\}$ as:
   $g(x) = 1$ iff $x_{2j\ -1} = x_{2j} = 1$ for some $j \in [m]$ and $x_i = 0$ for $i \ne 2j-1, 2j$.
   This gives $s_0(g) = 1$, $s_1(g) = bs_1(g) = 2m$, $bs_0(g) = m$.
   Let $f = OR_{2m} \circ g$. Then $s_0(f) = s_1(f) = bs_1(f) = 2m$, $bs_0(f) = 2m^2$, giving $bs(f) = \frac{1}{2}\ s(f)^2$.
2. Virza's function [19] Define $g \colon \{0,1\}^{2m+1} \to \{0,1\}$ as:
   $g(x) = 1$ iff one of the following holds:

1) $\exists j \in [m]$ such than $(x_{2j-1} = x_{2j} = 1)$ and $(x_i = 0 \ \ \forall\, i \neq 2j-1, 2j)$.
2) $(x_{2m+1} = 1)$ and $(x_i = 0 \,\forall\, i \neq 2m+1)$.
This gives $s_0(g) = 1$, $s_1(g) = bs_1(g) = 2m+1$, $bs_0(g) = m+1$.
Let $f = OR_{2m+1} \circ g$. Then $s_0(f) = s_1(f) = bs_1(f) = 2m+1$, $bs_0(f) = (m+1)(2m+1)$.
Therefore, $bs(f) = \frac{1}{2}s(f)^2 + \frac{1}{2}s(f)$.

3. Ambainis and Sun's function [5] Define $g\colon \{0,1\}^{2(2m+1)} \to \{0,1\}$ as:
$g(x) = 1$ iff $\exists j \in [2m+1]$ such that:
1) $x_{2j-1} = x_{2j} = 1$, and
2) For all $i \in [m]$, $x_{2j+2i} = x_{2j-2i} = x_{2j-2i-1} = 0$.
Here, the index of x is taken modulo $(2(2m+1))$ i.e. we index x as if it were laid around a circle.
This gives $s_0(g) = 1$, $s_1(g) = bs_1(g) = 3m+2$, $bs_0(g) = 2m+1$.
Let $f = OR_{3m+2} \circ g$. Then $s_0(f) = s_1(f) = bs_1(f) = 3m+2$, $bs_0(f) = (3m+2)(2m+1)$.
Therefore, $bs(f) = \frac{2}{3}s(f)^2 - \frac{1}{3}s(f)$.

4. Function based on Hamming Code [10]:
Consider the hamming code on $m = 2^r - 1$ bits.
Define $g\colon \{0,1\}^m \to \{0,1\}$ as:
$g(x) = 1$ iff x is a codeword of the hamming code on m bits.
This gives $s_0(g) = 1$, $s_1(g) = bs_1(g) = m$, $bs_0(g) = \frac{m+1}{2}$.
Let $f = OR_m \circ g$. Then, $s_0(f) = s_1(f) = bs_1(f) = m$, $bs_0(f) = \frac{m(m+1)}{2}$. Thus $bs(f) = \frac{1}{2}s(f)^2 + \frac{1}{2}s(f)$.

Finally, we describe a construction by Chakraborty that does not involve function composition. Another similar construction appeared in [7].

5. Chakraborty's function [7] For integers k, m such that $2 < k < m$ and $2k \mid m$, the function $g_k\colon \{0,1\}^m \to \{0,1\}$ is defined as follows.
For $x = (x_0, \dots x_{m-1})$, $g_k(x) = 1$ iff $\exists\, i \in \{0, \dots m-1\}$ such that $x_i = x_{i+1(\mod m)} = 1$ and $x_j = 0$ for all $j \in \{i+2(\mod m), \dots, i+k-1(\mod m)\}$.
Then, $s_0(g_k) = \frac{2m}{k}$, $s_1(g_k) = k$, $bs_0(g_k) = \frac{m}{2}$ and $bs_1(g_k) = k$.

Therefore, setting $k = \sqrt{2m}$ gives $s(g_{\sqrt{2m}}) = \sqrt{2m}$ and $bs(g_{\sqrt{2m}}) = \frac{m}{2}$. So we have $bs(g_{\sqrt{2m}}) = \frac{1}{4}s(g_{\sqrt{2m}})^2$.

3 New Building Blocks for Quadratic Separation

Here we define several new functions that we will use as inner or outer functions in various function compositions to obtain quadratic separations.

3.1 A General Framework Based on Certificates

For an odd integer m, consider a set of partial assignments $C = \{c_1, c_2, \dots c_m\}$ on a set of variables $\mathcal{X}$ with $|\mathcal{X}| = 2m$. We say that the set of partial assignments C is good if it satisfies the following 2 properties:
(a) The distance between any two partial assignments $c_i, c_j \in C$ is at least 3.
(b) Each partial assignment c_i has exactly 2 bits set to 1, $\frac{3}{2}(m-1)$ bits set to 0 and the remaining bits free.
Consider the function g_C defined by a good set of partial assignments. We prove the following lemma for such a function g_C:

▶ **Lemma 13.** *For an odd integer m, and any function $g_C\colon \{0,1\}^{2m} \to \{0,1\}$ defined by a good set of partial assignments C, we have:*

(1) $s_0(g_C) = 1$
(2) $s_1(g_C) = \frac{3m+1}{2}$
(3) $bs_0(g_C) \geq m$

Proof. We first note that the set of partial assignments C also forms a set of 1-certificates for g_C such that every 1-input agrees with exactly one partial assignment from C.

1. $s_0(g_C) = 1$. This follows from property (a) of a good set of partial assignments since: for any 0-input x, there is at most one certificate $c_i \in C$ such that x is at a distance 1 from it.
2. $s_1(g_C) = \frac{3m+1}{2}$. This follows from property (a): Consider a 1-input x agreeing with c_i. The bits fixed by c_i form exactly the set of sensitive bits for f on x, since any two certificates in C are at a distance of at least 3 from each other.
3. $bs_0(g_C) \geq m$. Follows from properties (a),(b): Consider the input 0^{2m} which is a 0-input (due to property (b)).
   Recall that any two certificates c_i, c_j must be at a distance at least 3 from each other. But since each certificate only sets exactly 2 bits to 1, this implies that the bits set to 1 by c_i must be disjoint from the bits set to 1 by c_j, for any $c_i, c_j \in C$.
   Therefore, for the 0-input 0^{2m}, the pair of bits set to 1 by a certificate c_i gives a sensitive block for every $i \in [m]$. All these blocks are mutually disjoint and therefore $bs_0(g_C) \geq m$. ◀

▶ **Theorem 14.** *Consider any function $g_C\colon \{0,1\}^{2m} \to \{0,1\}$ defined by a good set of partial assignments C, for an odd integer m. Then the function $f = OR_{\frac{3m+1}{2}} \circ g_C$ has:*

$$bs(f) = \frac{2}{3}s(f)^2 - \frac{1}{3}s(f).$$

We note that the inner function defined by Ambainis and Sun [5] can be shown to fit into this framework.

We will use Lemma 13 to analyze the functions defined in subsection 3.3.

3.2 Using Finite Field Multiplication

In this subsection, we give constructions of families of functions based on Finite Field Multiplication, which achieve quadratic separation between block sensitivity and sensitivity.

Fix an irreducible polynomial p of degree m in $\mathbb{F}_2[z]$ and consider the representation of the elements of $\mathbb{F}_{2^m}$ as univariate polynomials modulo p.
For $a \in \{0,1\}^m$, we interpret $a = (a_0, \ldots a_{m-1})$ as an element of $\mathbb{F}_{2^m}$ under this representation.

▶ **Definition 15** (Function based on Finite Field Multiplication)**.** The function $g_{FF}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ is defined as follows:
$g_{FF}(a,b) = 1$ iff $a \cdot b = c$, where $c \in \mathbb{F}_{2^m}$ is the element represented as $(0, \ldots, 0, 1)$ and multiplication is over the field $\mathbb{F}_{2^m}$.

We prove the following lemma listing the values of sensitivity and block sensitivity for the function g_{FF}:

▶ **Lemma 16.** *For the function $g_{FF}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$, we have:*
- $s_0(g_{FF}) \leq 2$
- $bs_0(g_{FF}) \geq m$
- $s_1(g_{FF}) = 2m$
- $bs_1(g_{FF}) = 2m$

Proof.
- $s_0(g_{FF}) \leq 2$:
  For any non-zero $a \in \mathbb{F}_{2^m}$, there exists a unique $b \in \mathbb{F}_{2^m}$ such that $a \cdot b = (0, \dots, 0, 1)$ i.e. $g_{FF}(a, b) = 1$. Therefore, for any input $(a, b) \in g_{FF}^{-1}(0)$, at most 1 bit j of a may be flipped to get $a^j \cdot b = (0, 0 \dots 1)$ i.e. a has at most 1 sensitive bit. Similarly, at most 1 bit of b may be sensitive.
- $s_1(g_{FF}) = 2m$
  Consider any input $(a, b) \in g_{FF}^{-1}(1)$. Flipping any bit of a or b changes the value of the product $a \cdot b$. Therefore every bit of (a, b) is sensitive, giving $s_1(g_{FF}) = 2m$.
- $bs_0(g_{FF}) \geq m$
  Consider the 0-input $a = (0, \dots 0), b = (0, \dots 0)$.
  For each $j \in \{0, \dots m-1\}$, we can flip the pair of bits (a_j, b_{m-1-j}), so that their product becomes $c = (0, \dots, 0, 1)$. This gives m disjoint sensitive blocks.
- $bs_1(g_{FF}) = 2m$
  This follows since: $2m \geq bs_1(g_{FF}) \geq s_1(g_{FF}) = 2m$ ◀

The following theorem follows from Lemma 16 and the OR-composition Lemma.

▶ **Theorem 17.** *The function $f = OR_m \circ g_{FF}$ has:*

$$bs(f) \geq \frac{1}{4} s(f)^2$$

We now modify the function g_{FF} to improve the constant of separation from $\frac{1}{4}$ to $\frac{1}{2}$.

▶ **Definition 18.** The function $g^*_{FF}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ is defined as follows: $g^*_{FF}(a, b) = 1$ iff the following two conditions hold:
1. $a \cdot b = c$, where $c \in \mathbb{F}_{2^m}$ is the element represented as $(0, \dots, 0, 1)$ and multiplication is over the field $\mathbb{F}_{2^m}$
2. $a_0 \oplus a_1 \dots \oplus a_{m-1} = 1$

▶ **Lemma 19.** *For the function $g^*_{FF}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$, we have:*
- $s_0(g^*_{FF}) = 1$
- $bs_0(g^*_{FF}) \geq m$
- $s_1(g^*_{FF}) = 2m$
- $bs_1(g^*_{FF}) = 2m$

Proof. Note that Conditions 1. and 2. of Definition 18 both have to hold for 1 inputs, and at least one is violated for 0 inputs.
- $s_0(g^*_{FF}) = 1$
  For a 0-input (a, b) which satisfies condition 1., flipping any bit of a or b changes the product $a \cdot b$ and condition 1. is no longer satisfied. Therefore, such a 0-input has no sensitive bit.
  For any 0-input (a, b) which leaves condition 1. unsatisfied, both a and b can have at most one sensitive bit each as observed in the proof of Lemma 16.
  We further note that for any given 0-input (a, b), only one of a or b can have a sensitive bit because condition 2 has to hold for 1-inputs. Therefore, $s_0(g^*_{FF}) = 1$.

- $s_1(g^*_{FF}) = 2m$
  Consider any 1-input (a, b). Flipping any bit of a or b changes the value of the product $a \cdot b$ and condition 1. is no longer satisfied. Therefore every bit of (a, b) is sensitive, giving $s_1(g^*_{FF}) = 2m$.
- $bs_0(g^*_{FF}) \geq m$
  Consider the 0-input $a = (0, \ldots 0), b = (0, \ldots 0)$.
  For each $j \in \{0, \ldots m-1\}$, we can flip the pair of bits (a_j, b_{m-1-j}), so that their product becomes $c = (0, 0 \ldots 1)$ to satisfy the first condition. Since a^j has exactly one 1, the second condition is satisfied as well, and $g^*_{FF}(a^j, b^{m-1-j}) = 1$. This gives m disjoint sensitive blocks and therefore, $bs_0(g^*_{FF}) \geq m$.
- $bs_1(g^*_{FF}) = 2m$
  This follows since: $2m \geq bs_1(g^*_{FF}) \geq s_1(g^*_{FF}) = 2m$. ◀

Using Lemma 19 and the OR-composition Lemma gives the following theorem.

▶ **Theorem 20.** *The function $f = OR_{2m} \circ g^*_{FF}$ has:*

$$bs(f) \geq \frac{1}{2} s(f)^2.$$

▶ Remark. We could replace $c = (0, \ldots, 0, 1)$ in the above definitions by other field elements and still achieve quadratic separations. In fact using any $c \in \mathbb{F}_{2^m}$, we would get $s_0 \leq 2$ and $s_1 = 2m$ for the inner function. However, we need to choose c carefully to guarantee that bs_0 of the inner function is large enough.

3.3 Using Polynomial Multiplication

We now describe another family of functions similar in essence to the one involving finite field multiplication, but easier to analyze.

Here we consider polynomials over the Integers. For $a \in \{0, 1\}^m$, we interpret the bits of $a = (a_0, \ldots a_{m-1})$ as the coefficients of a univariate polynomial p_a that is $p_a(z) = a_0 + a_1 z + \ldots a_{m-1} z^{m-1}$.

▶ **Definition 21** (Function based on Polynomial Multiplication)**.** The function $g_{poly} \colon \{0, 1\}^m \times \{0, 1\}^m \to \{0, 1\}$ is defined as follows:
$g_{poly}(a, b) = 1$ iff $p_a(z) \cdot p_b(z)$ has a non-zero coefficient for z^{m-1} and has coefficient 0 for z^j for all $j < m - 1$.

It is convenient to use the following equivalent definition.

▶ **Definition 22** (Alternative definition)**.** Consider the set of partial assignments $C = \{c_0, c_1 \ldots c_{m-1}\}$ on variables (a, b) where $a = (a_0, \ldots a_{m-1})$ and $b = (b_0, \ldots b_{m-1})$ defined as below:
For every $i \in \{0, \ldots m-1\}$,

$$c_i(a_j) = \begin{cases} 1, & \text{if } \ j = i \\ 0, & \text{if } j < i \\ \star, & \text{if } j > i \end{cases}$$

$$c_i(b_j) = \begin{cases} 1, & \text{if } \ j = m-1-i \\ 0, & \text{if } j < m-1-i \\ \star, & \text{if } j > m-1-i \end{cases}$$

Now the function g_{poly} is the function defined by the set of partial assignments C i.e. g_C.

We now analyze this function for its sensitivity and block sensitivity:

▶ **Lemma 23.** *For $g_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$, we have:*

- $s_0(g_{poly}) = 2$
- $bs_0(g_{poly}) = m$
- $s_1(g_{poly}) = m+1$
- $bs_1(g_{poly}) = m+1$

Proof.

1. $\boldsymbol{s_0(g_{poly}) = 2}$:
   $s_0(g_{poly}) \leq 2$**:** Let (a,b) be any 0-input of g_{poly}. Let $i \in \{0,\dots m-1\}$ be the smallest index such that $a_i = 1$ (if it exists - let $i = m$ if it does not) and j be the smallest index such that $b_j = 1$(if it exists - let $j = m$ otherwise). Then, we have two cases:
   Case 1: $\boldsymbol{i + j > m-1}$. In this case, the only bits which can be flipped to change the value of g_{poly} from 0 to 1 are a_{m-1-i} (unless $i = m$) and b_{m-1-j} (unless $j = m$).
   Case 2: $\boldsymbol{i + j < m-1}$. Now, the only way to flip a bit and possibly change the value of g_{poly} to 1 is by flipping the bits a_i or b_j.
   $s_0(g_{poly}) \geq 2$**:** The following 0-input (a,b) achieves $s_0(g_{poly},(a,b)) = 2$:
   Let $a_{m-1} = 1$, $a_i = 0\ \forall i < (m-1)$.
   Similarly, $b_{m-1} = 1$, $b_i = 0\ \forall i < (m-1)$.
   Notice that (a,b) has 2 sensitive bits: a_0 and b_0.
2. $\boldsymbol{s_1(g_{poly}) = m+1}$:
   First, we observe from the alternative definition of g_{poly} that every 1-input of g_{poly} agrees with a certificate c_i from the set C. Therefore, $C_1(g_{poly}) \leq (m+1)$.
   Therefore, $s_1(g_{poly}) \leq (m+1)$.
   Also, note that every two certificates of C are at a distance of at least 2 from each other. Therefore, for any 1-input (a,b) of g_{poly}, each of the $m+1$ bits where it agrees with $c_i \in C$ is sensitive. So $s_1(g_{poly}) = m+1$.
3. $\boldsymbol{bs_0(g_{poly}) = m}$:
   We first prove $bs_0(g_{poly}) \geq m$. Consider the 0-input with $a_i = b_i = 0\ \forall\ i \in \{0,\dots m-1\}$. We can flip the pair of bits a_i, b_{m-1-i} for $i \in \{0,\dots m-1\}$ so that the function changes value from 0 to 1. Therefore, $bs_0(g_{poly}) \geq m$.
   Next we prove $bs_0(g_{poly}) \leq m$. Since $s_0(g_{poly}) = 2$, any 0-input other than the all-0 input can have only at most 2 blocks of size 1 each and all the other blocks must have size at least 2. Therefore, $bs_0(g_{poly}) \leq 2 + (m-2) = m$.
4. $\boldsymbol{bs_1(g_{poly}) = m+1}$:
   As observed before, $C_1(g_{poly}) \leq (m+1)$. Also, $s_1(g_{poly}) = m+1$.
   Since $s_1(g_{poly}) \leq bs_1(g_{poly}) \leq C_1(g_{poly})$, we have $bs_1(g_{poly}) = m+1$. ◀

Lemma 23 and the OR-composition Lemma imply the following theorem.

▶ **Theorem 24.** *Consider $g_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ for any odd integer m. Then the function $f = OR_{\frac{m+1}{2}} \circ g_{poly}$ has:*

$$bs(f) = \frac{1}{2}s(f)^2 - \frac{1}{2}s(f)$$

We modify the above function to improve the constant of separation from $\frac{1}{2}$ to $\frac{2}{3}$.

▶ **Definition 25.** The function $g^*_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$, is defined as: $g^*_{poly}(a,b) = 1$ iff all the following conditions are met:

1. $p_a(z)\cdot p_b(z)$ has a non-zero coefficient for z^{m-1} and has coefficient 0 for z^j for all $j < m-1$
2. If j is the smallest index such that $a_j = 1$, then
   $a_i = 0$ for all i such that $i > j$ and $i \oplus j = 1$
3. If k is the smallest index such that $b_k = 1$, then
   $b_i = 0$ for all i such that $i > k$ and $i \oplus k = 0$

It is again helpful to consider an equivalent definition based on certificates.

▶ **Definition 26** (Alternative definition)**.** Consider the set of partial assignments $C' = \{c'_0, c'_1, \dots c'_{m-1}\}$ on variables (a,b) where $a = (a_0, \dots a_{m-1})$ and $b = (b_0, \dots b_{m-1})$ defined as below:
For every $i \in \{0, \dots m-1\}$,

$$c'_i(a_j) = \begin{cases} 1, & \text{if } \ j = i \\ 0, & \text{if } j < i \\ 0, & \text{if } j > i \text{ and } i \oplus j = 1 \\ \star, & \text{if } j > i \text{ and } i \oplus j = 0 \end{cases}$$

$$c'_i(b_j) = \begin{cases} 1, & \text{if } \ j = m-1-i \\ 0, & \text{if } j < m-1-i \\ 0, & \text{if } j > m-1-i \text{ and } (m-1-i) \oplus j = 0 \\ \star, & \text{if } j > m-1-i \text{ and } (m-1-i) \oplus j = 1 \end{cases}$$

Now the function g^*_{poly} is the function defined by the set of partial assignments C' i.e. $g_{C'}$.

▶ **Lemma 27.** *Consider $g^*_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ for any odd integer m. Then*

- $s_0(g^*_{poly}) = 1$
- $bs_0(g^*_{poly}) \geq m$
- $s_1(g^*_{poly}) = \frac{3m+1}{2}$
- $bs_1(g^*_{poly}) = \frac{3m+1}{2}$

Proof. It is clear from the alternative definition of g^*_{poly} that it is defined by a set of good assignments.Therefore, we can use Lemma 13 to prove that:

- $s_0(g^*_{poly}) = 1$
- $bs_0(g^*_{poly}) \geq m$
- $s_1(g^*_{poly}) = \frac{3m+1}{2}$

Furthermore, from the alternative definition of g^*_{poly}, every 1-input has a certificate of size at most $\frac{3m+1}{2}$.
Therefore, $bs_1(g^*_{poly}) \leq C_1(g^*_{poly}) \leq \frac{3m+1}{2}$.
Also, $bs_1(g^*_{poly}) \geq s_1(g^*_{poly}) = \frac{3m+1}{2}$.
Therefore $bs_1(g^*_{poly}) = \frac{3m+1}{2}$. ◀

The following theorem follows from Lemma 27 and the OR-composition Lemma.

▶ **Theorem 28.** *Consider $g^*_{poly}\colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ for any odd integer m. Then the function $f = OR_{\frac{3m+1}{2}} \circ g^*_{poly}$ has:*

$$bs(f) \geq \frac{2}{3}s(f)^2 - \frac{1}{3}s(f).$$

Note that this bound matches the current best quadratic separation of [5].

4 Additional Properties of Function Composition

As we noted in Section 2, properties of function composition have been formally studied in [17, 9] in the context of separating sensitivity and block sensitivity. Here we take a closer look at the effect of function composition on the measures 0-sensitivity, 1-sensitivity, 0-block sensitivity and 1-block sensitivity. These properties provide the tools we need to obtain quadratic separation of both 0-block sensitivity and 1-block sensitivity from sensitivity.

First we define measures to quantify the number of sensitive bits for f on x which are equal to 0 and those that are equal to 1 in x.

▶ **Definition 29.** For a function $f\colon \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$, we define:
$\sigma_1(f,x) = |\{i | x_i = 1 \ \text{ AND } \ f(x) \neq f(x^i)\}|$,
$\sigma_0(f,x) = |\{i | x_i = 0 \ \text{ AND } \ f(x) \neq f(x^i)\}|$.

We will use the following notation. We index the bits of the input $y \in \{0,1\}^{mn}$ to $f \circ g$ as $y = (y_{11}, y_{12}, \dots y_{1m}, y_{21}, \dots y_{2m}, \dots y_{n1}, \dots y_{nm})$.
We denote by y_i the i-th group of m bits of y, that is $y_i = (y_{i1}, y_{i2}, \dots y_{im})$.

▶ **Lemma 30.** *For any functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$, we have:*

$$s_0(f \circ g) = \max_{x \in f^{-1}(0)} \{\sigma_0(f,x)s_0(g) + \sigma_1(f,x)s_1(g)\},$$

$$s_1(f \circ g) = \max_{x \in f^{-1}(1)} \{\sigma_0(f,x)s_0(g) + \sigma_1(f,x)s_1(g)\}.$$

Proof. We first prove the first equation. The second equation has an analogous proof.

LHS $\geq$ RHS. Consider the input $a \in f^{-1}(0)$ for which $(\sigma_0(f,a)s_0(g) + \sigma_1(f,a)s_1(g))$ is maximized. Note that $s_0(g), s_1(g)$ don't change for different choices of $a \in f^{-1}(0)$. Now, consider an input $y \in \{0,1\}^{mn}$ such that, $a = (g(y_1), \dots g(y_n))$ and for each $i \in [n]$, if $a_i = g(y_i) = 0$, then $s(g, y_i) = s_0(g)$ and if $a_i = g(y_i) = 1$, then $s(g, y_i) = s_1(g)$. So if $a_i = 0$, we choose as y_i a 0-input of g which achieves the 0-sensitivity of g, and similarly, if $a_i = 1$, we choose as y_i a 1-input of g which achieves the 1-sensitivity of g.
Since $a \in f^{-1}(0)$, y must be a 0-input of $f \circ g$. Therefore, we have

$$s_0(f \circ g) \geq s(f \circ g, y) \geq \sigma_0(f,a)s_0(g) + \sigma_1(f,a)s_1(g).$$

LHS $\leq$ RHS. Consider the input $y \in \{0,1\}^{mn}$ which achieves the 0-sensitivity of $f \circ g$ i.e. $s_0(f \circ g) = s(f \circ g, y)$. Let $g(y_1) = x_1$, $g(y_2) = x_2$ and so on, and let $x = (x_1, x_2 \dots x_n)$. Consider the expression $(\sigma_0(f,x)s_0(g) + \sigma_1(f,x)s_1(g))$. Now, if a bit y_{ij} of y is sensitive for $f \circ g$, then the bit $x_i = g(y_i)$ must be a sensitive bit for f on x.
Now, consider the set $\mathcal{X}_0$ of indices $i \in [n]$ constructed the following way: i is included in $\mathcal{X}_0$ iff $x_i = g(y_i) = 0$ and there is a bit y_{ij} sensitive for $f \circ g$ on y.
Similarly, we define the set $\mathcal{X}_1$ of indices $i \in [n]$ constructed the following way: i is included in $\mathcal{X}_1$ iff $x_i = g(y_i) = 1$ and there is a bit y_{ij} sensitive for $f \circ g$ on y.
Note that for every $i \in \mathcal{X}_0$, the bit x_i is a 0-bit of x and f is sensitive to the i-th bit on x.
So $|\mathcal{X}_0| \leq \sigma_0(f,x)$.
Similarly $|\mathcal{X}_1| \leq \sigma_1(f,x)$.
Now,

$$\begin{aligned} s(f \circ g, y) &= \sum_{i \in \mathcal{X}_0} s(g, y_i) + \sum_{j \in \mathcal{X}_1} s(g, y_j) \\ &\leq \sum_{i \in \mathcal{X}_0} s_0(g) + \sum_{j \in \mathcal{X}_1} s_1(g) \\ &\leq \sigma_0(f,x)s_0(g) + \sigma_1(f,x)s_1(g). \end{aligned}$$

Therefore,

$$s_0(f \circ g) = s(f \circ g, y) \leq \sigma_0(f,x)s_0(g) + \sigma_1(f,x)s_1(g) \leq RHS.$$ ◀

To simplify the equations of Lemma 30 (at the cost of being less precise), we define

- $\sigma_0^0(f) := \max_{x \in f^{-1}(0)} \sigma_0(f,x)$
- $\sigma_0^1(f) := \max_{x \in f^{-1}(1)} \sigma_0(f,x)$
- $\sigma_1^0(f) := \max_{x \in f^{-1}(0)} \sigma_1(f,x)$
- $\sigma_1^1(f) := \max_{x \in f^{-1}(1)} \sigma_1(f,x)$

Finally, we define:

$$\sigma_0(f) := \max\{\sigma_0^0(f), \sigma_0^1(f)\}$$

$$\sigma_1(f) := \max\{\sigma_1^0(f), \sigma_1^1(f)\}$$

We can now use Lemma 30 to get the following bounds:

▶ **Corollary 31.** *For any functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$*

$$s_0(f \circ g) \leq \sigma_0^0(f)s_0(g) + \sigma_1^0(f)s_1(g)$$

$$s_1(f \circ g) \leq \sigma_0^1(f)s_0(g) + \sigma_1^1(f)s_1(g)$$

Note that the equalities in Lemma 30 change to inequalities in Corollary 31, since the max of $\sigma_0(f,x)$ and $\sigma_1(f,x)$ may be achieved on different inputs among $x \in f^{-1}(0)$ (or among $x \in f^{-1}(1)$, respectively).
We now state some simple observations for these measures.

▶ **Lemma 32.** *For any function $f\colon \{0,1\}^n \to \{0,1\}$, and any input x, we have:*
1. $s(f,x) = \sigma_0(f,x) + \sigma_1(f,x)$
2. $\sigma_0^0(f) \leq s_0(f)$
3. $\sigma_1^0(f) \leq s_0(f)$
4. $\sigma_0^0(f) + \sigma_1^0(f) \geq s_0(f)$
5. $\sigma_0^1(f) \leq s_1(f)$
6. $\sigma_1^1(f) \leq s_1(f)$
7. $\sigma_0^1(f) + \sigma_1^1(f) \geq s_1(f)$

The proof of Lemma 32 is straightforward from the definitions.

Now we present an observation about these measures for monotone functions.

▶ **Lemma 33.** *For any monotone function $f\colon \{0,1\}^n \to \{0,1\}$, we have:*
- $\sigma_0^1(f) = 0$
- $\sigma_1^0(f) = 0$

The proof follows from the definition of monotone functions.

We now consider the effects of function composition on 0- block sensitivity and 1-block sensitivity.

▶ **Lemma 34.** *For any functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$, we have:*

$$bs_0(f \circ g) \geq bs_0(f) \cdot \min\{bs_0(g), bs_1(g)\},$$

$$bs_1(f \circ g) \geq bs_1(f) \cdot \min\{bs_0(g), bs_1(g)\}.$$

Proof. Consider $x \in \{0,1\}^n$ such that $f(x) = 0$ and $bs_0(f) = bs(f,x)$.
Now, consider input $y \in \{0,1\}^{mn}$ such that, $x = (g(y_1), \dots g(y_n))$ and for each $i \in [n]$, if $x_i = g(y_i) = 0$, then $bs(g, y_i) = bs_0(g)$ and if $x_i = g(y_i) = 1$, then $bs(g, y_i) = bs_1(g)$. So if $x_i = 0$, we choose as y_i a 0-input of g on which its 0-block sensitivity is achieved, and similarly, if $x_i = 1$, we choose as y_i a 1-input of g on which its 1-block sensitivity is achieved. Now, we claim that $bs_0(f \circ g, y) \geq bs_0(f) \min\{bs_0(g), bs_1(g)\}$. To see this, let $\rho_1, \rho_2, \dots, \rho_k$ be the disjoint sensitive blocks for f on x where $k = bs(f,x)$. For each of these sensitive blocks, there are at least $\min\{bs_0(g), bs_1(g)\}$ disjoint blocks of y such that flipping any of them changes the value of $f \circ g$. This gives at least $bs_0(f) \cdot \min\{bs_0(g), bs_1(g)\}$ disjoint sensitive blocks for $f \circ g$ on the input y, and the first equation follows.
The second equation can be proved in an analogous way. ◀

We get a stronger form of Lemma 34 if f satisfies some additional conditions.

▶ **Lemma 35.** *For any functions $f \colon \{0,1\}^n \to \{0,1\}$ and $g \colon \{0,1\}^m \to \{0,1\}$
if f satisfies (1) $f(0^n) = 0$, (2) $bs_0(f) = bs(f, 0^n)$, then $bs_0(f \circ g) \geq bs_0(f) \cdot bs_0(g)$
and if f satisfies (1) $f(1^n) = 1$, (2) $bs_1(f) = bs(f, 1^n)$, then $bs_1(f \circ g) \geq bs_1(f) \cdot bs_1(g)$*

Proof. Consider input $y \in \{0,1\}^{mn}$ such that, $0^n = (g(y_1), \dots g(y_n))$ and $bs(g, y_i) = bs_0(g)$ for each $i \in [n]$.

Now, we claim that $bs(f \circ g, y) \geq bs_0(f) bs_0(g)$. To see this, let $\rho_1, \rho_2, \dots, \rho_k$ be the disjoint sensitive blocks for f on input 0^n, where $k = bs(f, 0^n)$. For each of these $k = bs(f, 0^n)$ sensitive blocks, there are $bs_0(g)$ disjoint blocks of y that we can flip and change the value of $f \circ g$.
This gives $bs_0(f) \cdot bs_0(g)$ disjoint sensitive blocks for $f \circ g$ on the input y.
The second inequality can be proved analogously. ◀

Comparing the statement of Lemma 35 with Lemma 11 of Tal [17] we note that in the context of 0-block sensitivity and 1-block sensitivity it is enough to require an additional condition for the outer function. On the other hand the condition on the inner function in Lemma 11 of Tal [17] is necessary as illustrated by considering $f = OR_n$ and $g = AND_n$.

Note that the conditions we require are similar to, but slightly different from being in z-good form: It follows from the definition, that if f is in z-good form, then $bs(f) = bs_z(f)$. Our conditions do not require that $bs(f) = bs_z(f)$ for a specific z.

5 Quadratic Separation of both $bs_0(f)$ and $bs_1(f)$ from $s(f)$

We obtain constructions of functions with quadratic separation of both 0-block sensitivity and 1-block sensitivity from sensitivity by considering various compositions of our new building blocks as well as some of the inner functions used in previous quadratic separations.

▶ **Theorem 36.** *Consider $g_{poly} \colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$.
Let $f \colon \{0,1\}^{4m^2} \to \{0,1\}$ be defined as $f = g_{poly} \circ g_{poly}$.
Then, we have:*

- $s_0(f) = 2(m-1)$
- $bs_0(f) \geq m^2$
- $s_1(f) = 4(m-1)$
- $bs_1(f) \geq m(m+1)$

Therefore, we have:

$$\min\{bs_0(f), bs_1(f)\} = \Omega(s(f)^2).$$

Proof. In this proof, we refer to g_{poly} by g, and we use the notation $bs_{min}(f) = \min\{bs_0(f), bs_1(f)\}$.

We first prove the following claims about σ-values for g:

▶ **Claim 37.** *For any input $x \in g^{-1}(0)$ exactly one of the following must be true:*
- $\sigma_0(g, x) = s(g, x)$ *and* $\sigma_1(g, x) = 0$
- $\sigma_1(g, x) = s(g, x)$ *and* $\sigma_0(g, x) = 0$

Proof of Claim. For any 0-input $x = (a, b)$ of g, (1) of Lemma 32 states that:
$\sigma_0(g, x) + \sigma_1(g, x) = s(g, x)$.
As in the definition of g_{poly}, consider the polynomials $p_a(z), p_b(z)$. If the lowest degree monomial of $p_a(z)p_b(z)$ with a non-zero coefficient is z^t then, we have 2 cases:
Case 1: $t < m - 1$. In this case, no 0-bit of a or b can be sensitive. Therefore, $\sigma_0(g, x) = 0$ and $\sigma_1(g, x) = s(g, x)$.
Case 2: $t > m - 1$. In this case, no 1-bit of a or b can be sensitive. Therefore, $\sigma_1(g, x) = 0$ and $\sigma_0(g, x) = s(g, x)$. ◀

▶ **Claim 38.** *For an input $x \in g^{-1}(1)$,*
- $\sigma_0(g, x) = m - 1$
- $\sigma_1(g, x) = 2$

Proof of Claim. Recall the alternative definition based on certificates. Any 1-input x of g belongs to a unique subcube given by a certificate $c_i \in C$. Since the subcubes corresponding to different certificates in C are disjoint and at a distance of at least 2 from each other, every bit of x that is fixed by c_i is sensitive.
Since each certificate fixes exactly 2 bits to 1 and $(m-1)$ bits to 0, we have $\sigma_0(g, x) = (m-1)$ and $\sigma_1(g, x) = 2$. ◀

We can now use Lemma 30 to compute the sensitivity of f:
$s_0(f) = 2s_1(g) = 2(m - 1)$.
$s_1(f) = (m - 1) \cdot 2 + 2 \cdot (m - 1) = 4(m - 1)$
Since $g(0^{2m}) = 0$ and $bs_0(g) = bs(g, 0^{2m})$, we can use Lemma 35 to get:
$bs_0(f) \geq bs_0(g)^2 = m^2$.
We can use Lemma 34 to get:
$bs_1(f) \geq bs_1(g) \cdot \min\{bs_0(g), bs_1(g)\} = m(m + 1)$.
Therefore, we have $bs(f) \geq \frac{s(f)^2}{16}$ and $bs_{min}(f) = \Omega(s(f)^2)$. ◀

We prove the following general theorem:

▶ **Theorem 39.** *For functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^n \to \{0,1\}$ such that the following conditions hold:*
1. $\sigma_1(f) = c_1$*, where* c_1 *is some fixed constant*
2. $s_0(g) = c_2$*, where* c_2 *is some fixed constant*
3. $bs_0(f), bs_1(f), bs_0(g), bs_1(g) = \theta(n)$

We have,

$$\min\{bs_0(f \circ g), bs_1(f \circ g)\} = \Omega(s(f \circ g)^2).$$

The proof is straightforward from Corollary 31 and Lemma 34.

This theorem allows us to use various compositions of our new building blocks and some of the inner functions of previous constructions to obtain other functions with both 0-block sensitivity and 1-block sensitivity quadratically larger than sensitivity.

In particular, let f, g be any two functions from the following list of functions: Rubinstein's inner function [15], Virza's inner function [19], Ambainis and Sun's inner function [5], g_{poly}, g^*_{poly}. In addition, we can also let g be g_{FF}, g^*_{FF}, or the inner function of the function based on Hamming Code [10]. Then, $bs_0(f \circ g)$ and $bs_1(f \circ g)$ are both quadratically larger than $s(f \circ g)$.

References

1 Andris Ambainis. Polynomial degree vs. quantum query complexity. *J. Comput. Syst. Sci.*, 72(2):220–238, 2006. doi:10.1016/j.jcss.2005.06.006.

2 Andris Ambainis, Mohammad Bavarian, Yihan Gao, Jieming Mao, Xiaoming Sun, and Song Zuo. Tighter Relations between Sensitivity and Other Complexity Measures. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 101–113, 2014. doi:10.1007/978-3-662-43948-7_9.

3 Andris Ambainis and Krisjanis Prusis. A Tight Lower Bound on Certificate Complexity in Terms of Block Sensitivity and Sensitivity. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 33–44, 2014. doi:10.1007/978-3-662-44465-8_4.

4 Andris Ambainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Sensitivity Versus Certificate Complexity of Boolean Functions. In *Computer Science – Theory and Applications*, pages 16–28, Cham, 2016. doi:10.1007/978-3-319-34171-2_2.

5 Andris Ambainis and Xiaoming Sun. New separation between *s(f)* and *bs(f)*. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:116, 2011. URL: http://eccc.hpi-web.de/report/2011/116.

6 Harry Buhrman and Ronald De Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

7 Sourav Chakraborty. On the Sensitivity of Cyclically-Invariant Boolean Functions. *CoRR*, abs/cs/0501026, 2005. URL: http://arxiv.org/abs/cs/0501026, arXiv:cs/0501026.

8 Stephen A. Cook, Cynthia Dwork, and Rüdiger Reischuk. Upper and Lower Time Bounds for Parallel Random Access Machines without Simultaneous Writes. *SIAM J. Comput.*, 15(1):87–97, 1986. doi:10.1137/0215006.

9 Justin Gilmer, Michael Saks, and Srikanth Srinivasan. Composition limits and separating examples for some boolean function complexity measures. *Combinatorica*, 36(3):265–311, June 2016. doi:10.1007/s00493-014-3189-x.

10 Parikshit Gopalan, Rocco A. Servedio, Avishay Tal, and Avi Wigderson. Degree and Sensitivity: tails of two distributions. *CoRR*, abs/1604.07432, 2016. arXiv:1604.07432.

11 Pooya Hatami, Raghav Kulkarni, and Denis Pankratov. Variations on the Sensitivity Conjecture. *Theory of Computing, Graduate Surveys*, 4:1–27, 2011. doi:10.4086/toc.gs.2011.004.

12 Claire Kenyon and Samuel Kutin. Sensitivity, Block Sensitivity, and L-block Sensitivity of Boolean Functions. *Inf. Comput.*, 189(1):43–53, February 2004. doi:10.1016/j.ic.2002.12.001.

13 Noam Nisan. CREW PRAMs and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991. doi:10.1137/0220062.

14 Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, December 1994. doi:10.1007/BF01263419.

15 David Rubinstein. Sensitivity vs. block sensitivity of Boolean functions. *Combinatorica*, 15(2):297–299, June 1995. doi:10.1007/BF01200762.

16 Hans-Ulrich Simon. A tight ω(loglog n)-bound on the time for parallel RAM's to compute nondegenerated boolean functions. *Information and Control*, 55(1):102–107, 1982. `doi:10.1016/S0019-9958(82)90477-6`.

17 Avishay Tal. Properties and applications of boolean function composition. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 441–454, 2013. `doi:10.1145/2422436.2422485`.

18 Avishay Tal. Personal communication, 2018.

19 Madars Virza. Sensitivity versus block sensitivity of Boolean functions. *Information Processing Letters*, 111(9):433–435, 2011. `doi:10.1016/j.ipl.2011.02.001`.