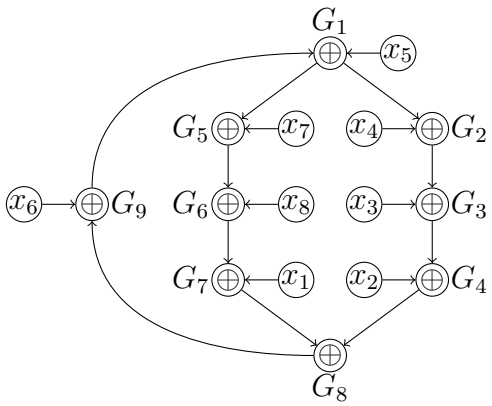


$$
\begin{aligned}
G_1 &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \\
G_2 &= x_1 \oplus x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \\
G_3 &= x_1 \oplus x_2 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \\
G_4 &= x_1 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \\
G_5 &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_8 \\
G_6 &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \\
G_7 &= x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \\
G_8 &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_7 \oplus x_8 \\
G_9 &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_6 \oplus x_7 \oplus x_8
\end{aligned}
$$

$$
\begin{aligned}
G_1 &= G_9 \oplus x_5 \\
G_2 &= G_1 \oplus x_4 \\
G_3 &= G_2 \oplus x_3 \\
G_4 &= G_3 \oplus x_2 \\
G_5 &= G_1 \oplus x_7 \\
G_6 &= G_5 \oplus x_8 \\
G_7 &= G_6 \oplus x_1 \\
G_8 &= G_4 \oplus G_7 \\
G_9 &= G_8 \oplus x_6
\end{aligned}
\qquad
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\times
\begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \\ G_5 \\ G_6 \\ G_7 \\ G_8 \\ G_9 \end{bmatrix}
=
\begin{bmatrix} x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_7 \\ x_8 \\ x_1 \\ 0 \\ x_6 \end{bmatrix}
$$

Figure 2: A simple example of a cyclic xor-circuit. In this case all the gates are labeled with $\oplus$. The affine functions computed by the gates are shown to the right of the circuit. The bottom row shows the program computed by the circuit as well as the corresponding linear system.

Rule 2: If G is trivial, i.e., it computes a constant operation c, remove G and "embed" this constant to the next gates. That is, for every gate H fed by G, replace the operation $h(g,t)$ computed in this gate (where g is the input from G and t is the other input) by the operation $h'(g,t) = h(c,t)$. (Clearly, h' depends on at most one argument, which is not optimal, and in this case after removing G one typically applies Rule 3 or Rule 2 to its successors.)

Rule 3: If G is passing, i.e., it computes an operation depending only on one of its inputs, remove G by reattaching its outgoing wires to that input. This may also require changing the operations computed at its successors (the corresponding input may be negated; note that an and-type gate (xor-type gate) remains an and-type gate (xor-type gate)).

If G feeds itself and depends on another input, then the self-loop wire (which would now go nowhere) is dropped. (Note that if G feeds itself it cannot depend on the self-loop input.)

If G has no outgoing edges it must be an output gate (otherwise it would be removed by Rule 0). In this special case, we remove G and mark the corresponding input of G (or its negation) as the output gate.

Rule 4: If G is a 1-gate that feeds a single gate Q, Q is distinct from G itself, and Q is also fed by one of G's inputs, then replace in Q the incoming wire going from G by a wire going from the other input of G (this might also require changing the operation at Q); then remove G. We call such a gate G *useless*.

Rule 5: If the inputs of G coincide (I_1 and I_2 refer to the same node) then we replace the binary operation $g(x,y)$ computed in G with the operation $g'(x,y) = g(x,x)$. Then perform the same operation on G as described in Rule 3 or 2.

Proposition 3. *Each of the Rules 1–5 removes one internal gate, introduces at most four new troubled gates. None of the rules change the functions of n input variables computed in the gates that are not removed. A fair semicircuit remains a fair semicircuit.*

Proof. Fairness. The circuit remains fair since no rule changes the set of solutions of the system.

New troubled gates. For all the rules, the only gates that may become troubled are I_1, I_2 (if they are and-type gates), and the gates they feed after the transformation (if I_1 or I_2 is a variable). Each of I_1, I_2 may create at most two new troubled gates. Hence each rule, when applied, introduces at most four new troubled gates. □

3.2.3 Affine substitutions

In this subsection, we show how to make substitutions that do create cycles. This will be needed in order to make affine substitutions. Namely, we take a gate computing an affine function $x_1 \oplus \bigoplus_{i \in I} x_i \oplus c$ (where $c \in \{0,1\}$ is a constant) and "rewire" a circuit so that this gate is replaced by a trivial gate computing a constant $b \in \{0,1\}$, while x_1 is replaced by an internal gate. The resulting circuit over $x_2, \ldots, x_n$ may be viewed as the initial circuit under the substitution $x_1 \leftarrow \bigoplus_{i \in I} x_i \oplus c \oplus b$. The "rewiring" is formally explained below; however, before that we need to prove a structural lemma (which is trivial for acyclic circuits) that guarantees its success.

For an xor-circuit, we say that a gate G depends on a variable x if G computes an affine function in which x is a term. Note that in a circuit without cycles this means that precisely one of the inputs of G depends on x, and one could trace this dependency all the way to x, therefore there always exists a path from x to G. In the following lemma we show that it is always possible to find such a path in a fair cyclic circuit too. However, it may be possible that some nodes on this path do not depend on x. Note that dependencies in cyclic circuits are sometimes counterintuitive. For example, in Figure 2, gate G_4 is fed by x_2 but does not depend on it.

Lemma 1. *Let C be a fair cyclic xor-circuit, and let the gate G depend on the variable x. Then there is a path from x to G.*

Proof. Let us substitute all variables in C except for x to 0. Since G depends on x, it can only compute x or its negation.

Let $\mathcal{R}$ be the set of internal gates that are reachable from x, and $\mathcal{U}$ be the set of internal gates that are not reachable from x. Let us enumerate the gates in such a way that gates from $\mathcal{U}$ have smaller indices than gates from $\mathcal{R}$. Then the circuit C corresponds to the system

$$\begin{bmatrix} U & 0 \\ R_1 & R_2 \end{bmatrix} \times \mathcal{G} = \begin{bmatrix} L_U \\ L_R \end{bmatrix},$$

where $\mathcal{G} = (g_1, \ldots, g_{|C|})^T$ is a vector of unknowns (the gates' values), U is the principal submatrix corresponding to $\mathcal{U}$ (a square submatrix whose rows and columns correspond to the gates from $\mathcal{U}$). Note that

- the upper right part of the matrix is 0, because there are no wires going from $\mathcal{R}$ to $\mathcal{U}$, and thus unknowns corresponding to gates from $\mathcal{R}$ do not appear in the equations corresponding to gates from $\mathcal{U}$,
- L_U is a vector of constants, it cannot contain x since $\mathcal{U}$ is not reachable from x,
- L_R is a vector of affine functions of x, since all other inputs are substituted by zeros.

If U is singular, then the whole matrix is singular, which contradicts the fairness of C. Therefore, U is nonsingular, i.e., the values $\mathcal{G}' = (g_1, \ldots, g_{|\mathcal{U}|})^T$ are uniquely determined by

$U \times \mathcal{G}' = L_U$, and they are constant (independent of x). This means that G cannot belong to $\mathcal{U}$.

□

We now come to rewiring.

Lemma 2. *Let C be a fair semicircuit with input gates $x_1, \ldots, x_n$ and internal gates $G_1, \ldots, G_m$. Let G be a gate not reachable by a directed path from any and-type gate. Assume that G computes the function $x_1 \oplus \bigoplus_{i \in I} x_i \oplus c$, where $I \subseteq \{2, \ldots, n\}$. Let $b \in \{0, 1\}$ be a constant. Then one can transform C into a new circuit C' with the following properties:*

1. *graph-theoretically, C' has the same gates as C, plus a new internal gate Z; some edges are changed, in particular, x_1 is disconnected from the circuit;*
2. *the operation in G is replaced by the constant operation b;*
3. $\mathrm{in}_{C'}(Z) = 2$, $\mathrm{out}_{C'}(G) = \mathrm{out}_C(G) + 1$, $\mathrm{out}_{C'}(x_1) = 0$. $\mathrm{out}_{C'}(Z) = \mathrm{out}_C(x_1) - 1$.
4. *The indegrees and outdegrees of all other gates are the same in C and C'.*
5. *C' is fair.*
6. *all gates common for C' and C compute the same functions on the affine subspace defined by $x_1 \oplus \bigoplus_{i \in I} x_i \oplus c \oplus b = 0$, that is, if $f(x_1, \ldots, x_n)$ is the function computed by an internal gate in C and $f'(x_2, \ldots, x_n)$ is the function computed by its counterpart in C', then $f(\bigoplus_{i \in I} x_i \oplus c \oplus b, x_2, \ldots, x_n) = f'(x_2, \ldots, x_n)$. The gate Z computes the function $\bigoplus_{i \in I} x_i \oplus c \oplus b$ (which on the affine subspace equals x_1).*

Proof. Consider a path from x_1 to G that is guaranteed to exist by Lemma 1. Denote the internal gates on this path by $G_1, \ldots, G_k = G$. Denote by $T_1, \ldots, T_k$ the other inputs of these gates. Note that we assume that $G_1, \ldots, G_k$ are pairwise different gates while some of the gates $T_1, \ldots, T_k$ may coincide with each other and with some of $G_1, \ldots, G_k$ (it might even be the case that $T_i = G_i$).

The transformation is as shown in Figure 3. The gates $A_0, \ldots, A_k$ are shown on the picture just for convenience: any of $x_1, Z, G_1, \ldots, G_k$ may feed any number of gates, not just one A_i.

To show the fairness of C', assume the contrary, that is, the sum of a subset of rows of the new matrix is zero. The row corresponding to $G_k = b$ must belong to the sum (otherwise we would have only rows of the matrix for C, plus an extra column). However, this would mean that if we sum up the corresponding lines of the system (not just the matrix) for C, we get $G_k = \mathrm{const} \oplus \bigoplus_{j \in J} x_j$ where $J \not\ni 1$ (note that x_1 was replaced by Z in the new system, and cancelled out by our assumption). This contradicts the assumption of the Lemma that G_k computes the function $x_1 \oplus \bigoplus_{i \in I} x_i \oplus c$. Therefore, the matrix for C' has full rank.

The programs shown next to the circuits explain that for $x_1 = \bigoplus_{i \in I} x_i \oplus c \oplus b$, the gates $G_1, \ldots, G_k$ compute the same values in C' and C; the value of Z is also clearly correct. □

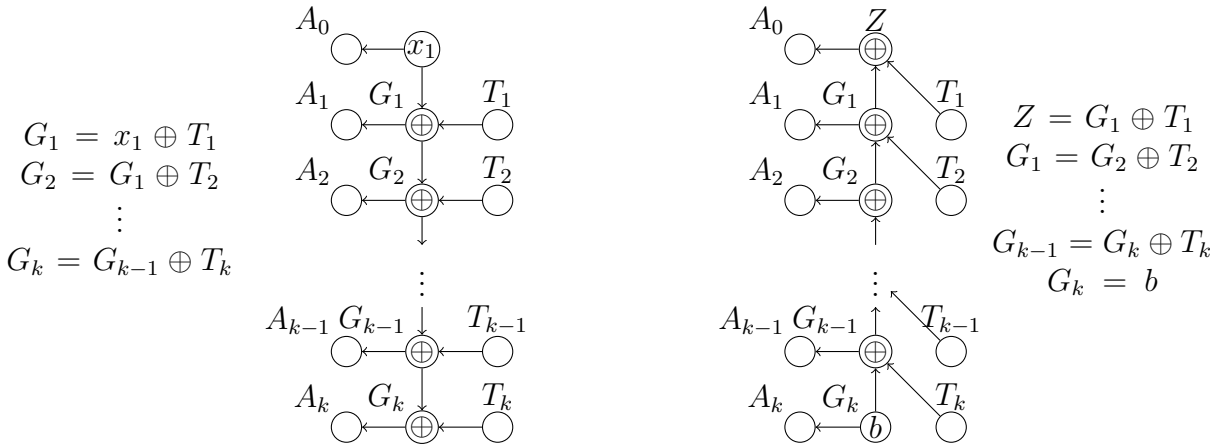


Figure 3: This figure illustrates the transformation from Lemma 2. We use $\oplus$ as a generic label for xor-type gates. That is, in the picture, gates labelled $\oplus$ may compute the function $\equiv$.

Corollary 1. *This transformation does not introduce new troubled gates.*

Proof. Indeed, the gates being fed by $G_1, \ldots, G_{k-1}, G_k, Z$ are not fed by variables; these gates themselves are not and-type gates; other gates do not change their degrees or types of input gates. □

After we apply the transformation, we apply Rule 2 to G. Since the only troubled gates introduced by this rule are the inputs of the removed gate, no troubled gates are introduced (and one gate, G itself, is eliminated, thus the combination of Lemma 2 and Rule 2 does not increase the number of gates).

3.3 Read-once depth-2 quadratic sources

We generalize affine sources as follows.

Definition 1. *Let the set of variables $\{x_1, \ldots, x_n\}$ be partitioned into three disjoint sets $F, L, Q \subseteq \{1, \ldots, n\}$ (for* free*,* linear*, and* quadratic*). Consider a system of equalities that contains*

- *for each variable x_j with $j \in Q$, a quadratic equality of the form*

$$x_j = (x_i \oplus c_i)(x_k \oplus c_k) \oplus c_j \,,$$

where $i, k \in F$ and c_i, c_k, c_j are constants; the variables from the right-hand side of all the quadratic substitutions are pairwise disjoint;

Definition 2. *Let $R \subseteq \mathbb{F}_2^n$ be an rdq-source of dimension d, let the free variables be $x_1, x_2, \ldots, x_d$, and let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ be a function. Then f restricted to R, denoted $f|_R$, is a function $f|_R\colon \mathbb{F}_2^d \to \mathbb{F}_2$, defined by $f|_R(x_1, \ldots, x_d) = f(R(x_1, \ldots, x_d))$.*

Note that affine sources are precisely rdq-sources with $Q = \emptyset$. We define dispersers for rdq-sources similarly to dispersers for affine sources.

Definition 3. *An* rdq-disperser *for dimension $d(n)$ is a family of functions $f_n\colon \mathbb{F}_2^n \to \mathbb{F}_2$ such that for all sufficiently large n, for every rdq-source R of dimension at least $d(n)$, $f_n|_R$ is non-constant.*

The following proposition shows that affine dispersers are also rdq-dispersers for related parameters.

Proposition 4. *Let R be an rdq-source of $\mathbb{F}_2^n$ of dimension d. Then R contains an affine subspace of dimension at least $d/2$.*

Proof. For each quadratic substitution $x_j = (x_i \oplus c_i)(x_k \oplus c_k) \oplus c_j$, further restrict R by setting $x_i = 0$. This replaces a quadratic substitution by two affine substitutions $x_i = 0$ and $x_j = c_i(x_k \oplus c_k) \oplus c_j$; the number of free variables is decreased by one. Also, since the free variables do not occur on the left-hand side, the newly introduced affine substitution is consistent with the previous affine substitutions.

Since the variables occurring on the right-hand side of our quadratic substitutions are disjoint we have initially that $2|Q| \leq |F| = d$, so the number of newly introduced affine substitutions is at most $d/2$. □

Note that it is important in the proof that protected variables do not appear on the left-hand sides. The proposition above is obviously false for quadratic *varieties*: no Boolean function can be non-constant on all sets of common roots of $n - o(n)$ quadratic polynomials. For example, the system of $n/2$ quadratic equations $x_1x_2 = x_3x_4 = \ldots = x_{n-1}x_n = 1$ defines a single point, so any function is constant on this set.

Corollary 2. *An affine disperser for sublinear dimension is also an rdq-disperser for sublinear dimension.*

3.4 Circuit complexity measure

For a circuit C and a straight-line program R defining an rdq-source, we define the following circuit complexity measure:

$$\mu(C, R) = g + \alpha_Q \cdot q + \alpha_T \cdot t + \alpha_I \cdot i\,,$$

where g is the number of internal gates in C, q is the number of quadratic substitutions in R, t is the number of troubled gates in C, and i is the number of *influential* input gates in C. We say that an input is influential if it feeds at least one gate or is protected (recall that a

variable is protected if it occurs in the right-hand side of a quadratic substitution in R). The constants $\alpha_Q, \alpha_T, \alpha_I > 0$ will be chosen later.

Proposition 3 implies that when a gate is removed from a circuit by applying a normalization rule the measure μ is reduced by at least $\beta = 1 - 4\alpha_T$. The constant α_T will be chosen to be very close to 0 (certainly less than 1/4), so $\beta > 0$.

In order to estimate the initial value of our measure, we need the following lemma.

Lemma 3. *Let C be a circuit computing an affine disperser $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ for dimension d, then the number of troubled gates in C is less than $\frac{n}{2} + \frac{5d}{2}$.*

Proof. Let V be the set of the inputs, $|V| = n$. In what follows we let $\sqcup$ denote the disjoint set union. Let us call two inputs x and y neighbors if they feed the same troubled gate. Assume to the contrary that $t \geq \frac{n}{2} + \frac{5d}{2}$. Let v_i be the number of variables feeding exactly i troubled gates. Since a variable feeding a troubled gate must have outdegree 2, $v_i = 0$ for $i > 2$. By double counting the number of wires from inputs to troubled gates, $2t = v_1 + 2v_2$. Since $v_1 + v_2 \leq n$,

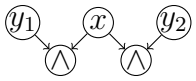
$$n + 5d \leq 2t = v_1 + 2v_2 \leq n + v_2.$$

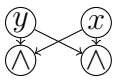
Let T be the set of inputs that feed two troubled gates, $|T| = v_2 \geq 5d$. We now construct two disjoint subsets $X \subset T$ and $Y \subset V$ such that

- $|X| = d$,
- there are $|Y|$ consistent linear equations that make the circuit C independent of variables from $X \sqcup Y$.

When the sets X and Y are constructed the theorem statement follows immediately. Indeed, we first take $|Y|$ equations that make C independent of $X \sqcup Y$, then we set all the remaining variables $V \setminus (X \sqcup Y)$ to arbitrary constants. After this, the circuit C evaluates a constant (since it does not depend on variables from $X \sqcup Y$ and all other variables are set to constants). We have $|Y| + |V \setminus (X \sqcup Y)| = |V \setminus X| = n - d$ linear equations which contradicts the assumption that f is an affine disperser for dimension d.

Now we turn to constructing X and Y. For this we will repeat the following routine d times. First we pick any variable $x \in T$, it feeds two troubled gates, let y_1 and y_2 be neighbors of x (y_1 may coincide with y_2). We add x to X, also we add y_1, y_2 to Y. Note that it is possible to assign constants to y_1 and y_2 to make C independent of x. (See the figure below. If y_1 differs from y_2, then we substitute constants to them so that they eliminate troubled gates fed by x and leave C independent of x. If y_1 coincides with y_2, then either $x = c$, or $y_1 = c$, or $y_1 = x \oplus c$ eliminates both troubled gates for some constant c; if we make an $x = c$ substitution, then formally we have to interchange x and y, that is, add y rather than x to X.) Each of y_1, y_2 has at most one neighbor different from x. We remove x, y_1, y_2, neighbors of y_1 and y_2 (at most five vertices total) from the set T, if they belong to it. Since at each step we remove at most five vertices from T, we can repeat this routine d times. Since we remove the neighbors of y_1 and y_2 from T, we guarantee that in all future steps when we pick an input, its neighbors do not belong to Y, so we can make arbitrary substitutions to them and leave the system consistent.





□

We are now ready to formulate our main result.

Theorem 1. *Let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ be an rdq-disperser for dimension d and C be a fair semicircuit computing f. Let $\alpha_Q, \alpha_T, \alpha_I \geq 0$ be some constants, and $\alpha_T \leq 1/4$. Then $\mu(C, \emptyset) \geq \delta(n-d-2)$ where*

$$\delta := \alpha_I + \min\left\{\frac{\alpha_I}{2}, 4\beta, 3+\alpha_T, 2\beta+\alpha_Q, 5\beta-\alpha_Q, 2.5\beta+\frac{\alpha_Q}{2}\right\}, \tag{1}$$

and $\beta = 1 - 4\alpha_T$.

We defer the proof of this theorem to the next section. This theorem, together with Corollary 2, implies a lower bound on the circuit complexity of affine dispersers.

Corollary 3. *Let $\delta, \beta, \alpha_Q, \alpha_T, \alpha_I$ be constants as above, then the circuit size of an affine disperser for sublinear dimension is at least*

$$\left(\delta - \frac{\alpha_T}{2} - \alpha_I\right) n - o(n)\,.$$

Proof. Note that $q = 0$, $i \leq n$, $t < \frac{n}{2} + \frac{5d}{2}$ (see Lemma 3). Thus, the circuit size is

$$\begin{aligned} g &= \mu - \alpha_Q \cdot q - \alpha_T \cdot t - \alpha_I \cdot i > \delta(n-d-2) - \alpha_T \cdot \left(\frac{n}{2} + \frac{5d}{2}\right) - \alpha_I \cdot n \\ &= \left(\delta - \frac{\alpha_T}{2} - \alpha_I\right) n - \left(\delta + \frac{5\alpha_T}{2}\right) d - 2\delta = \left(\delta - \frac{\alpha_T}{2} - \alpha_I\right) n - o(n)\,. \end{aligned}$$

□

The maximal value of $\delta - \frac{\alpha_T}{2} - \alpha_I$ satisfying the condition from Corollary 3 is given by the following linear program: maximize $\delta - \frac{\alpha_T}{2} - \alpha_I$ subject to

$$\begin{aligned} \beta + 4\alpha_T &= 1 \\ \alpha_T, \alpha_Q, \alpha_I, \beta &\geq 0 \\ \delta &\leq \alpha_I + \min\left\{\frac{\alpha_I}{2}, 4\beta, 3+\alpha_T, 2\beta+\alpha_Q, 5\beta-\alpha_Q, 2.5\beta+\frac{\alpha_Q}{2}\right\}. \end{aligned}$$

The optimal values for this linear program are

$$\begin{aligned}
\alpha_T &= \frac{1}{43}, \\
\alpha_Q &= 1 + 22\alpha_T = \frac{65}{43}, \\
\alpha_I &= 6 + 2\alpha_T = 6 + \frac{2}{43}, \\
\beta &= 1 - 4\alpha_T = \frac{39}{43}, \\
\delta &= 9 + 3\alpha_T = 9 + \frac{3}{43}.
\end{aligned}$$

This gives a $(3 + \frac{1}{86})n - o(n)$ lower bound for an affine disperser for sublinear dimension.

3.5 Gate elimination

In order to prove Theorem 1 we first show that it is always possible to make a substitution and decrease the measure by δ.

Theorem 2. *Let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ be an rdq-disperser for dimension d, let R be an rdq-source of dimension $s \geq d + 2$, and let C be an optimal (i.e., C with the smallest $\mu(C, R)$) fair semicircuit computing the function $f|_R$. Then there exist an rdq-source R' of dimension $s' < s$ and a fair semicircuit C' computing the function $f|_R$ such that*

$$\mu(C', R') \leq \mu(C, R) - \delta(s - s')\,.$$

Before we proceed to the proof, we show how to infer the main theorem from this claim:

Proof of Theorem 1. We prove that for optimal C computing $f|_R$, $\mu(C, R) \geq \delta(s - d - 2)$. We do it by induction on s, the dimension of R. Note that the statement is vacuously true for $s \leq d + 2$, since μ is nonnegative. Now suppose the statement is true for all rdq-sources of dimension strictly less than s for some $s > d + 2$, and let R be an rdq-source of dimension s. Let C be a fair semicircuit computing $f|_R$. Let R' be the rdq-source of dimension s' guaranteed to exist by Theorem 2, and let C' be a fair semicircuit computing $f|_R$. We have that

$$\mu(C, R) \geq \mu(C', R') + \delta(s - s') \geq \delta(s - d - 2),$$

where the second inequality comes from the induction hypothesis. □

3.5.1 Proof sketch

The proof of Theorem 2 is based on a careful consideration of a number of cases. Before considering all of them formally, we show a high-level picture of the case analysis.

We fix the values of constants $\alpha_T, \alpha_Q, \alpha_I, \beta, \delta$ to the optimal values: $\alpha_T = \frac{1}{43}, \alpha_Q = \frac{65}{43}, \alpha_I = 6\frac{2}{43}, \beta = \frac{39}{43}, \delta = 9\frac{3}{43}$. Now it suffices to show that we can always make one

It is conceivable that when we count several eliminated gates, some of them coincide, so that we actually eliminate fewer gates. Usually in such cases we can prove that some other gates become trivial. This and other degenerate cases are handled in the full proof in the next subsection.

3.5.2 Full proof

Proof of Theorem 2. Since normalization does not increase the measure and does not change R, we may assume that C is normalized.

In what follows we will further restrict R by decreasing the number of free variables either by one or by two, then we will implement these substitutions in C and normalize C afterwards. Formally, we do it as follows:

- We add an equation or two to R.
- Since we now compute the disperser on a smaller set, we simplify C (in particular, we disconnect the substituted variables from the rest of the circuit). For this, we
  - change the operations in the gates fed by the substituted variables or restructure the xor part of the circuit according to Lemma 2,
  - apply some normalization rules to remove some gates (and disconnect substituted variables).
- We count the decrease of μ.
- We further normalize the circuit (without increase of μ) to bring it to the normalized state required for the next induction step.

Since $s \geq d + 2$, even if we add two more lines to R, the disperser will not become a constant. This, in particular, implies that if a gate becomes constant then it is not an output gate and hence feeds at least one other gate. By going through the possible cases we will show that it is always possible to perform one or two consecutive substitutions matching at least one of the following types (by $\Delta\mu$ we denote the decrease of the measure after subsequent normalization).

1. Perform two consecutive affine substitutions to reduce the number of influential inputs by at least three. Per one substitution, this gives $\Delta\mu \geq 1.5\alpha_I$.
2. Perform one affine substitution to reduce the number of influential inputs by at least 2: $\Delta\mu \geq 2\alpha_I$ (numerically, this case is subsumed by the previous one).
3. Perform one affine substitution to kill four internal gates: $\Delta\mu \geq 4\beta + \alpha_I$.
4. Perform one constant substitution to eliminate three internal gates including at least one troubled gate so that no new troubled gate is introduced: $\Delta\mu \geq \alpha_I + 3 + \alpha_T$.

5. Perform one *quadratic* substitution to kill five internal gates: $\Delta\mu \geq 5\beta - \alpha_Q + \alpha_I$.

6. Perform two affine substitutions to kill at least five internal gates and replace a quadratic substitution by an affine one, reducing the measure by at least $5\beta + \alpha_Q + 2\alpha_I$. Per substitution this is $\Delta\mu \geq 2.5\beta + \frac{\alpha_Q}{2} + \alpha_I$.

7. Perform one affine substitution to kill two internal gates and replace one quadratic substitution by an affine one: $\Delta\mu \geq 2\beta + \alpha_Q + \alpha_I$.

All substitutions that we perform are of the form such that adding them to an rdq-source results in a new rdq-source.

We check all possible cases of (C, R). In every case we assume that the conditions of the previous cases are not satisfied. We also rely on the specified order of applications of the normalization rules where applicable.

Note that the measure can accidentally drop less than we expect if new troubled gates emerge. We take care of this when counting the number of internal gates that disappear, recall Proposition 3 that guarantees the decrease of β per one eliminated gate. If some additional gate accidentally disappears, it may introduces new troubled gates but does not increase the measure, because $\beta \geq 0$.

Cases:

1. The circuit contains a protected variable q that either feeds an and-type gate or feeds at least two internal gates. Then there is a type 7 substitution of q by a constant.

2. The circuit contains a protected 0-variable q occurring in the right-hand side of a quadratic substitution together with some variable q'. We substitute a constant to q'. After this neither q nor q' are influential, so we have a type 2 substitution.

   Note that after this case all protected variables are 1-variables feeding xor gates.

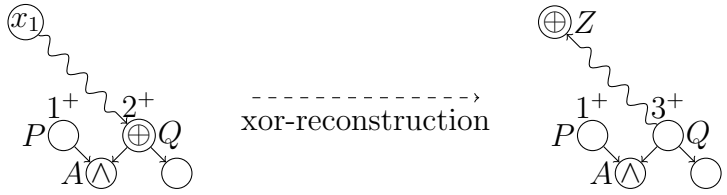
3. The circuit contains a variable x feeding an and-type gate T, and $out(x) + out(T) \geq 4$. Then if x gets the value that trivializes T, we remove four gates: T by Rule 2, and descendants of x and T by Rule 3. If some of these descendants coincide, this gate becomes trivial (instead of passing) and is removed by Rule 2 (instead of Rule 3), and an additional gate (a descendant of this descendant) is removed by Rule 3. This makes a type 3 substitution.

   Note that after this case all variables feeding and-gates have outdegree one or two.

4. There is an and-type gate T fed by two input gates x and y, one of which (say, x) has outdegree 1. Adopt the notation from the following picture. In this and all the subsequent pictures we show the outdegrees near the gates that are important for the case analysis.

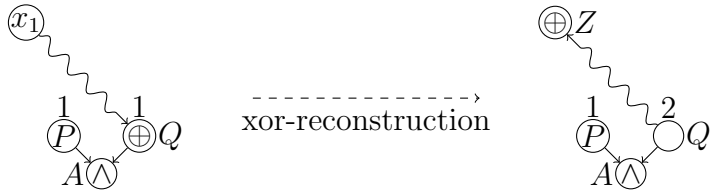
Below we go through several subcases depending on the type of the gate P.

i. Q is a 2^+-gate. We recall the general picture of xor-reconstruction.



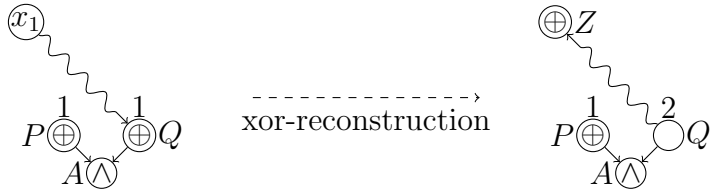
After the reconstruction, there are at least three descendants of Q and at least one descendant of A, a type 3 substitution.

ii. Q is an internal 1-gate and P is an input gate. Then P has outdegree 1 and is unprotected (see Cases 10, 1).



Note that $P = x_1$ since the only outgoing edge of P goes to an and-type gate. This means that P is left untouched by the xor-reconstruction. After trivializing A the circuit becomes independent of both x_1 and P giving a type 2 substitution.

iii. Q is an internal 1-gate and P is an internal gate. Then P is a 1-gate (if the outdegree of P were larger we would switch the roles of P and Q).



Again, P is left untouched by the xor-reconstruction since it only has one successor and it is of and-type while the xor-reconstruction is performed in the linear part of the circuit. After the substitution, we remove two successors of Q, at least one successor of A, and make P a 0-gate. A type 3 substitution. Note that P cannot be a successor of Q because of Rule 4.

(b) All variables in the affine function computed by Q are protected.

i. Both inputs to Q, say x_j and x_k, are variables, and they occur in the same quadratic substitution $w = (x_j \oplus c)(x_k \oplus c') \oplus c''$. Then perform a substitution $x_j = x_k \oplus c'''$ (using Proposition 2) in order to trivialize the gate A. It kills the quadratic substitution (and does not harm other quadratic substitutions, because x_j and x_k could not occur in them), Q, A, its descendant (and more, but we do not need it), which makes $\Delta\mu \geq 3\beta + \alpha_Q + \alpha_I$, a type 7 substitution.

which (call it x_j) have a couple x_k that does not feed P. We substitute x_k by a constant and normalize the descendant of x_k. It only kills one xor-gate fed by x_k and makes x_j unprotected. Note that at this point P is still a 1-xor. We then trivialize A by substituting x_j by an affine function. Similarly to Case 11(a)iii, this kills four gates and gives, for two substitutions, $\Delta\mu \geq 5\beta + \alpha_Q + 2\alpha_I$. A type 6 substitution.

C. The only case when the condition of the previous case does not apply is the following: P computes an affine function on a single variable x_i, Q computes an affine function on a single variable x_j, the variables x_i and x_j appear together in a quadratic substitution, and moreover x_i feeds Q while x_j feeds P. But this is just impossible. Indeed, since x_i is a protected variable it only feeds Q. As Q computes an affine function on x_i, Lemma 1 guarantees that there is a path from x_i to Q. But this path must go through P and A leading to a cycle that goes through an and-type gate A.

□

Acknowledgements

Research is partially supported by NSF (grant 1319051) and the Government of the Russian Federation (grant 14.Z50.31.0030). We also would like to thank Dmitry Itsykson and Alexander Knop who survived a six-hour seminar on the proof and made valueable comments.

References

[And87] Alexander E. Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes. *Moscow Univ. Math. Bull.*, 42(1):63–66, 1987.

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *CCC-98*, 1998.

[BK12] Eli Ben-Sasson and Swastik Kopparty. Affine dispersers from subspace polynomials. *SIAM J. Comput.*, 41(4):880–914, 2012.

[BKS+10] Boaz Barak, Guy Kindler, Ronen Shaltiel, Benny Sudakov, and Avi Wigderson. Simulating independence: New constructions of condensers, Ramsey graphs, dispersers, and extractors. *J. ACM*, 57(4), 2010.

[Blu84] Norbert Blum. A Boolean function requiring $3n$ network size. *Theor. Comput. Sci.*, 28:337–345, 1984.