Multiparty Karchmer–Wigderson Games and Threshold Circuits

Alexander Kozachinskiy* Vladimir Podolskii†

Received July 16, 2020; Revised May 8, 2022; Published June 12, 2022

Abstract. We propose a generalization of the Karchmer–Wigderson communication games to the multiparty setting. Our generalization turns out to be tightly connected to circuits consisting of threshold gates. This allows us to obtain new explicit constructions of such circuits for several functions. In particular, we provide an explicit (polynomial-time computable) log-depth monotone formula for the Majority function, consisting only of 3-bit majority gates and variables. This resolves a conjecture of Cohen et al. (CRYPTO’13).

1 Introduction

Karchmer and Wigderson established a tight connection between circuit depth and communication complexity [10] (see also [12, Chapter 9]). They showed that for every Boolean function


*The work of A. Kozachinskiy was performed at the Steklov International Mathematical Center and supported by the Ministry of Science and Higher Education of the Russian Federation (agreement no. 075-15-2022-265).

†Supported by the HSE University Basic Research Program and by the Ministry of Science and Higher Education of the Russian Federation (agreement no. 075-15-2022-265).

ACM Classification: F.1.3, F.1.2

AMS Classification: 68Q17, 68Q25

Key words and phrases: Karchmer–Wigderson games, threshold circuits, threshold gates, majority function

© 2022 Alexander Kozachinskiy and Vladimir Podolskii
Licensed under a Creative Commons Attribution License (CC-BY) DOI: 10.4086/toc.2022.v018a015
one can define a communication game whose communication complexity exactly equals the depth of \( f \) in the standard De Morgan basis. This discovery turned out to be very influential in Complexity Theory. A lot of circuit depth lower bounds as well as formula size lower bounds rely on this discovery \([9, 13, 4, 6, 3]\). Karchmer–Wigderson games have been used also in adjacent areas like Proof Complexity (see, e.g., \([14]\)).

Karchmer–Wigderson games represent a deep connection of two-party communication protocols with De Morgan circuits. Loosely speaking, one party is responsible for \( \land \)-gates and the other party is responsible for \( \lor \)-gates. In this paper, we address the question of what would be a natural generalization of Karchmer–Wigderson games to the multiparty setting. Is it possible to obtain in this way a connection with other types of circuits?

We answer positively to this question: we suggest such a generalization and show its connection to circuits consisting of threshold gates. To motivate our results we first present applications we get from this new connection.

### 1.1 Applications to circuits

It is well-known that the Majority function, \( \text{MAJ}_{2n+1} \), can be computed by a monotone circuit of depth \( O(\log n) \). This was first proved by Ajtai, Komlós and Szemerédi \([1]\). More specifically, they constructed a polynomial-time computable sorting network of depth \( O(\log n) \), now known as the AKS sorting network. In turn, any sorting network can be easily converted into a monotone circuit of the same depth, computing the Majority function. Namely, we first replace every comparator of the network by a pair of an \( \land \)-gate and an \( \lor \)-gate, and then notice that the median output of the network coincides with the Majority function.

The construction of Ajtai, Komlós and Szemerédi is fairly complicated, and has a large constant before the \( \log n \). Valiant \([15]\) gave a much simpler proof of the existence of a monotone formula of depth \( O(\log n) \) for the Majority function. He used the probabilistic method, so his argument does not give an explicit construction of such a formula.

Several authors (see, e.g., \([5, 2]\)) noticed that Valiant’s proof actually gives a formula of depth \( O(\log n) \) for \( \text{MAJ}_{2n+1} \), consisting only of \( \text{MAJ}_3 \)-gates (and, importantly, with no constants). Once again, this formula is not explicit. On the other hand, the AKS sorting network gives an explicit formula of depth \( O(\log n) \) for \( \text{MAJ}_{2n+1} \) which consists of \( \land \)-gates and \( \lor \)-gates. There is no obvious way to convert it into a formula which consists of \( \text{MAJ}_3 \)-gates and does not use constants\(^2\). For brevity, we will call these formulas \( \text{MAJ}_3 \)-formulas. So there is a natural question—is it possible to construct a \( \text{MAJ}_3 \)-formula of depth \( O(\log n) \) for \( \text{MAJ}_{2n+1} \), deterministically in polynomial time?

This question was stated as a conjecture by Cohen et al. in \([2]\). First, they showed that the answer is positive under some cryptographic assumptions. Second, they constructed (unconditionally) a polynomial-time computable \( \text{MAJ}_3 \)-formula of depth \( O(\log n) \) which coincides with \( \text{MAJ}_n \) for all inputs in which the fraction of ones is bounded away from \( 1/2 \) by \( 2^{-\Theta(\sqrt{\log n})} \).

---

\(^1\)For technical convenience, in this paper we always assume that the Majority function has an odd number of inputs.

\(^2\)If we had constants, we could easily express the disjunction and the conjunction by a \( \text{MAJ}_3 \) gate: \( \text{MAJ}_3(x, y, 0) = x \land y \), \( \text{MAJ}_3(x, y, 1) = x \lor y \).
We show that the conjecture of Cohen et al. is true (unconditionally).

**Theorem 1.1.** There exists a polynomial-time computable $\text{MAJ}_3$-formula of depth $O(\log n)$ for $\text{MAJ}_{2n+1}$.

We use the AKS sorting network in the proof. In fact, one can use any polynomial-time computable construction of a monotone circuit of depth $O(\log n)$ for $\text{MAJ}_{2n+1}$. We also obtain the following general result:

**Theorem 1.2.** If there is a monotone formula (i.e., a formula consisting of $\land$-gates and $\lor$-gates of fan-in 2) for $\text{MAJ}_{2n+1}$ of size $s$, then there is a $\text{MAJ}_3$-formula for $\text{MAJ}_{2n+1}$ of size $O(s \cdot n^{\log_2(3)}) = O(s \cdot n^{1.58..})$.

The transformation from the last theorem, however, is not efficient. We can make this transformation polynomial-time computable, provided $\log_2(3)$ is replaced by $1/(1 – \log_3(2)) \approx 2.71$. In turn, we view Theorem 1.2 as a potential approach to obtain super-quadratic lower bounds on the formula size of $\text{MAJ}_{2n+1}$ in the $\{\text{MAJ}_3\}$ basis. Arguably, this basis may be easier to analyze than the standard monotone basis. The best known size upper bounds in the $\{\land, \lor\}$ basis and the $\{\text{MAJ}_3\}$ basis are, respectively, $O(n^{5.5})$ and $O(n^{4.20})$ [7]. Both bounds are due to Valiant’s method (see [7] also for the limitations of Valiant’s method).

We also study a generalization of the conjecture of Cohen et al. to threshold functions. By $\text{THR}^b_n$ we denote the following Boolean function:

$$\text{THR}^b_n : \{0, 1\}^b \rightarrow \{0, 1\}, \quad \text{THR}^b_n(x) = \begin{cases} 1 & x \text{ contains at least } a \text{ ones,} \\ 0 & \text{otherwise.} \end{cases}$$

For some reasons (to be discussed below) a natural generalization would be a question of whether $\text{THR}^{k+1}_{n+1}$ can be computed by formula of depth $O(\log n)$ which does not use constants and consists only of $\text{THR}^{k+1}_2$-gates. We will call such formulas $Q_k$-formulas (note that $Q_2$-formulas are $\text{MAJ}_3$-formulas, because $\text{THR}_2^3 = \text{MAJ}_3$). This question was also addressed by Cohen et al. in [2]. First, they observed that there is a construction of depth $O(n)$ (and exponential size). Second, they gave an explicit construction of depth $O(\log n)$, which coincides with $\text{THR}^{k+1}_{n+1}$ for all inputs in which the fraction of ones is bounded away from $1/k$ by $O(1/\sqrt{\log n})$.

However, no exact (even non-explicit) construction with sublinear depth or subexponential size was known. In particular, Valiant’s probabilistic construction does not work for $k \geq 3$. In this paper, we improve depth $O(n)$ to depth $O(\log^2 n)$ and size $\exp(O(n))$ to size $n^{O(1)}$ for this problem.

**Theorem 1.3.** For any constant $k \geq 3$ there exists a polynomial-time computable $Q_k$-circuit of depth $O(\log^2 n)$ for $\text{THR}^{k+1}_{n+1}$ (that is, this circuit does not use constants and consists only of $\text{THR}^{k+1}_2$-gates).

### 1.2 Applications to Multiparty Secure Computations

The conjecture stated in [2] was motivated by applications to Secure Multiparty Computations. The paper [2] establishes an approach to construct efficient multiparty protocols based on...
protocols for a small number of players. More specifically, in their framework one starts with a protocol for a small number of players and a formula $F$ computing a certain boolean function. Then one combines a protocol for a small number of players with itself recursively, where the recursion mimics the formula $F$.

It is shown in [2] that from our result it follows that for any $n$ there is an explicit polynomial size protocol for $n$ players secure against a passive adversary that controls any $t < \frac{n}{2}$ players. It is also implicit in [2] that from Theorem 1.3 for $k = 3$ it follows that for any $n$ there is a protocol of size $2^{O(\log^2 n)}$ for $n$ players secure against an active adversary that controls any $t < \frac{n}{2}$ players. An improvement of the depth of the formula in Theorem 1.3 to $O(\log n)$ would result in a polynomial size protocol [2].

1.3 Multiparty Karchmer–Wigderson games

We now introduce our main conceptual contribution – multiparty Karchmer–Wigderson games.

Let us start with an example. Consider the ordinary monotone Karchmer–Wigderson game for $\text{MAJ}_{2n+1}$. In this game, Alice receives a string $x \in \text{MAJ}_{2n+1}^{-1}(0)$ and Bob receives a string $y \in \text{MAJ}_{2n+1}^{-1}(1)$. In other words, the number of ones in $x$ is at most $n$ and the number of ones in $y$ is at least $n + 1$. The goal of Alice and Bob is to find some coordinate $i$ such that $x_i = 0$ and $y_i = 1$. Next, imagine that Bob flips each of his input bits. After that, each party has a vector with at most $n$ ones. Now Alice and Bob have to find a coordinate in which both vectors are 0.

There is a natural generalization of this problem to the multiparty setting. Assume that there are $k$ parties, and each receives a Boolean vector of length $kn + 1$ with at most $n$ ones. Let the goal of parties be to find a coordinate in which all $k$ input vectors are 0. How many bits of communication are needed for that?

For $k = 2$ the answer is $O(\log n)$, because there exists a monotone formula of depth $O(\log n)$ for $\text{MAJ}_{2n+1}$, and hence the monotone Karchmer–Wigderson game for $\text{MAJ}_{2n+1}$ can be solved in $O(\log n)$ bits of communication. For $k \geq 3$ we are only aware of a simple $O(\log^2 n)$-bit solution based on the binary search.

Note that in this problem, each party receives a vector on which $\text{THR}_{n+1}^k$ equals 0. The goal is to find a common zero. We can consider a similar problem for any function $f$ satisfying a so-called $Q_k$-property: any $k$ vectors from $f^{-1}(0)$ have a common zero. In the next definition we define the $Q_k$-property formally and also introduce the related $R_k$-property.

**Definition 1.4.** Let $Q_k$ be the set of all Boolean functions $f$ satisfying the following property: for all $x^1, x^2, \ldots, x^k \in f^{-1}(0)$ there is a coordinate $i$ such that $x^1_i = x^2_i = \ldots = x^k_i = 0$.

Further, let $R_k$ be the set of all Boolean functions $f$ satisfying the following property: for all $x^1, x^2, \ldots, x^k \in f^{-1}(0)$ there is a coordinate $i$ such that $x^1_i = x^2_i = \ldots = x^k_i$.

For $f \in Q_k$ let the $Q_k$-communication game for $f$ be the following communication problem. There are $k$ parties, the $j$th party receives a Boolean vector $x^j \in f^{-1}(0)$. The goal of parties is to find any coordinate $i$ such that $x^1_i = x^2_i = \ldots = x^k_i = 0$.

Similarly, we can define $R_k$-communication games for functions from $R_k$. In the $R_k$-communication games, the objective of parties is slightly different: their goal is to find any coordinate $i$ and a bit $b$ such that $x^1_i = x^2_i = \ldots = x^k_i = b$. 
Note that $R_2$ contains all self-dual functions – that is, functions that take opposite values in the opposite vertices of a Boolean cube. Similarly, monotone self-dual functions belong to $Q_2$. It is easy to see that $R_2$-communication games are equivalent to Karchmer–Wigderson games for self-dual functions (one party should flip all the input bits). Moreover, $Q_2$-communication games are equivalent to monotone Karchmer–Widgerson games for monotone self-dual functions.

In this paper, we consider $R_k$-communication games as a multiparty generalization of Karchmer–Wigderson games. In turn, $Q_k$-communication games are considered as a generalization of monotone Karchmer–Widgerson games. To justify this choice, one should relate them to some type of circuits.

1.4 Connection to threshold gates and the main result

Every function from $Q_k$ can be lower bounded by a $Q_k$-circuit (that is, by a circuit that does not use constants and consists only of $\text{THR}^{k+1}_2$-gates). More precisely, let us write $C \leq f$ for a Boolean circuit $C$ and a Boolean function $f$ if for all $x \in f^{-1}(0)$ we have $C(x) = 0$. Then the following proposition holds:

**Proposition 1.5** ([2]). The set $Q_k$ is equal to the set of all Boolean functions $f$ for which there exists a $Q_k$-circuit $C \leq f$.

There is a similar characterization of the set $R_k$ via so-called $R_k$-circuits. These are circuits that does not use constants and consist of $\text{THR}^{k+1}_2$-gates and negations that can only be applied to input variables.

**Proposition 1.6.** The set $R_k$ is equal to the set of all Boolean functions $f$ for which there exists an $R_k$-circuit $C \leq f$.

The proof from [2] of Proposition 1.5 with obvious modifications also works for Proposition 1.6.

Given $f \in Q_k$, what is the minimal depth of a $Q_k$-circuit $C \leq f$? We show that this quantity is equal (up to a constant factor) to the communication complexity of the $Q_k$-communication game for $f$.

**Theorem 1.7.** Let $k \geq 2$ be any constant. Then for any $f \in Q_k$ the following two quantities are equal up to a constant factor:

- the communication complexity of the $Q_k$-communication game for $f$;
- the minimal $d$ for which there exists a depth-$d$ $Q_k$-circuit $C \leq f$.

Similar result can be obtained for $R_k$-communication games.

**Theorem 1.8.** Let $k \geq 2$ be any constant. Then for any $f \in R_k$ the following two quantities are equal up to a constant factor:

- the communication complexity of the $R_k$-communication game for $f$;
• the minimal \( d \) for which there exists a depth-\( d \) \( R_k \)-circuit \( C \leq f \).

The proof of each theorem is divided into two parts:

(a) transformation of a depth-\( d \) \( Q_k \)-circuit (and \( R_k \)-circuit) \( C \leq f \) into an \( O(d) \)-bit protocol computing the \( Q_k \)-communication game (\( R_k \)-communication game, resp.) for \( f \);

(b) transformation of a \( d \)-bit protocol computing the \( Q_k \)-communication game (or \( R_k \)-communication game) for \( f \) into a \( Q_k \)-circuit \( C \leq f \) (an \( R_k \)-circuit \( C \leq f \), resp.) of depth \( O(d) \).

The first part is simple and the main challenge is the second part. In Section 6 we also formulate refined versions of Theorems 1.7 and 1.8. We refine these theorems in the following two directions. First, we take into account circuit size and for this we consider dag-like communication protocols. Second, we show that transformations (a-b) can be done in polynomial time (under some mild assumptions).

We derive our upper bounds on the depth of \( \text{MAJ}_{2n+1} \) and \( \text{THR}_{n+1} \) (Theorems 1.1 and 1.3) from Theorem 1.7. We first solve the corresponding \( Q_k \)-communication games with small number of bits of communication. Namely, for the case of \( \text{MAJ}_{2n+1} \) we use the AKS sorting network to solve the corresponding \( Q_2 \)-communication game with \( O(\log n) \) bits of communication. For the case of \( \text{THR}_{n+1} \) with \( k \geq 3 \) we solve the corresponding \( Q_k \)-communication game by a simple binary search protocol with \( O(\log^2 n) \) bits of communication. This is where we get depth \( O(\log n) \) for Theorem 1.1 and depth \( O(\log^2 n) \) for Theorem 1.3. Again, some special measures should be taken to make the resulting circuits polynomial-time computable and to control their size\(^3\).

1.5 Our techniques: hypotheses games

As we already mentioned, the hard part of our main result is to transform a protocol into a circuit.

For this, we develop a new language to describe threshold circuits. For every \( f \) in \( Q_k \) and in \( R_k \) we introduce the corresponding \( Q_k \)-hypotheses game and \( R_k \)-hypotheses game, resp., for \( f \). We show that strategies in these games are equivalent to \( Q_k \)-circuits and \( R_k \)-circuits. It turns out that strategies are more convenient than circuits to simulate protocols, since strategies and protocols operate in the same top-bottom manner.

Once we establish the equivalence of circuits and hypotheses games, it remains for us to transform a communication protocol into a strategy in a hypotheses game. This is an elaborate construction presented in Propositions 5.3 and 5.8.

Here is how we define these games. Fix \( f : \{0,1\}^n \to \{0,1\} \). There are two players, Nature and Learner. Before the game starts, Nature privately chooses \( z \in f^{-1}(0) \) which then cannot be changed. The goal of Learner is to find some \( i \in [n] \) such that \( z_i = 0 \). The game proceeds in

\(^3\)We should only care about the size in case of Theorem 1.3, because depth \( O(\log n) \) immediately gives polynomial size.
rounds. At each round, Learner specifies \( k + 1 \) families \( \mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_k \subseteq f^{-1}(0) \) to Nature. We understand this as if Learner makes the following \( k + 1 \) hypotheses about \( z \):

\[
\text{“} z \in \mathcal{H}_0 \text{”,} \\
\text{“} z \in \mathcal{H}_1 \text{”,} \\
\vdots \\
\text{“} z \in \mathcal{H}_k \text{”}.
\]

Learner loses immediately if less than \( k \) hypotheses are true, i.e., if the number of \( j \in \{0, 1, \ldots, k\} \) satisfying \( z \in \mathcal{H}_j \) is less than \( k \). Otherwise, Nature points out to some hypothesis which is true. In other words, Nature specifies to Learner some \( j \in \{0, 1, \ldots, k\} \) such that \( z \in \mathcal{H}_j \). The game then proceeds in the same manner for some finite number of rounds. At the end, Learner outputs an integer \( i \in [n] \). We say that Learner wins if \( i = 0 \).

It is not hard to show that Learner has a winning strategy in the \( Q_k \)-hypotheses game for \( f \) if and only if \( f \in Q_k \). It is instructive to give a proof of the “if” part of this claim: if \( f \in Q_k \), then Learner has a winning strategy in the \( Q_k \)-hypotheses game for \( f \). We will denote by \( Z \) the set of all \( z \)’s that are compatible with Nature’s answers so far. At the beginning, \( Z = f^{-1}(0) \). If \( |Z| \geq k + 1 \), Learner takes any distinct \( z^1, z^2, \ldots, z^{k+1} \in Z \) and makes the following hypotheses:

\[
\text{“} z \neq z^1 \text{”,} \\
\text{“} z \neq z^2 \text{”,} \\
\vdots \\
\text{“} z \neq z^{k+1} \text{”}.
\]

At least \( k \) hypotheses are true, and any Nature’s response strictly reduces the size of \( Z \). When the size of \( Z \) becomes equal to \( k \), Learner is ready to give an answer due to the \( Q_k \)-property of \( f \).

This strategy requires exponential in \( n \) number of rounds. This can be easily improved to \( O(n) \) rounds. Indeed, instead of choosing \( k + 1 \) distinct elements of \( Z \) split \( Z \) into \( k + 1 \) disjoint almost equal parts. Then let the \( i \)th hypotheses be “\( z \) is not in the \( i \)th part”. Any Nature’s response reduces the size of \( Z \) by a constant factor, until the size of \( Z \) is \( k \).

For \( f \in Q_k \) we can now ask what is the minimal number of rounds in a Learner’s winning strategy. The following proposition gives an exact answer:

**Proposition 1.9.** For any \( f \in Q_k \) the following holds. Learner has a \( d \)-round winning strategy in the \( Q_k \)-hypotheses game for \( f \) if and only if there exists a depth-\( d \) \( Q_k \)-circuit \( C \leq f \).

**Proposition 1.9** is the core result for our applications. For instance, we prove **Theorem 1.1** by giving an explicit \( O(\log n) \)-round winning strategy of Learner in the \( Q_2 \)-hypotheses game for \( \text{MAJ}_{2n+1} \). Let us now sketch our construction of this strategy argument (a complete proof can be found in Section 4).
Assume that the Nature’s input vector is \( z \in \text{MAJ}^{-1}_{2n+1}(0) \). We start by finding two integers \( i, j \in [2n + 1] \) such that either \( z_i = 0 \) or \( z_j = 0 \). This can be achieved in \( O(\log n) \) rounds. Namely, we maintain a set \( S \subseteq [2n + 1] \) with a property that \( z \) equals 0 on some coordinate from \( S \). Initially, \( S = [2n + 1] \). Until the size of \( S \) is 2, we split \( S \) into 3 almost equal parts \( S_1, S_2, S_3 \). We then make the following 3 hypotheses: “\( z \) is 0 on some coordinate from \( S_1 \cup S_2 \)”, “\( z \) is 0 on some coordinate from \( S_1 \cup S_3 \)”, “\( z \) is 0 on some coordinate from \( S_2 \cup S_3 \)”. At least 2 hypotheses are true, and any Nature’s response decreases the size of \( S \) by a factor of 3/2.

Consider a moment when the size of \( S \) became equal to 2, and assume that \( S = \{i, j\} \). Learner knows that either \( z_i = 0 \) or \( z_j = 0 \). It is not hard, at the cost of one more round, to exclude an option that \( z_i = z_j = 0 \). So we may assume from now on, that either \( z_i = 0, z_j = 1 \) or \( z_i = 1, z_j = 0 \). At the final stage of our construction, we take any polynomial-time computable monotone formula \( F \) of depth \( O(\log n) \) for \( \text{MAJ}_{2n+1} \) (for instance, one that can be obtained from the AKS sorting network). We start to descend from the output gate of \( F \) to one of \( F \)'s inputs. Throughout this process, we maintain the following invariant: if \( g \) is the current gate, then either \( g(z) = 0 \land z_i = 0 \) or \( g(\neg z) = 1 \land z_j = 0 \) (here \( \neg \) denotes the bitwise negation). Now, assume w.l.o.g. that \( g \) is an \( \land \)-gate, and let \( g = g_1 \land g_2 \). Note that among the following 3 statements:

\[
\begin{align*}
  \text{“}\ g_1(z) &= 0 \land z_i = 0 \land z_j = 1", \\
  \text{“}\ g_1(z) &= 1 \land g_2(z) = 0 \land z_i = 0 \land z_j = 1", \\
  \text{“}\ g_1(\neg z) &= g_2(\neg z) = 1 \land z_i = 1 \land z_j = 0”,
\end{align*}
\]

one is true and two are false. We make 3 hypothesis, each calling one of these 3 statements false. Nature responses by indicating a false statement. If it indicates the third statement, then we already know that \( z_i = 0 \). Otherwise, we can either descend to \( g_1 \) or to \( g_2 \). Overall, the last stage of our construction costs at most depth(\( F \)) = \( O(\log n) \) rounds. If we reach an input to \( F \), we output the index of the corresponding variable.

In \( R_k \)-hypotheses games, Nature and Learner play in the same way except that now Learner’s objective is to find some pair \( (i, b) \in [n] \times \{0, 1\} \) such that \( z_i = b \). The following analog of Proposition 1.9 holds:

**Proposition 1.10.** For any \( f \in R_k \) the following holds. Learner has a \( d \)-round winning strategy in the \( R_k \)-hypotheses game for \( f \) if and only if there exists a depth-\( d \) \( R_k \)-circuit \( C \leq f \).

### 1.6 Organization of the paper

In Section 2 we give Preliminaries. In Section 3 we formally define \( Q_k \)-hypotheses games and \( R_k \)-hypotheses games, and show their equivalence to \( Q_k \)-circuits and \( R_k \)-circuits, resp. In Section 4 we establish our results for the Majority function—Theorems 1.1 and 1.2. Then in Section 5 we obtain Theorems 1.7 and 1.8—that is, we show that \( Q_k \)-communication games are equivalent to \( Q_k \)-circuits and \( Q_k \)-hypotheses games, and analogously for \( R_k \)-communication games.
Figure 1: Our equivalence results. An arrow from, say, “$Q_k$-hypotheses games” to “$Q_k$-communications games” references a transformation of strategies in $Q_k$-hypotheses games into protocols for $Q_k$-communication games.

More detailed references concerning these equivalences can be found on Figure 1.6.

In Section 5 we also establish a weak version of Theorem 1.3. Namely, we show that there exists a $Q_k$-circuit of depth $O(\log^2 n)$ for $\text{THR}^{k+1}_n$. Unfortunately, results of Section 5 are not sufficient to make this circuit polynomial-time computable.

To overcome this, in Section 6 we refine Theorems 1.7 and 1.8 in order to take into account the circuit size and polynomial-time computability (see Theorems 6.1 and 6.5 below). Then in Section 7 we derive Theorem 1.3 from results of Section 6. Additionally, in Section 7 we provide another proof for Theorem 1.1, via results of Section 6. We deduce from our argument a direct elementary proof of Theorem 1.1 in Section 8. Finally, in Section 9 we formulate some open problems.

2 Preliminaries

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$. For a set $W$ we denote the set of all subsets of $W$ by $2^W$. For two sets $A$ and $B$ by $B^A$ we mean the set of all (total) functions from $A$ to $B$.

We usually use subscripts to denote coordinates of vectors. In turn, we usually use superscripts to numerate vectors.

We use a standard terminology for Boolean functions, formulas and circuits [8]. By $\text{MAJ}_n$ we denote the majority function on $n$ inputs. By $\text{THR}^m_k$ we denote the function $\text{THR}^m_k : \{0, 1\}^m \rightarrow \{0, 1\}^k$ which outputs 1 if and only if the number of 1’s in the input is at least $k$.

We denote the size of a circuit $C$ by $\text{size}(C)$ and the depth by $\text{depth}(C)$. By De Morgan formulas/circuits we mean formulas/circuits consisting of $\wedge$-gates and $\vee$-gates of fan-in 2 and also negations that can only be applied to input variables. By monotone formulas/circuits we mean formulas/circuits consisting of $\wedge$-gates and $\vee$-gates of fan-in 2.

We also consider the following classes of circuits/formulas. By $Q_k$-circuits and $Q_k$-formulas we mean circuits and formulas, resp., that consist of $\text{THR}^{k+1}_2$ gates. We also call $Q_2$-circuits and $Q_2$-formulas $\text{MAJ}_3$-circuits and $\text{MAJ}_3$-formulas. Similarly, by $R_k$-circuits and $R_k$-formulas we
mean circuits and formulas, resp., that consist of THR\(^{(k+1)}\) gates and also negations that can only be applied to input variables. We stress that it is not allowed to use constants in \(Q_1\)-circuits and \(R_k\)-circuits. For all classes of circuits and formulas considered in this paper, we assume that negations do not contribute to the depth.

We use the notion of deterministic communication protocols in the multiparty \emph{number-in-hand} model. Additionally, to capture the circuit size in our results, we consider not only standard \emph{tree-like} protocols, but also \emph{dag-like} protocols. This notion was considered by Sokolov in [14]. We use a slightly different variant of this notion. We provide all necessary definitions in the next subsection.

### 2.1 Dags and dag-like communication protocols

We consider directed acyclic graphs (dags) with possibly more than one directed edge from one node to another. A terminal node of a dag \(G\) is a node with no out-going edges. Given a dag \(G\), let

- \(V(G)\) denote the set of nodes of \(G\);
- \(T(G)\) denote the set of terminal nodes of \(G\).

For \(v \in V(G)\) let \(Out_G(v)\) be the set of all edges of \(G\) that start at \(v\). A dag \(G\) is called \(t\)-ary if for every non-terminal node \(v\) of \(G\) we have \(|Out_G(v)| = t\). An ordered \(t\)-ary dag is a \(t\)-ary dag \(G\) equipped with a mapping from the set of edges of \(G\) to \(\{0, 1, \ldots, t - 1\}\). This mapping, restricted to \(Out_G(v)\), must be injective for every \(v \in V(G) \setminus T(G)\). The value of this mapping on an edge \(e\) will be called the \emph{label} of \(e\). In other words, any \(t\) edges that start at the same node must have different labels.

By a path in \(G\) we mean a sequence of edges \(\langle e_1, e_2, \ldots, e_m \rangle\) such that for every \(j \in [m - 1]\) edge \(e_j\) ends in the same node in which \(e_{j+1}\) starts. Note that there may be two distinct paths visiting the same nodes in the same order, because we allow parallel edges.

We say that a node \(w\) is a descendant of a node \(v\) if there is a path from \(v\) to \(w\). We call \(w\) a successor of \(v\) if there is an edge from \(v\) to \(w\). A node \(s\) is called the \emph{starting node} if every other node is a descendant of \(s\). Note that any dag has at most one starting node (otherwise there will be a cycle in this dag).

If a dag \(G\) has the starting node \(s\), then by the depth of \(v \in V(G)\) we mean the maximal length of a path from \(s\) to \(v\). The depth of \(G\) then is the maximal depth of its nodes.

Let \(X_1, X_2, \ldots, X_k, Y\) be finite sets.

**Definition 2.1.** A \(k\)-party dag-like communication protocol \(\pi\) with inputs from \(X_1 \times X_2 \times \ldots X_k\) and with outputs from \(Y\) is a tuple \(\langle G, P_1, P_2, \ldots, P_k, \phi_1, \phi_2, \ldots, \phi_k, I \rangle\), where

- \(G\) is an ordered 2-ary dag with the starting node \(s\);
- \(P_1, P_2, \ldots, P_k\) is a partition of \(V(G) \setminus T(G)\) into \(k\) disjoint subsets;
- \(\phi_i\) is a function from \(P_i \times X_i\) to \(\{0, 1\}\);
We say that when parties reach a terminal $\mathcal{G}$ parties take one labeled by $\phi_i(v, x)$, where $x \in X_i$ is the input of the $i$th party. Namely, the $i$th party communicates a bit to all the other parties. Namely, the $i$th party communicates the bit $b = \phi_i(v, x)$, where $x \in X_i$ is the input of the $i$th party. Among the two edges starting at $v$, parties take one labeled by $b$ and descend to one of the successors of $v$ along this edge. Finally, when parties reach a terminal $t$, they output $l(t)$.

We say that $x \in X_i$ is $i$-compatible with an edge $e$ from $v$ to $w$ if one of the following two conditions hold:

- $v \notin P_i$;
- $v \in P_i$ and $e$ is labeled by $\phi_i(v, x)$.

We say that $x \in X_4$ is $i$-compatible with a path $p = \langle e_1, e_2, \ldots, e_m \rangle$ of $\mathcal{G}$ if for every $j \in [m]$ it holds that $x$ is $i$-compatible with $e_j$. Intuitively, this means that the $i$th party, having input $x$, communicates along $p$ (from those nodes of $p$ where this party is the one to communicate). Finally, we say that $x \in X_i$ is $i$-compatible with a node $v \in V(\mathcal{G})$ if there is a path $p$ from $s$ to $v$ such that $x$ is $i$-compatible with $p$.

We say that an input $(x^1, x^2, \ldots, x^k) \in X_1 \times X_2 \times \ldots \times X_k$ visits a node $v \in V(\mathcal{G})$ if there is a path $p$ from $s$ to $v$ such that for every $i \in [k]$ it holds that $x^i$ is $i$-compatible with $p$. Note that there is a unique $t \in T(\mathcal{G})$ such that $(x^1, x^2, \ldots, x^k)$ visits $t$.

To formulate an effective version of Theorem 1.7 and Theorem 1.8, we need the following definition.

**Definition 2.2.** The light form of a $k$-party dag-like communication protocol

$$\pi = \langle \mathcal{G}, P_1, P_2, \ldots, P_k, \phi_1, \phi_2, \ldots, \phi_k, l \rangle$$

is the tuple $\langle \mathcal{G}, P_1, P_2, \ldots, P_k, l \rangle$.

In other words, to obtain the light form of $\pi$ we just forget about $\phi_1, \phi_2, \ldots, \phi_k$. So the light form only contains the underlying ordered dag of $\pi$, the partition of non-terminal nodes between parties and the labels of terminals. On the other hand, in the light form there is no information at all how parties communicate at non-terminal nodes.

A protocol $\pi$ computes a relation $S \subseteq X_1 \times X_2 \times \ldots \times X_k \times \mathcal{Y}$ if the following holds. For every $(x^1, x^2, \ldots, x^k) \in X_1 \times X_2 \times \ldots \times X_k$ there exist $y \in \mathcal{Y}$ and $t \in T(\mathcal{G})$ such that $(x^1, \ldots, x^k)$ visits $t$, $l(t) = y$ and $(x^1, x^2, \ldots, x^k, y) \in S$.

Using the language of relations, we can formally define $Q_k$-communication games and $R_k$-communication games. Given $f : \{0, 1\}^n \to \{0, 1\}$, $f \in Q_k$, we define the $Q_k$-communication
game for $f$ as the following relation:

$$S \subseteq \prod_{i=1}^{k} f^{-1}(0) \times \{0, 1\}.$$  

Similarly, given $f : \{0, 1\}^n \to \{0, 1\}, f \in R_k$, we define the $R_k$-communication game for $f$ as the following relation:

$$S \subseteq \prod_{i=1}^{k} f^{-1}(0) \times \{0, 1\}.$$  

It is easy to see that a dag-like protocol for $S$ can be transformed into a tree-like protocol (i.e., into a protocol whose underlying dag is a tree) of the same depth, but this transformation can drastically increase the size.

### 3 Formal treatment of $Q_k$-hypotheses and $R_k$-hypotheses games

Fix $f \in Q_k, f : \{0, 1\}^n \to \{0, 1\}$. Here we define Learner’s strategies in the $Q_k$-hypotheses game for $f$ formally. We consider not only tree-like strategies, but also dag-like. To specify a Learner’s strategy $S$ in the $Q_k$-hypotheses game for $f$, we have to specify:

- An ordered $(k + 1)$-ary dag $G$ with the starting node $s$;
- a subset $\mathcal{H}_j(p) \subseteq \{0, 1\}^n$ for every $j \in \{0, 1, \ldots, k\}$ and for every path $p$ in $G$ from $s$ to some node in $V(G) \setminus T(G)$;
- a number $i_t \in \{0\}^n$ for every terminal $t$ of $G$.

The underlying mechanics of the game is as follows. Let the Nature’s vector be $z \in f^{-1}(0)$. Learner and Nature descend from $s$ to one of the terminals of $G$. More precisely, a position in the game is determined by a path $p$, starting at $s$. If the endpoint of $p$ is not a terminal, Learner specifies sets $\mathcal{H}_0(p), \mathcal{H}_1(p), \ldots, \mathcal{H}_k(p)$ as its hypotheses. If less than $k$ of these sets contain $z$, Nature wins. Otherwise, Nature specifies some $j \in \{0, 1, \ldots, k\}$ such that $z \in \mathcal{H}_j(p)$. Among the $k + 1$ edges that start at the endpoint of $p$, the players take one which is labeled by $j$. After that, they extend $p$ by this edge. At some point, parties reach a terminal $t$ (i.e., the endpoint of $p$ becomes equal $t$). Then the game ends and Learner outputs $i_t$.

We stress that Learner’s output depends only on $t$ but not on a path to $t$ (unlike Learner’s hypotheses). This property will be crucial in establishing a connection between $Q_k$-hypotheses games and $Q_k$-circuits.

We now proceed to a formal definition of what it means that $S$ is winning for Learner.
We will formulate a stronger version of Proposition 1.9. For that we need a notion of the light form. To every path \( p \) starting at \( s \), if the following holds: Then every \( z \in f^{-1}(0) \) is compatible with \( p \). Otherwise, for every \( i \in \{1, \ldots, m\} \), it should hold that \( z \in \mathcal{H}_j((e_1, \ldots, e_{i-1})) \), where \( j \) is the label of \( e_i \). In other words, the label of \( e_i \) should be a possible Nature’s response to Learner’s hypotheses in the position \( (e_1, \ldots, e_{i-1}) \). Informally, this means that Nature, having input \( z \), can reach a position in the game which corresponds to a path \( p \).

We say that a strategy \( S \) is winning for Learner in the \( Q_k \)-hypotheses game for \( f \) if for every path \( p \), starting at \( s \), and for every \( z \in f^{-1}(0) \), compatible with \( p \), the following holds:

- if the endpoint of \( p \) is not a terminal, then the number of \( j \in \{0, 1, \ldots, k\} \) such that \( z \in \mathcal{H}_j(p) \) is at least \( k \);
- if the endpoint of \( p \) is \( t \in T(G) \), then \( z_{i_t} = 0 \).

We will formulate a stronger version of Proposition 1.9. For that we need a notion of the light form of a strategy \( S \). Namely, the light form of \( S \) is its ordered dag \( G \) equipped with a mapping which to every \( t \in T(G) \) assigns \( i_t \). In other words, the light form contains a “skeleton” of \( S \) and Learner’s outputs in terminals (and no information about Learner’s hypotheses).

We can identify the light form of any strategy \( S \) with a \( Q_k \)-circuit. Namely, put a \( \text{THR}_2^{k+1} \)-gate to every non-terminal node \( v \) of \( G \), and for every \( t \in T(G) \) put a variable \( x_{i_t} \) into \( t \). Set \( s \), the starting node of \( G \), to be the output gate.

**Proposition 3.1.** For all Boolean functions \( f \in Q_k \) the following holds:

(a) if \( S \) is a Learner’s winning strategy in the \( Q_k \)-hypotheses game for \( f \), then its light form, considered as a \( Q_k \)-circuit \( C \), satisfies \( C \leq f \).

(b) Assume that \( C \leq f \) is a \( Q_k \)-circuit. Then there exists a Learner’s winning strategy \( S \) in the \( Q_k \)-hypotheses game for \( f \) such that the light form of \( S \) coincides with \( C \).

In fact, in the paper we only use the item (a) of this proposition. But for completeness, we also provide a proof of the item (b).

**Proof of the item (a) of Proposition 3.1.** For a node \( v \in V(G) \) let \( f_v \) be the function computed by the circuit \( C \) at the gate corresponding to \( v \). We shall prove the following statement. For any path \( p \) starting at \( s \) and for any \( z \) which is compatible with \( p \) it holds that \( f_v(z) = 0 \), where \( v \) is the endpoint of \( p \). To see why this implies \( C \leq f \), take any \( z \in f^{-1}(0) \) and note that \( z \) is compatible with the path which starts and ends at \( s \). The endpoint of this path is \( s \) and hence \( 0 = f_s(z) = C(z) \).

We will prove the above statement by backward induction on the length of \( p \). The longest path \( p \) ends in some \( t \in T(G) \). By definition, \( f_t = x_{i_t} \). On the other hand, since \( S \) is winning, \( z_{i_t} = 0 \) for any \( z \) compatible with \( p \). In other words, \( f_t(z) = 0 \) for any \( z \) compatible with \( p \). The base is proved.

The induction step is the same if \( p \) ends in some other terminal. Now assume that \( p \) ends in \( v \in V(G) \setminus T(G) \). Take any \( z \in f^{-1}(0) \) compatible with \( p \). Let \( p_1 \) be the extension of \( p \) by the edge
which starts at \( v \) and is labeled by \( j \in \{0, 1, \ldots, k\} \). Next, let \( v_j \) be the endpoint of \( p_j \) (note that \( v_0, v_1, \ldots, v_k \) are successors of \( v \)). Since \( S \) is winning, the number of \( j \in \{0, 1, \ldots, k\} \) such that \( z \in \mathcal{H}_j(p) \) is at least \( k \). Hence the number of \( j \in \{0, 1, \ldots, k\} \) such that \( z \) is compatible with \( p_j \) is at least \( k \). Finally, by the induction hypothesis this means that the number of \( j \in \{0, 1, \ldots, k\} \) such that \( f_{v_j}(z) = 0 \) is at least \( k \). On the other hand:

\[
f_v = \text{THR}^{-1}_2(f_{v_0}, f_{v_1}, \ldots, f_{v_k}).
\]

Therefore, \( f_v(z) = 0 \), as required.

\( \square \)

**Proof of the item (b) of Proposition 3.1.** The circuit \( C \) should be the light form of \( S \). So to define \( S \), it remains to define hypotheses of Learner in \( S \). Consider any non-input gate \( g \) of \( C \). Let \( g_0, \ldots, g_k \) be gates that are fed to \( g \), that is, \( g = \text{THR}^{k+1}_2(g_0, \ldots, g_k) \). For every path \( p \) from the output gate to \( g \) we define the following hypotheses:

\[
\mathcal{H}_0(p) = g_0^{-1}(0), \ldots, \mathcal{H}_k(p) = g_k^{-1}(0).
\]

Note that \( \mathcal{H}_0(p), \ldots, \mathcal{H}_k(p) \) depend only on the endpoint of \( p \).

We have to show that this strategy of Learner is winning in the \( Q_k \)-hypotheses game for \( f \). First, let us observe the following: for any path \( p \) and for any \( z \in f^{-1}(0) \) which is compatible with \( p \) it holds that \( g(z) = 0 \), where \( g \) is the gate at the endpoint of \( p \). Indeed, if \( p \) is of length 0, then \( g \) is the output gate of \( C \), and hence \( g(z) = C(z) \leq f(z) \). Otherwise, note that the game can come to a gate \( g \) only if previously Nature indicated that \( g(z) = 0 \).

Now, consider any path \( p \) and any \( z \in f^{-1}(0) \) which is compatible with \( p \). We have to show two things. First, if \( p \) ends in some non-input gate \( g \), then the number of \( j \in \{0, 1, \ldots, k\} \) such that \( z \in \mathcal{H}_j(p) \) is at least \( k \). Second, if \( p \) ends in an input variable \( x_i \), then \( z_i = 0 \). The second claim is already established. As for the first claim, note that if \( z \) is compatible with \( p \), then, as we have proved, \( g(z) = \text{THR}^{k+1}_2(g_0, \ldots, g_k)(z) = 0 \). That is, the number of \( j \in \{0, 1, \ldots, k\} \) such that \( g_j(z) = 0 \) is at least \( k \), as required.

\( \square \)

Similarly, one can define \( R_k \)-hypotheses games for functions \( f \in R_k \). The only difference this time is that the goal of Learner is to find an index \( i \in [n] \) and a bit \( b \in \{0, 1\} \) such that \( z_i = b \). Correspondingly, the leaves of Learner’s strategies in \( R_k \)-hypotheses games will be labeled by pairs from \([n] \times \{0, 1\} \). When we turn these strategies into circuits, we turn leaves that are labeled by \((i, 0)\) into \( x_i \), and leaves that are labeled by \((i, 1)\) into \( \neg x_i \).

The same argument as in Proposition 3.1 establishes the following proposition.

**Proposition 3.2.** For all Boolean functions \( f \in R_k \) the following holds:

(a) if \( S \) is a Learner’s winning strategy in the \( R_k \)-hypotheses game for \( f \), then its light form, considered as an \( R_k \)-circuit \( C \), satisfies \( C \leq f \).

(b) Assume that \( C \leq f \) is an \( R_k \)-circuit. Then there exists a Learner’s winning strategy \( S \) in the \( R_k \)-hypotheses game for \( f \) such that the light form of \( S \) coincides with \( C \).

**Remark 3.3.** It might be unclear why we prefer to construct strategies instead of constructing circuits directly, because beside the circuit itself we should also specify Learner’s hypotheses. The reason is that strategies can be seen as proofs that the circuit we construct is correct.
4 Results for Majority

Theorem 4.1 (Restatement of Theorem 1.1). There exists a polynomial-time computable \( \text{MAJ}_3 \)-formula of depth \( O(\log n) \) for \( \text{MAJ}_{2n+1} \).

Proof. Due to the AKS sorting network [1], there exists an algorithm which in \( n^{O(1)} \) time produces a monotone formula \( F \) of depth \( d = O(\log n) \) which computes \( \text{MAJ}_{2n+1} \). Below we will define a strategy \( S_F \) in the \( Q_2 \)-hypotheses game for \( \text{MAJ}_{2n+1} \). Strategy \( S_F \) will be winning for Learner. Moreover, its depth will be \( d + O(\log n) \). In the end of the proof, we will refer to Proposition 3.1 to show that \( S_F \) yields a polynomial-time computable \( \text{MAJ}_3 \)-formula of depth \( O(\log n) \) for \( \text{MAJ}_{2n+1} \).

Strategy \( S_F \) has two phases. The first phase does not use \( F \) at all, only the second phase does. The objective of the first phase is to find some distinct \( i, j \in [2n + 1] \) such that either \( z_i = 0 \land z_j = 1 \) or \( z_i = 1 \land z_j = 0 \), where \( z \) is the Nature’s vector. This can be done as follows.

Lemma 4.2. One can compute in polynomial time a 3-ary tree of depth \( O(\log n) \) with the set \( v(T) \) of nodes, and a mapping \( w : v(T) \to 2^{[2n+1]} \) such that the following holds:

- if \( r \) is the root of \( T \), then \( w(r) = [2n + 1] \);
- if \( v \) is not a leaf of \( T \) and \( v_1, v_2, v_3 \) are 3 children of \( v \), then every element of \( w(v) \) is covered at least twice by \( w(v_1), w(v_2), w(v_3) \);
- if \( l \) is a leaf of \( T \), then \( w(r) \) is of size 2.

Proof. We start with a trivial tree, consisting only of the root, to which we assign \([2n + 1]\). Then at each iteration we do the following. We have a 3-ary tree in which nodes are assigned to some subsets of \([2n + 1]\). If every leaf is assigned to a set of size 2, we terminate. Otherwise, we pick any leaf \( l \) of the current tree which is assigned to a subset \( A \subseteq [2n + 1] \) of size at least 3. We split \( A \) into 3 disjoint subsets \( A_1, A_2, A_3 \) of sizes \(|A|/3\), \(|A|/3\) and \(|A| - 2\lfloor |A|/3 \rfloor\). We add 3 children to \( l \) (which become new leaves) and assign \( A_1 \cup A_2, A_1 \cup A_3, A_2 \cup A_3 \) to them.

The sizes of \( A_1 \cup A_2, A_1 \cup A_3, A_2 \cup A_3 \) do not exceed \(|A| - |A|/3| = |A| - |A|/3 + 2/3 = 2/3 \cdot (|A| + 1) \leq 2/3 \cdot (|A| + |A|/3) = 8/9 \cdot |A|\). Hence, the size of the set assigned to a node of depth \( h \) is at most \( \left( 8/9 \right)^h \cdot (2n + 1) \). This means that at any moment, the depth of the tree is at most \( \log_{9/8}(2n + 1) = O(\log n) \). Therefore, we terminate in \( 3^{O(\log n)} = n^{O(1)} \) iterations, because at each iteration we add 3 new nodes. Each iteration takes polynomial time.

We use \( T \) to find \( i, j \in [2n + 1] \) such that either \( z_i = 0 \) or \( z_j = 0 \). Namely, we descend from the root of \( T \) to one of its leaves. Learner maintains an invariant that the leftmost 0-coordinate of \( z \) is in \( w(v) \), where \( v \) is the current node of \( T \). Let \( v_1, v_2, v_3 \) be 3 children of \( v \). Learner, for every \( i \in [3] \), makes a hypothesis that the leftmost 0-coordinate of \( z \) is in \( w(v_i) \). Due to the properties of \( w \), at least two hypotheses are true. Nature indicates some \( v_i \) for which this is true, and Learner descends to \( v_i \). When Learner reaches a leaf, it knows a set of size two containing the leftmost 0-coordinate of \( z \). Let this set be \( \{i, j\} \).

Learner knows that either \( z_i \) or \( z_j \) is 0. Thus, \( (z_i, z_j) \in \{(0, 0), (0, 1), (1, 0)\} \). At the cost of one round, Learner can ask Nature to identify an element of \( \{(0, 0), (0, 1), (1, 0)\} \) which differs from
Assume that Learner knows a non-input gate phase.

Here (satisfies:

$$\text{exactly one of the following three statements is true for that argument, (4.1) holds for that},$$

If Nature says that (4.2) is false for true and two are false:

Second, the last gate $g_{d'}$ is an input variable (i.e., a leaf of $F$). Third, each $g \in \{g_0, g_1, \ldots, g_{d'}\}$ satisfies:

$$\left( g(z) = 0 \land (z_i, z_j) = (0, 1) \right) \lor \left( g(\neg z) = 1 \land (z_i, z_j) = (1, 0) \right).$$

Here $\neg z$ denotes the bitwise negation of $z$.

At the beginning, Learner sets $g_0 = g_{\text{out}}$ to be the output gate of $F$. Let us explain why (4.1) holds for $g_{\text{out}}$. Nature’s vector is an element of $\text{MAJ}_{2^{n+1}}(0)$. That is, the number of ones in $z$ is at most $n$. In turn, in $\neg z$ there are at least $n + 1$ ones. Since $g_{\text{out}}$ computes $\text{MAJ}_{2^{n+1}}$, we have that $g_{\text{out}}(z) = 0$ and $g_{\text{out}}(\neg z) = 1$. On the other hand, as guaranteed after the first phase, we have that $(z_i, z_j) = (0, 1) \lor (z_i, z_j) = (1, 0)$.

Assume now that the second phase is finished, that is, Learner has produced some $g_{d'} = x_k$ satisfying (4.1). Then by (4.1) either $g_{d'}(z) = z_k = 0$ or $g_{d'}(\neg z) = (\neg z)_k = 1$. We have $z_k = 0$ in both cases. Hence, Learner can output $k$.

It remains to explain how to fulfill the second phase. It is enough to show the following. Assume that Learner knows a non-input gate $g_l$ of $F$ of depth $l$ satisfying (4.1). Then in one round it can either find a gate $g_{l+1}$ of depth $l + 1$ satisfying (4.1) or give a correct answer to the game.

The gate $g_{l+1}$ will be one of the two gates which are fed to $g_l$. Assume first that $g_l$ is an $\land$-gate and $g_l = u \land v$. From (4.1) we conclude that among the following three statements one is true and two are false:

$$u(z) = 0 \text{ and } (z_i, z_j) = (0, 1),$$

$$u(z) = 1, v(z) = 0 \text{ and } (z_i, z_j) = (0, 1),$$

$$u(\neg z) = v(\neg z) = 1 \text{ and } (z_i, z_j) = (1, 0).$$

At the cost of one round Learner can ask Nature to indicate one statement which is false for $z$. If Nature says that (4.2) is false for $z$, then (4.1) holds for $g_{l+1} = v$ (because (4.1) follows from the disjunction of (4.3) and (4.4)). Next, if Nature says that (4.3) is false for $z$, then, by the same argument, (4.1) holds for $g_{l+1} = u$. Finally, if Nature says that (4.4) is false for $z$, then we know that $(z_i, z_j) = (0, 1)$, i.e., Learner can already output $i$.

One can deal in the same way with the case when $g_l$ is an $\lor$-gate and $g_l = u \lor v$. By (4.1) exactly one of the following three statements is true for $z$:

$$u(z) = v(z) = 0 \text{ and } (z_i, z_j) = (0, 1),$$

$$u(\neg z) = 1 \text{ and } (z_i, z_j) = (1, 0),$$

$$u(\neg z) = 0, v(\neg z) = 1 \text{ and } (z_i, z_j) = (1, 0).$$
Similarly, Learner asks Nature to indicate one statement which is false for z. If Nature says that (4.5) is false for z, then \((z_i, z_j) = (1, 0)\), i.e., Learner can output \(j\). Next, if Nature says that (4.6) is false for z, then (4.1) holds for \(g_{l+1} = v\). Finally, if Nature says that (4.7) is false for z, then (4.1) holds for \(g_{l+1} = u\).

Thus, \(S_F\) is a winning strategy of depth \(O(\log n)\) of Learner. Apply \text{Proposition 3.1} to \(S_F\). We obtain a MAJ\(_3\)-formula \(F' \leq \text{MAJ}_{2n+1}\) of depth \(O(\log n)\). In fact, \(F'\) computes \(\text{MAJ}_{2n+1}\). Indeed, \(F' \leq \text{MAJ}_{2n+1}\) means that \(F'\) outputs 0 on every input with at most \(n\) ones. On the other hand, \(F'\) consists of MAJ\(_3\) gates and hence \(F'\) computes a self-dual function (that is, it outputs opposite values in opposite vertices of the Boolean cube). Therefore, \(F'\) outputs 1 on every input with at least \(n + 1\) ones.

It remains to explain how to compute \(F'\) in polynomial time. To do so, by \text{Proposition 3.1} it is sufficient to compute in polynomial time the light form of \(S_F\), i.e., the underlying tree of \(S_F\) and the outputs of Learner in the leaves. It is easy to see that the light form of \(S_F\) is arranged as follows.

First, compute \(F\) and compute \(T\) from \text{Lemma 4.2}. For each leaf \(l\) of \(T\) do the following. Let \(w(l) = \{i, j\}\). Add 3 children to \(l\). Two of them will be leaves of \(S_F\), labeled by \(i\) and \(j\). We then attach a tree of \(F\) to the remaining child of \(l\). Then we add to every non-leaf node of \(F\) one more child so that now the tree of \(F\) is 3-ary. Each added child is a leaf of \(S_F\). If a child was added to an \(\land\)-gate, then Learner outputs \(i\) in this child. In turn, if a child was added to an \(\lor\)-gate, then Learner outputs \(j\) in it. Finally, there are leaves that were in \(F\) initially, each labeled by some input variable. In these nodes, Learner outputs the index of the corresponding input variable. ∎

**Theorem 4.3 (Restatement of Theorem 1.2).** If there is a monotone formula for \(\text{MAJ}_{2n+1}\) of size \(s\), then there is a MAJ\(_3\)-formula for \(\text{MAJ}_{2n+1}\) of size \(O(s \cdot n^{\log_3(3)}) = O(s \cdot n^{1.58\ldots})\).

**Proof.** Take any monotone formula \(F\) for \(\text{MAJ}_{2n+1}\) whose size is \(s\), and consider the corresponding Learner’s strategy \(S_F\) defined in the previous proof. Recall that \(S_F\) has two phases. The goal of the first phase is to find some \(i, j \in [2n + 1]\) such that either \(z_i = 0 \land z_j = 1\) or \(z_i = 1 \land z_j = 0\). To show the theorem, it is sufficient to accomplish the first phase in \(\log_3 n + O(1)\) rounds. Indeed, then the tree of \(S_F\) is a ternary tree of depth \(\log_3 n + O(1)\) with trees of the same size as formula \(F\) attached to leaves. Overall, its size is \(O(3^{\log_3(n)} + O(1) \cdot s) = O(n^{\log_3(3)} \cdot s)\).

A difference from the previous proof is that this time we do not care about explicitness. In the explicit construction from the previous proof we fulfil the first phase in \(\log_{3/2}(n) + O(1)\) rounds (in fact, we only bounded it from above by \(\log_{9/8}(n) + O(1)\), to avoid technical details). To improve it, we need the following lemma, which will be proved by the probabilistic method.

**Lemma 4.4.** There exists a formula \(D\) with the following properties:

- formula \(D\) is a complete ternary tree of depth \([\log_3(n)] + 10\);
- every non-leaf node of \(D\) contains a MAJ\(_3\)-gate and every leaf of \(D\) contains a conjunction of 2 variables;
- \(D(x) = 0\) for every \(x \in \{0, 1\}^{2n+1}\) with at most \(n\) ones.
Let us at first explain how to use formula $D$ from Lemma 4.4 to accomplish the first phase in $\log_2 n + O(1)$ rounds. First, as explained in the previous proof, it is sufficient to find two indices $i, j \in [2n + 1]$ such that either $z_i = 0$ or $z_j = 0$. To do so Learner, descends from the output gate of $D$ to some of its leaves. It maintains an invariant that for its current gate, it holds that 

$I_8 = 0$ or $I_9 = 0$. For the output gate, the invariant is true because by Lemma 4.4 $D$ is 0 on all Nature’s possible vectors. If we reached a leaf so that $I_8$ is a conjunction of two variables $I_8$ and $I_9$, then the first phase is fulfilled (by the invariant, $I_8 \land I_9 = 0$). Finally, if $g_i$ is a non-leaf node of $g$, i.e., a MAJ$_3$-gate, then in one round we can descend to one of the children of $g$, without violating the invariant. Indeed, since $I_8 = 0$, the same is true for at least 2 children of $g_i$. For each child $g_i$ of $g$ Learner makes a hypothesis that $g_i(z) = 0$. Any Nature’s response allows us to replace $g_i$ by some $g_i$.

**Proof of Lemma 4.4.** Independently for each leaf $l$ of $D$ choose $(i, j) \in [2n + 1]^2$ uniformly at random and put the conjunction $z_i \land z_j$ into $l$. It is enough to demonstrate that for any $x \in \{0, 1\}^{2n+1}$ with at most $n$ ones it holds that $\Pr[D(x) = 1] < 2^{-2n-1}$.

To do so, we use a modification of a standard Valiant’s argument. For any fixed $x$ with at most $n$ ones, let $p$ be the probability that a leaf $l$ of $D$ equals 1 on $x$. This probability is the same for all leaves and it does not exceed 1/4. Now, observe that:

$$\Pr[D(x) = 1] = \frac{f(f(\ldots f(p))) \ldots}{\lceil \log_2(n) \rceil + 10},$$

where $f(t) = t^3 + 3t^2(1-t) = 3t^2 - 2t^3$. Since, $3f(t) \leq (3t)^2$, we have:

$$3 \Pr[D(x) = 1] \leq (3p)^{2\lceil \log_2(n) \rceil + 10} \leq (3/4)^{1000n} < (1/2)^{-2n-1}.$$ 

□

5 Proof of the Main Theorem

Theorem 1.7 follows from Proposition 5.1 (Subsection 5.1) and Proposition 5.3 (Subsection 5.2). In turn, Theorem 1.8 follows from Proposition 5.2 (Subsection 5.1) and Proposition 5.8 (Subsection 5.2).

5.1 From circuits to protocols

**Proposition 5.1.** For any constant $k \geq 2$ the following holds. Assume that $f \in Q_k$ and $C \leq f$ is a $Q_k$-circuit. Then there is a protocol $\pi$, computing the $Q_k$-communication game for $f$, such that depth($\pi$) = $O$(depth($C$)).
Proof. Let the inputs of parties be \(z^1, \ldots, z^k \in f^{-1}(0)\). Parties descend from the output gate of \(C\) to one of the inputs. They maintain an invariant that for the current gate \(g\) of \(C\) it holds that \(g(z^1) = g(z^2) = \ldots = g(z^k) = 0\). If \(g\) is not yet an input, then \(g = \text{THR}_{2}^{k+1}(g_0, \ldots, g_k)\) for some gates \(g_0, \ldots, g_k\). We have for each \(z^i\) that \(g(z^i) = \text{THR}_{2}^{k+1}(g_0(z^i), \ldots, g_k(z^i)) = 0\). Hence for each \(z^i\) there is at most one gate out of \(g_0, \ldots, g_k\) satisfying \(g_j(z^i) = 1\). This means that in \(O(1)\) bits of communication parties can agree on the index \(j \in \{0, 1, \ldots, k\}\) satisfying \(g_j(z^1) = g_j(z^2) = \ldots = g_j(z^k) = 0\).

Thus, in \(O(\text{depth}(\pi))\) bits of communication they reach some input of \(C\). If this input contains a variable \(x_i\), then by the invariant we have \(z^1_i = z^2_i = \ldots = z^k_i = 0\), as required. \(\square\)

The same argument can be used to show the following proposition.

**Proposition 5.2.** For any constant \(k \geq 2\) the following holds. Assume that \(f \in \text{R}_{k}\) and \(C \leq f\) is an \(\text{R}_{k}\)-circuit. Then there is a protocol \(\pi\), computing the \(\text{R}_{k}\)-communication game for \(f\), such that \(\text{depth}(\pi) = O(\text{depth}(C))\).

### 5.2 From protocols to circuits

**Proposition 5.3.** For every constant \(k \geq 2\) the following holds. Let \(f \in \text{Q}_{k}\). Assume that \(\pi\) is a communication protocol computing the \(\text{Q}_{k}\)-communication game for \(f\). Then there is a \(\text{Q}_{k}\)-circuit \(C \leq f\) such that \(\text{depth}(\pi) = O(\text{depth}(C))\).

**Proof.** Set \(d = \text{depth}(\pi)\). By Proposition 3.1, it is enough to give an \(O(d)\)-round winning strategy of Learner in the \(\text{Q}_{k}\)-hypotheses game for \(f\).

We will use the following terminology. First, consider any subset \(Z \subseteq f^{-1}(0)\), any set \(C\) and any function \(g : Z \rightarrow C\). Then the \emph{g-value} of a tuple \((z^1, \ldots, z^k) \in Z^k\) is a vector \((g(z^1), \ldots, g(z^k)) \in C^k\).

Let \(V\) be the set of all nodes of the protocol \(\pi\) and let \(T\) be the set of all terminals of the protocol \(\pi\). Consider any set \(U \subseteq V\) and any set \(Z \subseteq f^{-1}(0)\). The following definition is crucial for our argument.

**Definition 5.4.** We say that \(U\) is \emph{complete} for \(Z\) if there exist a set \(C\) of size \(k\) and a function \(g : Z \rightarrow C\) with the following property: for every vector \(\tilde{c} \in C^k\) there exists a node \(u \in U\) such that all tuples from \(Z^k\) whose \(g\)-value is \(\tilde{c}\) visit \(u\) in the protocol \(\pi\). We also say that such \(g\) establishes \emph{completeness} of \(U\) for \(Z\).

In other words, \(U\) is complete for \(Z\) if there is a way of partitioning \(Z\) into at most \(k\) parts such that the following holds. Assume that somebody takes a tuple \((z^1, \ldots, z^k) \in Z^k\) and for every \(i \in [k]\) tells us the part of \(Z\) to which \(z^i\) belongs. Then we can determine a node \(u \in U\) such that the tuple \((z^1, \ldots, z^k)\) visits \(u\) in the protocol \(\pi\).

We now describe the Learner’s strategy. It proceeds in \(d\) iterations, each iteration takes \(O(1)\) rounds of the \(\text{Q}_{k}\)-hypotheses game. Now, we say that a set of nodes \(U \subseteq V\) is \emph{h-low} if all nodes of \(U\) that are not terminals are of depth at least \(h\). Learner maintains the following invariant.

**Invariant 5.5.** After \(h\) iterations, there exists an \(h\)-low set of nodes \(U\) which is complete for the set \(Z_h\) of all \(z \in f^{-1}(0)\) that are compatible with Nature’s responses after \(h\) iterations.
Let us first explain why this invariant holds in the beginning. We need to establish a 0-low set $U$ which is complete for $f^{-1}(0)$. We can take $U = \{s\}$, where $s$ is the starting node of $\pi$. This is because every tuple visits $s$ in the protocol $\pi$.

Next, let us explain that if Invariant 5.5 holds after $d$ iterations, then Learner is able to produce a correct answer to the $Q_k$-hypotheses game for $f$. Indeed, there exists a $d$-low set of nodes $U$ which is complete for $Z_d$. Note that $U$ consists only of terminals. Therefore, it is sufficient to establish the following lemma.

**Lemma 5.6.** Assume that $U \subseteq T$ is complete for $Z \subseteq f^{-1}(0)$. Then there exists $i \in [n]$ such that $z_i = 0$ for every $z \in Z$.

**Proof.** If $Z$ is empty, then there is nothing to prove. Otherwise, take $g: Z \rightarrow C$, $|C| = k$ which establishes completeness of $U$ for $Z$. Consider any vector $\vec{c} = (c_1, \ldots, c_k) \in C^k$ such that $\{c_i | i \in [k]\} = g(Z)$. There exists a node $u \in U$ such that any tuple from $Z^k$ whose $g$-value is $\vec{c}$ visits $u$. Note that $u$ is a terminal of $\pi$. Let $i \in [n]$ be the output of $\pi$ in $u$. We show that for any $z \in Z$ it holds that $z_i = 0$. Indeed, note that there exists a tuple $\vec{z} \in Z^k$ whose $g$-value is $\vec{c}$ and which includes $z$. This tuple visits $u$. Since $\pi$ computes the $Q_k$-communication game for $f$, every element of the tuple $\vec{z}$ should have 0 at the $i$th coordinate. In particular, this holds for $z$.

Finally, we describe how to maintain Invariant 5.5. Assume that it holds after $h$ iterations. Let us show how to perform the next iteration to maintain the invariant. We need a notion of a communication profile for that.

The communication profile of $z \in f^{-1}(0)$ with respect to a set of nodes $U \subseteq V$ is a function $p_z: U \rightarrow \{0, 1\}$, defined as follows. Take any $v \in U$. If $v$ is a terminal, set $p_z(v) = 0$. Otherwise, let $i \in [k]$ be the index of the party communicating at $v$. Set $p_z(v)$ to be the bit transmitted by the $i$th party at $v$ on input $z$. In the words, the communication profile of $z$ w.r.t. $U$ stores how all the parties communicate at nodes of $U$ on input $z$.

We also define the communication profile of a tuple $(z^1, \ldots, z^k) \in (f^{-1}(0))^k$ as $(p_{z^1}, \ldots, p_{z^k})$.

**Lemma 5.7.** Let $(z^1, \ldots, z^k), (y^1, \ldots, y^k) \in (f^{-1}(0))^k$ be two inputs visiting the same node $v \in V \setminus T$. Assume that their communication profiles with respect to $\{v\}$ coincide. Then these two inputs visit the same successor of $v$.

**Proof.** Let their common communication profile with respect to $\{v\}$ be $(p_1, \ldots, p_k)$. Next, assume that $i$ is the index of the party communicating at $v$. Then where these inputs descend from $v$ is determined by $p_i$. 

Here is what Learner does during the $(h + 1)$st iteration. It takes any $h$-low $U$ which is complete for $Z_h$. Then it takes any $g: Z_h \rightarrow C$, $|C| = k$ which establishes completeness of $U$ for $Z_h$. Note that w.l.o.g. $U$ is of size at most $k^k$. This is because for any vector $\vec{c} \in C^k$ we need exactly one node in $U$ for $\vec{c}$ to establish completeness.

Learner now devises a new function $g'$ whose domain is the set $Z_h$. The value of $g'(z)$ is a pair $(p_z, g(z))$, where $p_z$ is a communication profile of $z$ with respect to $U$. There are at most
we conclude that 

2^{|U|} \leq 2^k \text{ different communication profiles with respect to } U. \text{ Hence, the image of } g' \text{ is of size at most } 2^k \cdot k = O(1).

The goal of the \((h+1)\)st iteration is to narrow down the image of \(g'\) to a set of size \(k\). Learner does this as follows. While there exist \(k+1\) different possible values of \(g'\), Learner asks Nature to reject one of them. More specifically, for each of these \(k+1\) values, Learner make a hypothesis that \(g'(z)\) differs from this value. As the size of the image of \(g'\) in the beginning is \(O(1)\), this process takes \(O(1)\) rounds of the \(Q_k\)-hypotheses game. In the end of the \((h+1)\)st iteration, we are left with \(k\) possible values of \(g'\). Denote them by \((p_1, c_1), \ldots, (p_k, c_k)\).

In other words, we know that \(g'(Z_{h+1}) \subseteq \{(p_1, c_1), \ldots, (p_k, c_k)\}\) (recall that \(Z_{h+1}\) is the set of \(z \in f^{-1}(0)\) that are compatible with the Nature’s responses after \(h+1\) iterations). We show that \(g': Z_{h+1} \rightarrow \{(p_1, c_1), \ldots, (p_k, c_k)\} = C\) establishes completeness of some \((h+1)\)-low set of nodes \(U'\) for \(Z_{h+1}\). This will establish Invariant 5.5 after the \((h+1)\)st iteration.

Take any vector \(\vec{c} \in (C')^k\). It is enough to show that all the inputs from \((Z_{h+1})^k\) whose \(g'\)-value is \(\vec{c}\) visit the same node \(v'\) which is either a terminal or of depth at least \(h+1\). Then we just set \(U'\) to be the union of all such \(v'\) over all \(\vec{c} \in (C')^k\).

All tuples from \((Z_{h+1})^k\) with the same \(g'\)-value visit the same node \(v \in U\). This is because \(g'\)-value of a tuple determines its \(g\)-value, and hence we can use Invariant 5.5 for \(Z_h\) here. If \(v\) is a terminal, there is nothing left to prove. Otherwise, note that \(g'\)-value of a tuple also determines its communication profile with respect to \(U\), and hence with respect to \(\{v\} \subseteq U\). Therefore, by Lemma 5.7, all tuples with this \(g'\)-value visit the same successor of \(v\). \(\square\)

With straightforward modifications, one can obtain a proof of the following:

**Proposition 5.8.** For every constant \(k \geq 2\) the following holds. Let \(f \in R_k\). Assume that \(\pi\) is a communication protocol computing the \(R_k\)-communication game for \(f\). Then there is an \(R_k\)-circuit \(C \leq f\) such that \(depth(C) = O(depth(\pi))\).

**Corollary 5.9 (Weak version of Theorem 1.3).** For any constant \(k \geq 2\) there exists a \(Q_k\)-formula of depth \(O(\log^2 n)\) for \(\text{THR}_{n+1}^{kn+1}\).

**Proof.** We will show that there exists a protocol \(\pi\) of depth \(O(\log^2 n)\) computing the \(Q_k\)-communication game for \(\text{THR}_{n+1}^{kn+1}\). By Proposition 5.3 this means that there is a \(Q_k\)-formula \(F \leq \text{THR}_{n+1}^{kn+1}\) of depth \(O(\log^2 n)\).

It is easy to see that \(F\) actually coincides with \(\text{THR}_{n+1}^{kn+1}\). Indeed, assume for contradiction that \(F(x) = 0\) for some \(x\) with at least \(n+1\) ones. Then it is easy to construct \(x^2, \ldots, x^k\), each with \(n\) ones, such that \(x, x^2, \ldots, x^k\) have no common 0-coordinate. Since \(F(x) = F(x^2) = \ldots = F(x^k) = 0\), we conclude that \(F\) does not have the \(Q_k\)-property. But \(F\) is a \(Q_k\)-formula, so it gives a contradiction with Proposition 1.5.

Let \(\pi\) be the following protocol. Assume that the inputs to parties are \(x^1, x^2, \ldots, x^k \in \{0, 1\}^{kn+1}\). Without loss of generality, we may assume that each \(x^r\) has exactly \(n\) ones. For \(x \in \{0, 1\}^{kn+1}\) define \(\text{supp}(x) = \{i \in [kn+1] \mid x_i = 1\}\). Let \(T\) be a binary rooted tree of depth \(d = \log_2(n) + O(1)\) with \(kn+1\) leaves. Identify leaves of \(T\) with elements of \([kn+1]\). For a node \(v\) of \(T\), let \(T_v\) be the set of all leaves of \(T\) that are descendants of \(v\). Once again, we view \(T_v\) as a subset of \([kn+1]\).
The protocol proceeds in at most \( d \) iterations. After \( i \) iterations, for \( i = 0, \ldots, d \), parties agree on a node \( v \) of \( T \) of depth \( i \), satisfying the following invariant:

\[
\sum_{r=1}^{k} |\text{supp}(x^r) \cap T_v| < |T_v|.
\]

(5.1)

At the beginning, Invariant (5.1) holds just because \( v \) is the root, \( T_v = \{kn + 1\} \) and each \( \text{supp}(x^r) \) is of size \( n \).

After \( d \) iterations, \( v = l \) is a leaf of \( T \). Parties output \( l \). This is correct because by (5.1) we have \(|T_l| = 1 \implies |\text{supp}(x^r) \cap T_l| = 0 \implies x^r_l = 0 \) for every \( r \in [k] \).

Let us now explain what parties do at each iteration. If the current \( v \) is not a leaf, let \( v_0, v_1 \) be two children of \( v \). Each party sends \( |\text{supp}(x^r) \cap T_{v_0}| \) and \( |\text{supp}(x^r) \cap T_{v_1}| \), using \( O(\log n) \) bits. Since \( T_{v_0} \) and \( T_{v_1} \) is a partition of \( T_v \), we have:

\[
\sum_{l=0}^{1} \sum_{r=1}^{k} |\text{supp}(x^r) \cap T_{v_l}| = \sum_{r=1}^{k} |\text{supp}(x^r) \cap T_v| < |T_v| = \sum_{l=0}^{1} |T_{v_l}|.
\]

Thus the inequality:

\[
\sum_{r=1}^{k} |\text{supp}(x^r) \cap T_{v_0}| < |T_{v_0}|
\]

(5.2)

is true either for \( b = 0 \) or for \( b = 1 \). Let \( b^* \) be the smallest \( b \in \{0, 1\} \) for which (5.2) is true. Parties replace \( v \) by \( v_{b^*} \) and proceed to the next iteration.

There are \( d = O(\log n) \) iterations, each takes \( O(\log n) \) bits of communication. Hence \( \pi \) has depth \( O(\log^2 n) \), as required.

\[\square\]

**Remark 5.10.** The strategy from the proof of Proposition 5.3 is efficient only in terms of the number of rounds. In the next section, we present another version of this strategy. It will give us not only low depth but also explicit polynomial-size circuits. For that, however, we require a bit more from protocols for the \( Q_k \)-communication games.

### 6 Effective version

Fix \( f \in Q_k \). We say that a dag-like communication protocol \( \pi \) *strongly* computes the \( Q_k \)-communication game for \( f \) if for every terminal \( t \) of \( \pi \) and for every \( x \in f^{-1}(0) \) the following holds. If \( x \) is \( i \)-compatible with \( t \) for some \( i \in [k] \), then \( x_j = 0 \), where \( j = l(t) \) is the label of the terminal \( t \) in the protocol \( \pi \). In other words, there should be a path \( p \) to \( t \) such that, first, \( x_{l(t)} = 0 \), and second, one of the parties is compatible with this path on input \( x \). That is, for every node of \( p \) from where this party is the one to communicate, it communicates along \( p \) on \( x \). We stress that there might be no tuple from \((f^{-1}(0))^k\) which includes \( x \) and visits \( t \). Hence, a protocol which computes the \( Q_k \)-hypotheses game for \( f \) might not compute it in the strong sense.

Similarly, fix \( f \in R_k \). We say that a dag-like communication protocol \( \pi \) *strongly* computes the \( R_k \)-communication game for \( f \) if for every terminal \( t \) of \( \pi \) and for every \( x \in f^{-1}(0) \) the following
We compute what would be a configuration working tape takes polynomial time given the light form of working tape, but the result must depend only on the current content of the working tape and whose size is polynomial in \( \sigma \).

Next, we prove an effective version of Proposition 5.3.

**Theorem 6.1.** For every constant \( k \geq 2 \) there exists a polynomial-time algorithm \( A \) such that the following holds. Assume that \( f \in Q_k \) and \( \pi \) is a dag-like protocol which strongly computes the \( Q_k \)-communication game for \( f \). Then, given the light form of \( \pi \), the algorithm \( A \) outputs a \( Q_k \)-circuit \( C \leq f \) such that depth(\( C \)) is linear in depth(\( \pi \)) and size(\( C \)) is polynomial in size(\( \pi \)).

**Proof.** Let \( d = \text{depth}(\pi) \). We will again give an \( O(d) \)-round winning strategy of Learner in the \( Q_k \)-hypotheses game for \( f \). This time, however, we will ensure that, given the light form of \( \pi \), the light form of our strategy can be computed in polynomial time (in particular, its size will be polynomial in size(\( \pi \))). By Proposition 3.1, this will give us a \( Q_k \)-circuit \( C \leq f \) of depth \( O(d) \) whose size is polynomial in size(\( \pi \)) and which is polynomial-time computable from the light form of \( \pi \).

Instead of specifying the light form of our strategy directly, we will use the following trick. Assume that Learner has a working tape consisting of \( O(\text{size}(\pi)) \) cells, where each cell can store one bit. Learner memorizes all the Nature’s responses so that it always knows the current position of the game. But it does not store the sequence of Nature’s responses on the working tape (there might be no space for it). Instead, it first makes its hypotheses which depend on the current position in a game. Moreover, this should take polynomial time given the light form of \( \pi \).

The main purpose of the working tape manifests itself in the end. Namely, at some point, Learner decides to stop making hypotheses. This should be indicated on the working tape. More importantly, Learner’s output must depend only on the content of the working tape in the end (and not on the whole sequence of Nature’s responses). Moreover, this should take polynomial time to compute that output, given the light form of \( \pi \).

If a strategy satisfies these restrictions, then its light form is polynomial-time computable from the light form of \( \pi \). Indeed, the underlying dag will consist of all possible configurations of the working tape. Their number is polynomial in size(\( \pi \)), because there are \( O(\text{size}(\pi)) \) bits on the working tape. For all non-terminal configurations \( c \) we go through all \( r \in \{0, 1, \ldots, k\} \). We compute what would be a configuration \( c_r \) of the working tape if the current configuration is \( c \) and Nature’s response is \( r \). After that we connect \( c \) to \( c_0, c_1, \ldots, c_k \). Finally, we compute the outputs of Learner in all terminal configurations. This gives the light form of our strategy in time polynomial in size(\( \pi \)).

Let \( V \) be the set of nodes of \( \pi \) and \( T \) be the set of terminals of \( \pi \). We will work with multidimensional arrays of nodes. Namely, we will consider \( k \)-dimensional arrays in which every dimension is indexed by integers from \([k]\). Formally, such arrays are functions of the form \( M: [k]^k \to V \). We will use the notation \( M[c_1, \ldots, c_k] \) for the value of \( M \) on \((c_1, \ldots, c_k) \in [k]^k \).

We will use a slightly stronger notion of completeness than in the proof of Proposition 5.3.

**Definition 6.2.** We say that a multidimensional array \( M: [k]^k \to V \) is complete for a set...
$\mathcal{Z} \subseteq f^{-1}(0)$ if there exists a function $g$: $\mathcal{Z} \to [k]$ such that the following holds. For every $z \in \mathcal{Z}$, for every $i \in [k]$ and for every $(c_1, \ldots, c_k) \in [k]^k$ such that $c_i = g(z)$ it holds that the node $M[c_1, \ldots, c_k]$ is $i$-compatible with $z$ in the protocol $\pi$. We also say that such $g$ establishes completeness of $M$ for $\mathcal{Z}$.

We stress that in this definition $M[c_1, \ldots, c_k]$ should be $i$-compatible with $z$ even if there is no tuple $(z_1, \ldots, z_k)$ with $(g(z_1), \ldots, g(z_k)) = (c_1, \ldots, c_k)$. Intuitively, we can afford such a strong notion of completeness (compared to the proof of Proposition 5.3) because this time $\pi$ computes the $Q_k$-communication game for $f$ in the strong sense.

We now describe the Learner’s strategy. It proceeds in $d$ iterations, each iteration takes $O(1)$ rounds of the $Q_k$-hypotheses game. The working tape of Learner consists of:

- an integer $\text{iter}$;
- a multidimensional array $M$: $[k]^k \to V$;
- $O(1)$ additional bits of memory.

The integer $\text{iter}$ will never exceed $d \leq \text{size}(\pi)$, so to store all this information we will need $O(\log(\text{size}(\pi)))$ bits, as required. At each moment, $\text{iter}$ equals the number of iterations performed so far (at the beginning, $\text{iter} = 0$). Learner updates $M$ only at moments when $\text{iter}$ is incremented by 1. So let $M_h$ denote the content of the array $M$ when $\text{iter} = h$ (that is, after $h$ iterations). Learner stops making hypotheses when $\text{iter} = d$ (that is, after $d$ iterations).

We call an array of nodes $h$-low if every node in it is either a terminal or of depth at least $h$. Learner maintains the following invariant.

**Invariant 6.3.** $M_h$ is $h$-low and $M_h$ is complete for the set $\mathcal{Z}_h$ of all $z \in f^{-1}(0)$ that are compatible with Nature’s responses after $h$ iterations.

At the beginning, Learner sets every element of $M_0$ to be the starting node of $\pi$ so that Invariant 6.3 trivially holds.

Now, let us show that when $\text{iter} = d$, Learner is able to output a correct answer to the $Q_k$-hypotheses game in polynomial time, knowing only the current content of the working tape and the light form of $\pi$. Indeed, observe that $M_d$ consists only of terminals. Hence it is sufficient to establish the following lemma.

**Lemma 6.4.** Assume that $M$: $[k]^k \to T$ is complete for $\mathcal{Z} \subseteq f^{-1}(0)$. Let $l$ be the output of $\pi$ in the terminal $M[1, 2, \ldots, k]$. Then $z_l = 0$ for every $\mathcal{Z}$.

**Proof.** Since $\pi$ strongly computes the $Q_k$-communication game for $f$, it is enough to show that every $z \in \mathcal{Z}$ is $i$-compatible with $M[1, 2, \ldots, k]$ for some $i \in [k]$. Take $g$: $\mathcal{Z} \to [k]$ establishing completeness of $M$ for $\mathcal{Z}$. By definition, $z$ is $g(z)$-compatible with $M[1, 2, \ldots, k]$. \qed

Finally, we need to perform an iteration. Assume that $h$ iterations passed and Invariant 6.3 still holds. Let $U_h$ be the set of nodes appearing in $M_h$. Take any function $g$: $\mathcal{Z}_h \to [k]$ establishing completeness of $M_h$ for $\mathcal{Z}_h$.  

*THEORY OF COMPUTING,* Volume 18 (15), 2022, pp. 1–33

24
For any $z \in f^{-1}(0)$ we denote by $p_z$ a communication profile of $z$ with respect to $U_h$ (we use the same notion of a communication profile as in the proof of Proposition 5.3). Recall that $p_z$ is an element of $\{0, 1\}^{U_h}$, i.e., a function from $U_h$ to $\{0, 1\}$. Learner wants to gain some information about the pair $(p_z, g(z))$. In the beginning, Learner only knows that $(p(z), g(z)) \in \{0, 1\}^{U_h} \times [k]$. His goal is to narrow down the set of all possible values of the pair $(p(z), g(z))$ to $k$ values. He does so in the same manner as in the proof of Proposition 5.3. Namely, in each round Learner asks Nature to specify some $(p, c) \in \{0, 1\}^{U_h} \times [k]$ such that $(p(z), g(z)) \neq (p, c)$. Learner can do this until there are only $k$ pairs from $(p_1, c_1), \ldots, (p_k, c_k) \in \{0, 1\}^{U_h} \times [k]$ left which are not rejected by Nature. Learner stores each rejected $(p, c)$ on the working tape so that in the end he knows $(p_1, c_1), \ldots, (p_k, c_k)$. This takes $2^{\|U_h\| \cdot k} = O(1)$ rounds of the $Q_k$-hypothesis game and $O(1)$ additional bits of memory (we will free this memory once we compute $M_{h+1}$ so that we can use it again in the next iteration). After that, the $(h + 1)$st iteration is finished. Let us observe that for any $z$ which is compatible with the Nature’s responses after $h + 1$ iterations it holds that $(p_z, g(z)) \in \{(p_1, c_1), \ldots, (p_k, c_k)\}$, i.e,

$$(p_z, g(z)) \in \{(p_1, c_1), \ldots, (p_k, c_k)\} \text{ for all } z \in \mathcal{Z}_{h+1}. \quad (6.1)$$

After that, Learner updates $M_h$. He only needs to know $(p_1, c_1), \ldots, (p_k, c_k)$ (they can be extracted from the content of the working tape) and the light form of $\pi$. Namely, Learner determines $M_{h+1}[d_1, \ldots, d_k]$ for $(d_1, \ldots, d_k) \in [k]^k$ as follows. Consider the node $v = M_h[c_{d_1}, \ldots, c_{d_k}]$. If $v$ is a terminal, then set $M_{h+1}[d_1, \ldots, d_k] = v$. Otherwise, let $i \in [k]$ be the index of the party communicating at $v$. Look at $p_{d_i}$, it is a function from $U_h$ to $\{0, 1\}$. Define $r = p_{d_i}(v)$. Among two edges, starting at $v$, choose one which is labeled by $r$. Descend along this edge from $v$ and let the resulting successor of $v$ be $M_{h+1}[d_1, \ldots, d_k]$.

Obviously, computing $M_{h+1}$ takes time polynomial in size($\pi$). To show that Invariant 6.3 is maintained, we have to show that (a) $M_{h+1}$ is $(h + 1)$-low and (b) $M_{h+1}$ is complete for $\mathcal{Z}_{h+1}$.

The first part, (a), holds because each $M_{h+1}[d_1, \ldots, d_k]$ is either a terminal or a successor of a node of depth at least $h$. For (b) we define the following function:

$$g': \mathcal{Z}_{h+1} \rightarrow [k], \quad g'(z) = i, \text{ where } i \text{ is such that } (p_z, g(z)) = (p_i, c_i).$$

By (6.1) this definition is correct. We will show that $g'$ establishes completeness of $M_{h+1}$ for $\mathcal{Z}_{h+1}$.

For that, take any $z \in \mathcal{Z}_{h+1}$, $i \in [k]$ and $(d_1, \ldots, d_k) \in [k]^k$ such that $d_i = g'(z)$. We shall show that $z$ is $i$-compatible with a node $M_{h+1}[d_1, \ldots, d_k]$. By definition of $g'$, we have that $g(z) = c_{d_i}$. Recall that the function $g$ establishes completeness of $M_h$ for $\mathcal{Z}_h$. This means that $z$ is $i$-compatible with $v = M[c_{d_1}, \ldots, c_{d_k}]$. If $v$ is a terminal, then $M_{h+1}[d_1, \ldots, d_k] = v$ and there is nothing left to prove.

Otherwise, $v \in V \setminus T$. Let $j$ be the index of the party communicating at $v$. By definition, $M_{h+1}[d_1, \ldots, d_k]$ is a successor of $v$. If $j \neq i$, then any successor of $v$ is $i$-compatible with $z$ (because the $j$th party communicates at $v$, not the $i$th one). Finally, assume that $j = i$. The node $M_{h+1}[d_1, \ldots, d_k]$ is obtained from $v$ by descending along the edge which is labeled by $r = p_{d_i}(v)$. Hence, to show that $z$ is $i$-compatible with $M_{h+1}[d_1, \ldots, d_k]$, we should verify that the $i$th party transmits $r$ at $v$ on input $z$. Recall that $g'(z) = d_i$, which by definition of $g'$ means that $p_z = p_{d_i}$.
That is, \( p_{d_i} \) is the communication profile of \( z \) with respect to \( U_h \). In particular, \( r = p_{d_i}(v) = p_z(v) \) is the bit transmitted by the \( i \)th party on input \( z \) at \( v \), as required. \( \square \)

By the same argument, one can obtain an analog of the previous theorem for the \( R_k \) case.

**Theorem 6.5.** For every constant \( k \geq 2 \) there exists a polynomial-time algorithm \( A \) such that the following holds. Assume that \( f \in R_k \) and \( \pi \) is a dag-like protocol which strongly computes the \( R_k \)-communication game for \( f \). Then, given the light form of \( \pi \), the algorithm \( A \) outputs an \( R_k \)-circuit \( C \leq f \) such that depth(\( C \)) is linear in depth(\( \pi \)) and size(\( C \)) is polynomial in size(\( \pi \)).

7 Derivation of Theorems 1.1 and 1.3

In this section, we obtain Theorems 1.1 and 1.3 by devising protocols strongly computing the corresponding \( Q_k \)-communication games. Unfortunately, establishing strong computability requires diving into straightforward but tedious technical details, even for simple protocols.

**Alternative proof of Theorem 1.1.** We will show that there exists a protocol \( \pi \) of depth \( O(\log n) \) with a polynomial-time computable light form, strongly computing the \( Q_2 \)-communication game for \( \text{MAJ}_{2n+1} \). By Theorem 6.1, this means that there is a polynomial-time computable \( \text{MAJ}_3 \)-formula \( F \leq \text{MAJ}_{2n+1} \) of depth \( O(\log n) \). From the self-duality of \( \text{MAJ}_{2n+1} \) and \( \text{MAJ}_3 \) it follows that \( F \) computes \( \text{MAJ}_{2n+1} \).

Due to the AKS sorting network, there exists a polynomial-time computable monotone formula \( F' \) of depth \( O(\log n) \) for \( \text{MAJ}_{2n+1} \). Consider the following communication protocol \( \pi \). The tree of \( \pi \) coincides with the tree of \( F' \). Inputs to \( F' \) will be leaves of \( \pi \). In a leaf containing an input variable \( x_i \), the output of the protocol \( \pi \) is \( i \). Remaining nodes of \( \pi \) are \( \land \)-gates and \( \lor \)-gates. The first party communicates in \( \land \)-gates, while the second party communicates in \( \lor \)-gates. Let us now define how the parties communicate.

Fix an \( \land \)-gate \( g \) (which belongs to the first party). Let \( g_0, g_1 \) be gates that are fed to \( g \), i.e., \( g = g_0 \land g_1 \). There are two edges, starting at \( g \), one leads to \( g_0 \) (and is labeled by 0) and the other leads to \( g_1 \) (and is labeled by 1). Take an input \( a \in \text{MAJ}^{-1}_{2n+1}(0) \) to the first party. Having input \( a \), the first party transmits the bit \( r = \min\{c \in \{0, 1\} \mid g(c)(a) = 0\} \) at the gate \( g \). If the minimum is over the empty set, we set \( r = 0 \).

Take now an \( \lor \)-gate \( h \) (belonging to the second party). Similarly, there are two edges starting at \( h \), one leads to a gate \( h_0 \) (and is labeled by 0) and the other leads to a gate \( h_1 \) (and is labeled by 1). Take an input \( b \in \text{MAJ}^1_{2n+1}(0) \) to the second party. Having input \( b \), the second party transmits the bit \( r = \min\{c \in \{0, 1\} \mid h(c)(b) = 1\} \) at the gate \( h \). If the minimum is over the empty set, then we set \( r = 0 \). Here \( \neg \) denotes the bitwise negation. Description of the protocol \( \pi \) is finished.

Clearly, the protocol \( \pi \) is of depth \( O(\log n) \) and its light form is polynomial-time computable. It remains to argue that the protocol strongly computes the \( Q_2 \)-communication game for \( \text{MAJ}_{2n+1} \). Nodes of the protocol may be identified with gates of \( F' \). Consider any path \( p = (e_1, \ldots, e_m) \) in the protocol \( \pi \) which starts in the output gate \( g^0 \). Assume that \( e_j \) is an edge from \( g^{j-1} \) to \( g^j \). We shall show the following: if \( a \in \text{MAJ}^{-1}_{2n+1}(0) \) is 1-compatible with \( p \), then \( g^0(a) = g^1(a) = \ldots = g^m(a) = 0 \). Indeed, \( g^0(a) = 0 \) holds because \( F' \) computes \( \text{MAJ}_{2n+1} \). Now,
assume that \( g^j(a) = 0 \) is already proved. If \( g^j \) is an \( \lor \) gate, then \( g^{j+1}(a) = 0 \) just because \( g^{j+1} \) feeds to \( g^j \). Otherwise, \( g^j \) is an \( \land \) gate which therefore belongs to the first party. Let \( r \in \{0, 1\} \) be the label of the edge \( e_{i+1} \). Note that \( g^{j+1} = g^r \), where \( g^r_0, g^r_1 \) are two gates which are fed to \( g^j \).

Since \( a \) is 1-compatible with \( p \), it holds that \( r \) coincides with the bit that the first party transmits at \( g^j \) on input \( a \), i.e., with \( \min\{c \in \{0, 1\} \mid g^r_c(a) = 0\} \). The set over which the minimum is taken is non-empty, because \( g^j(a) = 0 \). In particular, \( r \) belongs to this set, which means that \( g^{j+1}(a) = g^r_j(a) = 0 \), as required.

Similarly, one can verify that if \( b \in \text{MAJ}_{2n+1}^{-1}(0) \) is 2-compatible with \( p \), then \( g^0(\neg b) = g^1(\neg b) = \ldots = g^m(\neg b) = 0 \). Overall, we get that if a leaf \( l \) contains a variable \( x_i \) and \( l \) is 1-compatible with \( a \) then \( a_i = 0 \), and if \( l \) is 2-compatible with \( b \) then \( \neg b_i = 1 \).

Hence the protocol strongly computes the \( Q_2 \)-communication game for \( \text{MAJ}_{2n+1} \).

**Proof of Theorem 1.3.** We will use the same protocol as in the proof of Corollary 5.9. This time, however, we have to define its light form more explicitly. We will obtain polynomial-size dag-like protocol of depth \( O(\log^2 n) \) with a polynomial-time computable light form, which strongly computes the \( Q_k \)-communication game for \( \text{THR}_{n+1}^{kn+1} \). By Theorem 6.1, this means that there is a polynomial-time computable \( Q_k \)-circuit \( C \leq \text{THR}_{n+1}^{kn+1} \) of depth \( O(\log^2 n) \). By an argument from Corollary 5.9, the circuit \( C \) coincides with \( \text{THR}_{n+1}^{kn+1} \).

We will use the same tree \( T \) as in the proof of Corollary 5.9. That is, \( T \) is a tree of depth \( O(\log n) \) with \( kn + 1 \) leaves. We identify its leaves with elements of \([kn+1]\). Every node \( v \) of \( T \) is associated with the set \( T_v \subseteq \{kn+1\} \) of leaves that are descendants of \( v \). We also use a notation \( \text{supp}(x) = \{i \in \{kn+1\} \mid x_i = 1\} \) for \( x \in \{0, 1\}^{kn+1} \).

Let us specify the underlying dag \( G \) of our protocol \( \pi \). For a node \( v \) of \( T \), let \( S_v \) be the set of all tuples \((s_1, s_2, \ldots, s_k) \in \{0, 1, \ldots, kn+1\}^k \) such that \( s_1 + s_2 + \ldots + s_k < |T_v| \). For every node \( v \) of \( T \) and for every \((s_1, s_2, \ldots, s_k) \in S_v \), the dag \( G \) will contain a node identified with a tuple \((v, s_1, s_2, \ldots, s_k) \). These nodes of \( G \) will be called main nodes (there will be some other nodes too). Observe that the number of main nodes is polynomial in \( n \). The starting node of \( G \) will be \((r, n, \ldots, n) \), where \( r \) is the root of \( T \). Note that if \( l \) is a leaf of \( T \), then \(|T_l| = 1\). Hence, the only main node having \( l \) as the first coordinate is \((l, 0, 0, \ldots, 0) \). The set of terminals of \( \pi \) will coincide with the set of all main nodes of the form \((l, 0, 0, \ldots, 0) \), where \( l \) is a leaf of \( T \). The output of \( \pi \) in \((l, 0, 0, \ldots, 0) \) is \( l \).

The communication in \( \pi \) is arranged as follows. First, we assume for simplicity that every party has a vector with exactly \( n \) ones (if there are less than \( n \) ones, one can add a necessary amount of ones to the input). The communication proceeds in \( O(\log n) \) phases. In the beginning of each phase, the parties belong to some main node \((v, s_1, \ldots, s_k) \). Then the first party sends two non-negative integers that sum up to \( s_1 \). After that, the second party sends two non-negative integers that sum up to \( s_2 \), and so on. More specifically, if the input to the \( i \)th party is \( x \in \{0, 1\}^{kn+1} \) and \(|T_v \cap \text{supp}(x)| \leq s_i \), then the \( i \)th party sends \(|T_{v_0} \cap \text{supp}(x)| \) and \(|T_{v_1} \cap \text{supp}(x)| \), where \( v_0 \) and \( v_1 \) are two successors of \( v \). Otherwise, it sends any two numbers that sum up to \( s_i \).

When all the numbers are sent, the parties move to some other main node. More specifically, let \( a_i \) and \( b_i \) be numbers sent by the \( i \)th party. If \( a_1 + \ldots + a_k < |T_{v_0}| \), the parties move the main node \((v_0, a_1, \ldots, a_k) \). Now, assume that \( a_1 + \ldots + a_k \geq |T_{v_0}| \). We claim that in this case we have
$b_1 + \ldots + b_k < |T_v|$. This is because by definition $(a_1 + \ldots + a_k) + (b_1 + \ldots + b_k) = s_1 + \ldots + s_k < |T_v| = |T_{v_0}| + |T_{v_1}|$. So if $a_1 + \ldots + a_k \geq |T_{v_0}|$, the parties move to the main node $(v_1, b_1, \ldots, b_k)$.

Observe that if the input to the $i$th party, $x$, has exactly $n$ ones, then throughout any execution of the protocol we have $|T_v \cap \text{supp}(x)| = s_i$, where $(v, s_1, \ldots, s_k)$ is our current main node. In other words, if a vector $x \in \{0, 1\}^{kn+1}$ with $n$ ones is $i$-compatible with a main node $(v, s_1, \ldots, s_k)$, then $|T_v \cap \text{supp}(x)| = s_i$. This implies that $\pi$ strongly computes the $Q_k$-communication game for $\text{THR}^{kn+1}_{n+1}$. Indeed, if a terminal $(l, 0, \ldots, 0)$ is $i$-compatible with $x$, then $|l \cap \supp(x)| = 0$, that is, $x_i = 0$. In other words, the output of $\pi$ in $(l, 0, \ldots, 0)$ is a correct answer for $x$, as required.

Overall, the light form of $\pi$ looks as follows. It consists of polynomially many main nodes that are arranged in a tree of depth $O(\log n)$. Each main node has a protocol of depth $O(\log n)$ attached to it. The leaves of this protocol are merged with some main nodes on the next level of $T$. Thus, $\pi$ is of depth $O(\log^2 n)$ and polynomial size, and its light form is polynomial-time computable.

8 Direct proof of Theorem 1.1

In this section we distill from our argument a direct proof of Theorem 1.1. We show that there exists a deterministic polynomial-time algorithm performing the following transformation

- **Input:** a monotone formula $F$ of depth $d$ computing $\text{MAJ}_{2n+1}$;
- **Output:** a $\text{MAJ}_3$-formula $\Phi$ of depth $d + O(\log n)$ computing $\text{MAJ}_{2n+1}$.

The existence of such an algorithm implies Theorem 1.1. Indeed, take the AKS sorting network and extract from it a polynomial-time computable monotone formula of depth $O(\log n)$ computing $\text{MAJ}_{2n+1}$. Then just plug $F$ into the transformation above. So it only remains to explain how to perform this transformation in polynomial time.

In the proof by $\{0, 1\}^{2n+1}_{\leq n}$ we denote the set of all $(2n+1)$-bit vectors with at most $n$ ones. This is also the set of vectors where $\text{MAJ}_{2n+1}$ equals 0. For $x \in \{0, 1\}^{2n+1}$ we denote by $\overline{x}$ the bitwise negation of $x$.

The following observation simplifies our task.

**Observation 8.1.** Assume that $\Phi$ is a $\text{MAJ}_3$-formula and

$$\Phi(x) = 0 \text{ for any } x \in \{0, 1\}^{2n+1}_{\leq n}.$$

Then $\Phi$ computes $\text{MAJ}_{2n+1}$.

**Proof.** It is already given that $\Phi$ equals 0 everywhere, where $\text{MAJ}_{2n+1}$ equals 0. It remains to show that $\Phi$ equals 1 everywhere, where $\text{MAJ}_{2n+1}$ equals 1. For that, we take any $x \in \{0, 1\}^{2n+1}$ with at least $n+1$ ones and show that $\Phi(x) = 1$. Formula $\Phi$ is constructed from self-dual gates and hence computes a self-dual function (recall that a Boolean function is self-dual if it takes opposite values in opposite vertices of the Boolean cube). This means that $\Phi(x) = \overline{\Phi}(\overline{x})$. Finally, notice that $\Phi(\overline{x}) = 0$ because $\overline{x} \in \{0, 1\}^{2n+1}_{\leq n}$. $\square$
The construction can naturally be split into two independent steps.

- **Step 1.** For any two distinct $i, j \in [2n + 1]$ construct from $F$ a $\text{MAJ}_3$-formula $\Phi_{i,j}$ of depth $d$ (i.e., of the same depth as $F$) such that
  \[ \Phi_{i,j}(x) = 0 \text{ for any } x \in \{0, 1\}^{2n+1}_{\leq n} \text{ such that } x_i + x_j = 1. \]

- **Step 2.** Assemble from the formulas $\Phi_{i,j}$ a $\text{MAJ}_3$-formula $\Phi$ of depth $d + O(\log n)$ satisfying
  \[ \Phi(x) = 0 \text{ for all } x \in \{0, 1\}^{2n+1}_{\leq n}. \]

By Observation 8.1 the formula $\Phi$ from the step 2 will compute $\text{MAJ}_{2n+1}$.

**Step 1.** We obtain $\Phi_{i,j}$ from $F$ in a way described in Figure 2. We only have to show that for all $x \in \{0, 1\}^{2n+1}_{\leq n}$ with $x_i + x_j = 1$ we have $\Phi_{i,j}(x) = 0$. The argument is different for the following two cases.

- **Case 1:** $x_i = 0$ and $x_j = 1$.
- **Case 2:** $x_i = 1$ and $x_j = 0$. 

---

**Figure 2:** Transforming $F$ into $\Phi_{i,j}$. 

for all $x \in \{0, 1\}^{2n+1}_{\leq n}$ with $x_i + x_j = 1$ we have $\Phi_{i,j}(x) = 0$. The argument is different for the following two cases.
Both cases rely on the following observation. Notice that \( \Phi_{i,j}(x) = \text{MAJ}_{2n+1}(x) \) for all \( x \in \{0,1\}^{2n+1} \) with \( x_i = 0, x_j = 1 \). This is because when we plug in \( x_i = 0, x_j = 1 \) into \( \Phi_{i,j} \), we obtain a formula which is equivalent to \( F \). Indeed, every \( \text{MAJ}_3 \)-gate in \( \Phi_{i,j} \) that were obtained from an \( \wedge \)-gate of \( F \) turns back into an \( \wedge \)-gate. Similarly, every \( \text{MAJ}_3 \)-gate in \( \Phi_{i,j} \) that were obtained from an \( \vee \)-gate of \( F \) turns back into an \( \vee \)-gate. To see this, note that \( \text{MAJ}_3(g_1, g_2, 0) = g_1 \wedge g_2 \) and \( \text{MAJ}_3(g_1, g_2, 1) = g_1 \vee g_2 \).

**Case 1.** This is an immediate consequence of the above observation. Formula \( \Phi_{i,j} \) coincides with \( \text{MAJ}_{2n+1} \) every time \( x_i = 0, x_j = 1 \), and for \( x \in \{0,1\}^{2n+1} \) we have \( \text{MAJ}_{2n+1}(x) = 0 \).

**Case 2.** Here we use a self-duality argument. Consider the bitwise negation of \( x \). Since \( \neg x \) has at least \( n + 1 \) ones, we have \( \text{MAJ}_{2n+1}(\neg x) = 1 \). Next, since \( (\neg x)_i = 0, (\neg x)_j = 1 \), we have \( \Phi_{i,j}(\neg x) = \text{MAJ}_{2n+1}(\neg x) = 1 \) by our observation. Finally, due to self-duality, \( \Phi_{i,j}(x) = \neg \Phi_{i,j}(\neg x) = 0 \), as required.

**Step 2.** We show that for any \( S \subseteq [2n+1], |S| \geq 2 \) one can construct (in deterministic polynomial time) a \( \text{MAJ}_3 \)-formula \( \Phi_S \) of depth at most \( d + 1 + \log_{9/8}(|S|) \) such that:

\[
\Phi_S(x) = 0 \text{ for all } x \in \{0,1\}^{2n+1} \text{ such that } x_i = 0 \text{ for some } i \in S.
\]

By setting \( \Phi = \Phi_{[2n+1]} \) we obtain a formula which is 0 everywhere on \( \{0,1\}^{2n+1} \), as required. Indeed, every \( x \in \{0,1\}^{2n+1} \) has a 0-coordinate in \([2n+1]\).

The construction is recursive. Assume first that \( |S| \geq 3 \). Partition \( S \) into 3 disjoint subsets \( S_1, S_2, S_3 \) of sizes \( |S|/3, |S|/3 \) and \( |S| - 2|S|/3 \). Construct recursively \( \Phi_{S_1 \cup S_2}, \Phi_{S_1 \cup S_3}, \Phi_{S_2 \cup S_3} \) and then set

\[
\Phi_S = \text{MAJ}_3(\Phi_{S_1 \cup S_2}, \Phi_{S_1 \cup S_3}, \Phi_{S_2 \cup S_3}).
\]

If \( |S| = 2 \) and \( S = \{i, j\} \), set

\[
\Phi_{\{i,j\}} = \text{MAJ}_3(\Phi_{i,j}, x_i, x_j),
\]

where \( \Phi_{i,j} \) is from the previous step. Description of the construction is finished. It remains to explain why this construction is correct, why the depth of \( \Phi_S \) is at most \( d + 1 + \log_{9/8}(|S|) \) and why the construction takes polynomial time.

- A recursive call is always for sets of smaller size. More specifically, it holds that:

\[
|S_1 \cup S_2|, |S_1 \cup S_3|, |S_2 \cup S_3| \leq \frac{8}{9} \cdot |S|.
\]

Indeed, the sizes of \( S_1 \cup S_2, S_1 \cup S_3, S_2 \cup S_3 \) do not exceed \( |S| - |S|/3 \leq |S| - |S|/3 + 2/3 = 1/3 \cdot (|S| + 1) \leq 2/3 \cdot (|S| + |S|/3) = 8/9 \cdot |S| \).

- We now show, by induction on \( |S| \), that \( \Phi_S(x) = 0 \) for all \( x \in \{0,1\}^{2n+1} \) that have a 0-coordinate in \( S \). First, consider the case \( S = \{i, j\} \). If there are exactly one 0-coordinate among \( i, j \), then by definition \( \Phi_{i,j}(x) = 0 \) and hence \( \Phi_{\{i,j\}}(x) = \text{MAJ}_3(\Phi_{i,j}(x), x_i, x_j) = \text{MAJ}_3(0, 0, 1) = 0 \). If both \( x_i = 0 \) and \( x_j = 0 \), then \( \Phi_{\{i,j\}}(x) = \text{MAJ}_3(\Phi_{i,j}(x), 0, 0) = 0 \).

Now, consider the case \( |S| \geq 3 \). A 0-coordinate of \( x \) lying in \( S \) lies also in exactly 2 sets out of \( S_1 \cup S_2, S_1 \cup S_3, S_2 \cup S_3 \). Hence, by the induction hypothesis, among
\(\Phi_{S_1 \cup S_2}(x), \Phi_{S_1 \cup S_3}(x), \Phi_{S_2 \cup S_3}(x)\) there are at least 2 zeroes. This means that \(\Phi_S(x) = \text{MAJ}_3(\Phi_{S_1 \cup S_2}(x), \Phi_{S_1 \cup S_3}(x), \Phi_{S_2 \cup S_3}(x)) = 0\).

- Again, by induction on \(|S|\) one can show that

\[
\text{depth}(S) \leq d + 1 + \log_{9/8}(|S|),
\]

For \(S = \{i, j\}\) the depth of \(\Phi_{\{i,j\}}\) is \(\text{depth}(\Phi_{\{i,j\}}) + 1 = d + 1 \leq d + 1 + \log_{5/4}(|S|)\). For \(|S| \geq 3\) assume that the claim is proved for \(\Phi_{S_1 \cup S_2}, \Phi_{S_1 \cup S_3}, \Phi_{S_2 \cup S_3}\). Then

\[
\text{depth}(\Phi_S) = 1 + \max\{\text{depth}(\Phi_{S_1 \cup S_2}), \text{depth}(\Phi_{S_1 \cup S_3}), \text{depth}(\Phi_{S_2 \cup S_3})\}
\leq 1 + d + 1 + \log_{9/8}\left(\frac{8}{9} \cdot |S|\right)
= d + 1 + \log_{9/8}(|S|).
\]

In the second line, we use the induction hypothesis (8.1).

- Similarly, the tree of recursive calls for \(\Phi_S\) has depth at most \(\log_{9/8}(|S|)\) and hence polynomial size. Therefore, the whole construction takes polynomial time.

9 Open problems

- Can the \(Q_k\)-communication game for \(\text{THR}^{\frac{k+1}{n+1}}\) be solved in \(o(\log^2 n)\) bits of communication for \(k \geq 3\)? Equivalently, can \(\text{THR}^{\frac{k+1}{n+1}}\) be computed by a \(Q_k\)-circuit of depth \(o(\log^2 n)\)? Or at least by an \(R_k\)-circuit of depth \(o(\log^2 n)\)?

- Are there any other interesting functions in \(Q_k\) and \(R_k\) which can be analyzed with our technique?

References


Alexander Kozachinskiy and Vladimir Podolskii


MULTIPARTY KARCHMER–WIGDERSON GAMES AND THRESHOLD CIRCUITS

AUTHORS

Alexander Kozachinskiy
Scientific Researcher
Department of Mathematical Logic
Steklov Mathematical Institute
Moscow, Russia
kozlach@mail.ru
https://kozlachinskiy.github.io/

Vladimir Podolskii
Leading Scientific Researcher
Steklov Mathematical Institute and
HSE University
Moscow, Russia
podolskii.vv@gmail.com
https://homepage.mi-ras.ru/~podolskii/

ABOUT THE AUTHORS

Alexander Kozachinskiy grew up in Moscow. For about 10 years, he was at the Moscow State University, first as an undergraduate, and then as a Ph. D. student advised by Nikolay Vereshchagin. He defended his thesis titled “Comparison of Communication, Information and Decision Tree Complexities” in 2019. Currently he works as a research fellow at the Steklov Mathematical Institute. He likes to invent algorithms in areas like Circuit Complexity, where people are mostly interested in lower bounds. Besides complexity, he works on Graph Games.

Vladimir Podolskii is a leading scientific researcher at the Steklov Mathematical Institute. He also holds a part-time position at HSE University. His main research interests are computational complexity, tropical algebra and logical foundations of computer science. He graduated from Moscow State University in 2009, advised by Nikolay Vereshchagin (and earlier by Alexander Razborov).