Introduction to Complexity Theory*
Lecture 23: The FORK Game

Guest Lecturer: Ran Raz
Notes taken by Dana Fisman and Nir Piterman

July 1, 1999

Summary: We analyze the game FORK that was introduced in the previous lecture. We give tight lower and upper bounds on the communication needed in a protocol solving FORK. This completes the proof of the lower bound on the depth of monotone circuits computing the function st-Connectivity.

1 Introduction

We have seen in the previous lecture the connection between circuit depth and communication complexity: We compared the depth of a circuit computing a function f to the number of bits transferred between two players playing a communication game G_f. We saw that given a communication protocol solving G_f using communication of c-bits, we can construc t a circuit computing f whose depth is c. On the other hand, given a c-depth circuit computing f we can plan a communication protocol for G_f which uses c bits. Thus, we established that the best communication protocol for solving G_f uses t he same amount of communication bits as the depth of the best circuit computing f. tion protocol. The plan was to use communication complexity to prove upper and lower bounds on the depths of circuits. Failing to reach satisfactory results using this model, the weaker model of monotone functions and *monotone* circuits was introduced. Since a mono tone circuit is in particular a circuit but not the other way around, proving lower bounds on monotone circuits does not prove the same lower bounds in the unrestricted model. All the same, our goal was to show a (tight) lower bound for the depth of monot one circuits computing st-Connectivity. The plan was to achieve the result by conducting a series of reductions. The first used the connection between circuit depth and communication complexity to reduce the question to a communication protocol. Then, various reductions between several kinds of games led us to the FORK game (see Definition 1). Based on a lower bound for the communication needed in FORK, the lower bound for st-Connectivity of $\Omega(log^2 n)$ was proven. The purpose of this lecture is to prove the lower bound of the FORK game. We will give a complete analysis of the FORK game showing that the communication between the two players is $\Theta(log_2 w \cdot log_2 l)$, thus supplying the missing link in the proof of the lower bound on the depth of st-Connectivity.

* Lecture Notes for a course given by Oded Goldreich at the Weizmann Institute of Science, Israel.

2 The FORK game

FORK is a game between two players. Each player gets a path from a predefined set of possible paths. Both paths start at the same point but have different end points. Hence, at least one fork point in which the two paths separate, must exist. The players' goal is to find such a fork point. More formally, we recall the definition of the FORK game given in previous lecture.

Definition 1 (FORK): *Given $n = l \cdot w$ vertices and three special vertices s, t_1, and t_2, where the n vertices are divided into l layers $l_1, \ldots, l_l$, and each layer contains w vertices:*

- Player I *gets a sequence of vertices $(x_1, x_2, \ldots, x_l)$, where for all $1 \leq i \leq l$: $x_i \in l_i$. For simplicity of notation we assume that* Player I *is given two more coordinates: $x_0 = s$, $x_{l+1} = t_1$.*
- Player II *gets a sequence of vertices $(y_1, y_2, \ldots, y_l)$, where for all $1 \leq i \leq l$: $y_i \in l_i$. Again we consider $y_0 = s$, $y_{l+1} = t_2$.*
- *Their goal is to find a coordinate i such that $x_i = y_i$ and $x_{i+1} \neq y_{i+1}$.*

In order to stress the fact that the inputs are elements of $[w]^l$, where $[w] = \{1, ..., w\}$, we slightly modified the definition, excluding the constant points s, t_1 and t_2 from the input sequences given to the players.

The following theorem states the main result of this lecture, giving the bounds on the communication complexity of solving the FORK game.

Theorem 2 *The communication complexity of the* FORK *game is $\Theta(log_2 w \cdot log_2 l)$.*

We first show an upper bound on the communication complexity of the game. The upper bound is given here only for the completeness of the discussion concerning the FORK problem.

3 An upper bound for the FORK game

The proof given here is very similar to the proof of the upper bound for the KW game (see previous lecture). We basically perform a binary search on the path to find the FORK point.

Proposition 3.1 *The communication complexity of the* FORK *game is $O(log_2 w \cdot log_2 l)$.*

Proof: For the sake of simplicity, the following notation is introduced. We denote $F_{a,b}$ as the FORK game where the inputs are of the form $\overline{x} = (x_a, x_{a+1}, ..., x_b)$ and $\overline{y} = (y_a, y_{a+1}, ..., y_b)$. Like in the general FORK gam e, we consider $\overline{x}$ and $\overline{y}$ as having two more coordinates. An $(a-1)$-coordinate such that $x_{a-1} = y_{a-1} = s$ is the origin point of the paths. A $(b+1)$-coordinate such that $x_{b+1} = t_1$ and $y_{b+1} = t_2$ are the endpoints of the paths.

First notice that if the length of the paths is only one, i.e., $a = b$, then the problem can be solved using $log_2 w$ bits, obeying the following protocol:

- Player I sends its input (this requires $log_2 w$ bits).
- Player II replies with 1 if he has the same coordinate and with 0 otherwise (one bit).

If they have the same coordinate, the fork point is found in that coordinate and then the paths separate to t_1 and t_2. Otherwise the fork is the point of origin $s = x_{a-1} = y_{a-1}$.

If the length of the paths is larger than 1, i.e., $a < b$, then the problem can be reduced to half using $log_2 w$ bits. (For simplicity, we assume that $\frac{a+b}{2}$ is an integer):

- Player I sends its middle layer node: $x_{\frac{(a+b)}{2}}$ (this requires $log_2 w$ bits).
- Player II checks if they have the same middle node: $y_{\frac{a+b}{2}} = x_{\frac{a+b}{2}}$.

  If so he sends 1 otherwise he sends 0 (one bit).

If the players have the same middle point (i.e., Player I sent 1), then there has to be a fork point between the middle and the end. Thus the game is reduced to $F_{\frac{a+b}{2}+1,b}$. On the other hand, if the middle points differ, then there has to be a fork point between the start and the middle. Thus the game is reduced to $F_{a,\frac{a+b}{2}-1}$. Note that there is no point in including the middle layer itself in the range of the reduced game. The reason being that in the first case, the mutual point in layer $\frac{a+b}{2}$ is actually the new origin point, while in the second case, the two points in layer $\frac{a+b}{2}$ are actually the new end points.

Therefore FORK (i.e. $F_{1,l}$ using this notation) is solved in $log_2 l$ iterations of the protocol, requiring transmission of $O(log_2 w \cdot log_2 l)$ bits. ■

Our goal now is to show that this upper bound is tight.

4 A lower bound for the FORK game

In order to show the lower bound we consider games that work only for a subset of the possible inputs. We perform some inductive process in which we establish the connection between games with different sets. Two kinds of transformations are considered:

1. Given a protocol that works for some set, we devise a protocol that works for smaller density sets with one less bit of communication.
2. Given a protocol that works for some set, we convert it into a protocol that works for sets of higher density but shorter paths using the same amount of communication.

When adapting the given protocol into a new one, some heavy computations are involved. However, recall that the only parameter considered during the application of the protocol is the number of bits transmitted. Thus, any computation done in the preparati on of the protocol and any computation done locally by either side is not taken into account.

4.1 Definitions

We first define a subgame of FORK that works only on a subset of the possible inputs. Given a subset $S \subseteq [w]^l$ we consider the game FORK^S where Player I gets an input $\overline{x} \in S$ and Player II gets an input $\overline{erliney} \in S$ and they have the same goal of finding the fork point between the two paths.

We will work only with subsets of $[w]^j$, where the power j (the length of the path) will change from time to time. We call w the width of the game. Throughout the discussion the width w of the game will be constant and we frequently ignore i t.

As explained above, the density of the set will be an important parameter in the transformations:

Definition 3 (density): *The* density *of a set $S \subseteq [w]^l$ is defined as $\alpha(S) = \frac{|S|}{w^l}$.*

Given a protocol solving a partial FORK game, we ask ourselves what is the density of the sets that this protocol works on. We are interested in the minimal protocol that will work for some set of density α:

Definition 4 (*(α, l) − protocol*): *A communication protocol will be called an* (α, l)-protocol *if it works for some set S of paths of length l with density $\alpha(S) = \alpha$.*

Definition 5 ($CC(\alpha, l)$): *Let $CC(\alpha, l)$ denote the smallest communication complexity of an $(\alpha, l)-$ protocol.*

Using this terminology, since there is just one set of density 1, a protocol for FORK is just a $(1, l) - protocol$ and so we are interested in $CC(1, l)$.

4.2 Reducing the density

The following lemma enables the first kind of transformation discussed above. Given a protocol that works for a set of a certain density we adapt it to a protocol that works for a subset of half that density. The new protocol will work with one less bit o f communication than the original protocol.

Lemma 4.1 *If there exists an $(\alpha, l) - protocol$ which works with c bits and $c > 0$, then there is also an $(\frac{\alpha}{2}, l) - protocol$ that works with $c - 1$ bits.*

By the above lemma, the best protocol for an α density set will require at least one more bit than the best protocol for an $\frac{\alpha}{2}$ density set. Thus, we have

Corollary 6 *If $CC(\alpha, l)$ is non-zero then $CC(\alpha, l)$ is greater than $CC(\frac{\alpha}{2}, l)$ by at least one;*

$$CC(\alpha, l) \geq CC(\tfrac{\alpha}{2}, l) + 1$$

Proof: The proof can be generalized for any communication protocol. As explained in detail in Lecture 21, we use the fact that any communication protocol can be viewed as the two parties observing a matrix. The matrix' rows correspond to the possible inputs of Player I and its columns correspond to the possible inputs of Player II. The entries of the matrix are the desired results of the protocol. Passing one bit between the two players partitions this matrix into two rectangles (horizontally if the first bit is sent by Player I and vertically if it is sent by Player II).

Let P be a non-zero communication $(\alpha, l) - protocol$ for the set S. Assume without loss of generality that Player I sends the first bit in the protocol P. Let $S_0 \subseteq S$ be those inputs in S for which Player I sends 0 as the f irst bit, and similarly let $S_1 \subseteq S$ be those inputs in S for which Player I sends 1 as the first bit. Since $S_0 \cup S_1 = S$, either S_0 or S_1 contains at least half the inputs in S. Assuming that this is S_0 then $\alpha(S_0) \geq \frac{\alpha(S)}{2}$. Let P' be a protocol that works like P but without sending the first bit, while the players assume that the value of this bit is 0. Obviously P' works if Player I gets inputs from S_0 and Player II gets inputs fr om S. In particular it works when both inputs are from the subset S_0. We conclude that P' is an $(\frac{\alpha}{2}, l) - protocol$ that works with one bit less than P.
The case where S_1 is larger than S_0 is identical. ■

In order to apply Lemma 4.1 the protocol must cause at least one bit to be sent by some player. Therefore, we would like to identify the sets for which a protocol consumes some communication. We show that for large enough sets, whose density is more than $\frac{1}{w}$, any protocol indeed uses non-zero communication.

Lemma 4.2 *Any protocol for a set whose density is larger than $\frac{1}{w}$ requires communication of at least one bit.*

Corollary 7 *For every l and every $\alpha > \frac{1}{w}$, $CC(\alpha, l) > 0$.*

Proof: We will show that if P is a protocol that works with no communication at all and solves FORK^S then the density of S is at most $\frac{1}{w}$.

Since no information is passed between the players, any of the players must establish his result depending only on his input. It could be the case where both players get the same path $\overline{x} = \overline{y}$. In this case the only fork point is in the last layer $x_l = y_l$ (with $x_{l+1} = t_1 \neq t_2 = y_{l+1}$). So Player I must always give the last layer as the fork point. If some $\overline{y} \in S$ has a different point in the last layer (i.e. $y_l \neq x_l$), the protocol will give a wrong answer. Hence for all the paths in S the last point in the path has to be j for some j between 1 and w. So the set S is in fact a subset of $[w]^{l-1} \times \{j\}$ and therefore its density is not greater than $\frac{w^{l-1}}{w^l} = \frac{1}{w}$. ∎

Note that using these two lemmas we can get a lower bound of $\Omega(log_2 w)$ bits for FORK. As long as the density is greater than $\frac{1}{w}$, Lemma 4.2 guarantees that we can apply Lemma 4.1. Repeating Lemma 4.1 less than $log_2 w$ times we know that the density has not decreased beyond $\frac{1}{w}$ and so Lemma 4.1 can be applied again.

$$CC(1, l) \geq CC(\tfrac{1}{2}, l) + 1 \geq CC(\tfrac{1}{4}, l) + 2 \geq ... \geq CC(\tfrac{1}{w}, l) + log_2 w$$

Considering our aim is to use this bound in connection with boolean circuits, the bound $CC(1, l) = \Omega(log_2 w)$ is insignificant. This is the case, since in order to read an input of length w we must use a circuit of at least $log_2 w$ depth anyhow.

4.3 Reducing the length

In order to reach the desired bound we need to manipulate the length of the paths as well. The main tool will be the following 'amplification' lemma that allows us, using an $(\alpha, l) - protocol$, to construct another protocol that works on a set of short er paths (of length $\frac{l}{2}$) but whose density is larger than α.

Lemma 4.3 *Let $\alpha \geq \frac{12}{w}$. If there exists an $(\alpha, l) - protocol$ for FORK^S, that uses c bits of communication, then there is also an $(\frac{\sqrt{\alpha}}{2}, \frac{l}{2}) - protocol$ that uses the same number of bits.*

For α's in the range $\frac{12}{w} < \alpha < \frac{1}{4}$ using this lemma increases the density of the set. Since:

$$\frac{12}{w} < \alpha < \frac{1}{4} \implies \frac{\sqrt{\alpha}}{2} > \alpha$$

The proof of the lemma uses the following technical claim:

Claim 4.4 *Consider an $n \times n$ matrix of 0-1. Denote by α the fraction of one-entries in the matrix and by α_i the fraction of one-entries in the i^{th} row. We say that a row i is* dense *if $\alpha_i \geq \frac{\alpha}{2}$. One of the following two cases hold:*

1. *there is some row i with $\alpha_i \geq \sqrt{\frac{\alpha}{2}}$*
2. *the number of dense rows is at least $\sqrt{\frac{\alpha}{2}} \cdot n$*

Proof: Intuitively, the claim says that either one of the rows is 'very dense' or there are a lot of dense rows.

Suppose by contradiction that the two cases do not hold. Let us calculate the density of one-entries in the entire matrix. Since Case 2 does not hold there are less than $\sqrt{\frac{\alpha}{2}} \cdot n$ dense rows. Since

Case 1 does not hold each of them has less than $\sqrt{\frac{\alpha}{2}} \cdot n$ one-entries. Hence the fraction of one-entries in all the dense rows is less than $\sqrt{\frac{\alpha}{2}} \cdot \sqrt{\frac{\alpha}{2}} = \frac{\alpha}{2}$. Non-dense rows contain less than $\frac{\alpha}{2} \cdot n$ one-entries and there are at most n non-dense rows, hence the fraction of one-entries in the non-dense rows is less than $\frac{\alpha}{2}$. Thus, the total fraction of one-entries is less than $\frac{\alpha}{2} + \frac{\alpha}{2} = \alpha$, contradicting the assumed fraction of one-entries in the matrix. ■

Proof: (Of Lemma 4.3)
Given an $(\alpha, l) - protocol$ we would like to show an $(\frac{\sqrt{\alpha}}{2}, \frac{l}{2}) - protocol$.
Let P be an $(\alpha, l) - protocol$. Assume that it works for the set S of paths in $[w]^l$ whose density is $\alpha(S) = \alpha$. We view the paths in S as two concatenated paths of half the length. Given $(s_1, ..., s_l)$ a path in S we denote $(s_1, ..., s_l)$ by $a \circ b$ where $a = (s_1, ..., s_{\frac{l}{2}})$, $b = (s_{\frac{l}{2}+1}, ..., s_l)$.

For any $a \in [w]^{\frac{l}{2}}$ we denote by $Suffix(a)$ the set of possible suffixes b for a, that form a path in S:

$$Suffix(a) = \{b \in [w]^{\frac{l}{2}} \mid a \circ b \in S\}$$

Consider a matrix whose rows and columns correspond to paths in $[w]^{\frac{l}{2}}$. An entry of the matrix (a, b) is 1 if the path $a \circ b$ is in S and 0 otherwise. Thus, the fraction of one-entries in the matrix is α (the density of S). Applying Claim 4.4 to the matrix, we get that this matrix satisfies either (1) or (2). Either there exists a prefix of a path in S that has a lot of suffixes, or there exist many prefixes of paths in S that have quite a lot of suffixes. In the first case we use the set of suffixes as the new set for which we build a new protocol, while in the second case we use the set of 'heavy' prefixes as the new set. In both cases we adapt the protocol P to work for the new set of half length paths. Details follow:

1. In case there is a prefix of a path a with at least $\sqrt{\frac{\alpha}{2}} \cdot w^{\frac{l}{2}}$ suffixes, we let $S' = Suffix(a)$ be the set of suffixes of that prefix, and the new protocol P' will work as following:

   Player I gets an input x in S', he concatenates it to a forming the path $a \circ x$ in S. In a similar way Player II forms the path $a \circ y$. Now they have paths in S and can apply the protocol P to find the fork point. Since the paths coincide in their first halves, the fork point must be in the second half.

   Note that if the first coordinate in x and y is different, the fork in the whole path can be found in the last coordinate of a. This is the case where for S' the fork was found in the point of origin of the paths, s.

2. In case there are many (i.e. at least $\sqrt{\frac{\alpha}{2}} \cdot w^{\frac{l}{2}}$) prefixes of paths with at least $\frac{\alpha}{2}$ suffixes we take S' to be the set of all the prefixes of 'dense' paths; that is, $S' = \{x : |Suffix(x)| \geq \frac{\alpha}{2} \cdot w^{\frac{l}{2}}\}$. We have $|S'| \geq \sqrt{\frac{\alpha}{2}} \cdot w^{\frac{l}{2}}$. For each possible input x in S' we will try and build two suffixes $b_1(x)$ and $b_2(x)$ such that for any two inputs x and y the suffixes $b_1(x)$ and $b_2(y)$ will not coincide. In this case a fork found between $x \circ b_1(x)$ and $y \circ b_2(y)$ must be in the first half of the path, since the second half is ensured not to coincide in any point.

   We suggest a method for building the suffixes (for simplicity we assume that w is even):

   For each layer $\frac{l}{2} + 1, ..., l$ we color half the nodes in the layer in orange and the other half in purple. If for every x, the suffix $b_1(x)$ will be colored orange and the suffix $b_2(x)$ will be colored purple the goal will be fulfilled. We wi ll show that such a coloring exists by showing that the probability for such a coloring over all random colorings is positive.

Claim 4.5 *There exists a coloring of all nodes such that for most $a \in S'$ there are suffixes $b_1(a)$ and $b_2(a)$ such that:*

- $a \circ b_1(a) \in S$ *and* $a \circ b_2(a) \in S$
- *all nodes in* $b_1(a)$ *are colored orange*
- *all nodes in* $b_2(a)$ *are colored purple*

We propose the following way to color the vertices in layers $\frac{l}{2}+1, ..., l$:

- Choose randomly $\frac{w}{2}$ paths $r_1, ..., r_{\frac{w}{2}}$ in $[w]^{\frac{l}{2}}$ and color all the vertices appearing in them in orange.
- In layers where the paths covered less than $\frac{w}{2}$ of the vertices choose randomly orange vertices to achieve $\frac{w}{2}$ vertices colored in orange.
- Color the rest of the vertices in purple.

By symmetry it is apparent that this method for coloring produces the same distribution as uniformly choosing $\frac{w}{2}$ nodes out of the w possible nodes in each layer (coloring them in orange and the rest in purple).

We shall show that for any a in S' there is a high probability that two such suffixes (i.e., one colored orange and one colored purple) can be found. Define S'' as the set of all prefixes a in S' that have two such suffixes. The expected size of S'' will be close to the size of S'. Therefore exists a coloring that induces such a set S'', whose size is very close to the size of S'.

For a path $a \in S'$, since the density of suffixes of a is at least $\frac{\alpha}{2}$, the probability that for some i, the random path r_i (chosen in the above process) is not a suffix of a is at most $(1-\frac{\alpha}{2})$. Since these paths are chosen independently and $\alpha > \frac{12}{w}$, the probability that none of the $\frac{w}{2}$ paths is a suffix of a is at most 0.05. This is because

$$Prob[r_i \notin Suffix(a)] \leq (1-\frac{\alpha}{2})$$

$$Prob[\forall i,\ r_i \notin Suffix(a)] \leq (1-\frac{\alpha}{2})^{\frac{w}{2}} < (1-\frac{6}{w})^{\frac{w}{2}} \approx e^{-3} \approx 0.049$$

There is no difference between the probability that the first suffix (i.e. the one colored orange) does not exist and the probability that the second suffix (i.e. the one colored purple) does not exist. Hence, we can use union bound to determine that the probability of either $b_1(a)$ or $b_2(a)$ not to exist, is smaller than 0.1.

Therefore, the expected size of S'' is at least 0.9 of the size of S'. Formally:

Define for every element a in S' a random variable $X_a = \begin{cases} 1 & \text{if } a \text{ has two such suffixes} \\ 0 & \text{otherwise} \end{cases}$

Obviously $|S''| = \Sigma_{a\in S'} X_a$ hence:

$$E(|S''|) = E(\Sigma_{a\in S'} X_a) = \Sigma_{a\in S'} E(X_a) = |S'| \cdot E(X_a) \geq 0.9 \cdot |S'|$$

So we have a set S'' of density at least $0.9 \cdot \sqrt{\frac{\alpha}{2}} \cdot w^{\frac{l}{2}} > \frac{\sqrt{\alpha}}{2} \cdot w^{\frac{l}{2}}$ such that for any path a of length $\frac{l}{2}$ in S'' we have two suffixes $b_1(a)$ and $b_2(a)$ colored orange and purple, respectively.

Player I will get an input x in $[w]^{\frac{l}{2}}$ and concatenate $b_1(x)$ to it, creating $x \circ b_1(x)$, a path in S. Player II, getting y, will create in a similar way $y \circ b_2(y)$. The two players will run the protocol P and find the desired fork point.

Note that if the last coordinate in x and y is equal, the fork can be found there. This is legitimate in S'' because the end points of the paths t_1 and t_2 are different.

We have seen that in both cases we were able to find subsets of $[w]^{\frac{l}{2}}$ of density at least $\frac{\sqrt{\alpha}}{2}$ for which we adapted the protocol P. Therefore we have proven:

$$CC(\alpha, l) \geq CC(\tfrac{\sqrt{\alpha}}{2}, \tfrac{l}{2})$$

■

4.4 Applying the lemmas to get the lower bound

We show that any protocol for solving the FORK game uses at least $\Omega(log_2 w \cdot log_2 l)$ bits of communication by employing Lemmas 4.1 and 4.3. From Lemma 4.2 we know that during all the transformat ions, we do not find a zero-bit protocol. For the simplicity of calculations we assume that l is a power of 2.

We start by reducing FORK to any subset of density $\frac{2}{\sqrt{w}}$.

$$CC(1, l) \geq CC(\tfrac{2}{\sqrt{w}}, l)$$

Using Lemma 4.1 by a series of $\Omega(log_2 w)$ transformations (all allowed by Lemma 4.2) we get:

$$CC(\tfrac{2}{\sqrt{w}}, l) \geq CC(\tfrac{16}{w}, l) + \Omega(log_2 w)$$

Using Lemma 4.3, we have

$$CC(\frac{16}{w}, l) \geq CC(\frac{2}{\sqrt{w}}, \frac{l}{2})$$

and so

$$CC(\frac{2}{\sqrt{w}}, l) \geq CC(\frac{2}{\sqrt{w}}, \frac{l}{2}) + \omega(log_2 w)$$

Iterate the last two steps $log_2 l$ times, getting

$$CC(1, l) = \Omega(log_2 l \cdot log_2 w)$$

We conclude that the communication complexity of the FORK game is $\Theta(log_2 l \cdot log_2 w)$.