IAS/Park City Mathematics Series
Volume 3, 1993

Circuit and Communication Complexity

Ran Raz

LECTURE 7
Lower bound for directed st-Connectivity

In this lecture and in the next one, we will prove that the monotone depth of directed st-Connectivity is $\Omega(\log^2 n)$. This shows that $mon\text{-}NC_1 \neq mon\text{-}NC_2$. The lower bound was first proved by Karchmer and Wigderson and we will present here a proof by Grigni and Sipser. A similar lower bound was also proved for the non-directed case. We will present here the proof for the directed case, since it is simpler. As we saw in the previous lecture, the monotone depth of a (monotone) function is equal to the deterministic communication complexity of the corresponding monotone KW game. The monotone KW game corresponding to directed st-Connectivity can be presented as the following game, denoted by M: Given n vertices and two additional vertices marked by s and t:

- Player 1 gets a directed path D from s to t.
- Player 2 gets a coloring C of all the vertices by the colors 0 and 1, s.t., $C(s) = 0$ and $C(t) = 1$.
- The goal is to find an edge $(u, v) \in D$, s.t., $C(u) = 0$ and $C(v) = 1$.

We already saw an upper bound of $O(\log^2 n)$ for the deterministic communication complexity of this game. Here, we will prove a matching lower bound of $\Omega(\log^2 n)$. To prove our lower bound, we will use another communication game, denoted by $FORK$.

[1]Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel.
E-mail address: `ranraz@wisdom.weizmann.ac.il`.

July 21, 2000

Based on lecture notes taken by Yael Tauman, Yoav Rodeh and Yoad Lustig.

7.1. The *FORK* Game

For the *FORK* game, we assume that $n = l \cdot w$, and that the n vertices are partitioned into l levels (layers) of w vertices each. We denote the l layers by $L_1, \ldots, L_l$. We assume that we also have 3 additional vertices, marked by s, t_1, t_2. We think of s as a vertex in layer 0, and we think of t_1, t_2 as vertices in layer $l+1$. The *FORK* game is defined as follows.

Definition 1. *(FORK):* Given $n, l, w, s, t_1, t_2, L_1, \ldots, L_l$ as above,

- Player 1 gets a sequence of vertices $(x_0, x_1, \ldots, x_l, x_{l+1})$, s.t., for every $1 \le i \le l$, $x_i \in L_i$, and $x_0 = s$, $x_{l+1} = t_1$.
- Player 2 gets a sequence of vertices $(y_0, y_1, \ldots, y_l, y_{l+1})$, s.t., for every $1 \le i \le l$, $y_i \in L_i$, and $y_0 = s$, $y_{l+1} = t_2$.
- The goal of the two players is to find an i, s.t., $x_i = y_i$ and $x_{i+1} \ne y_{i+1}$.

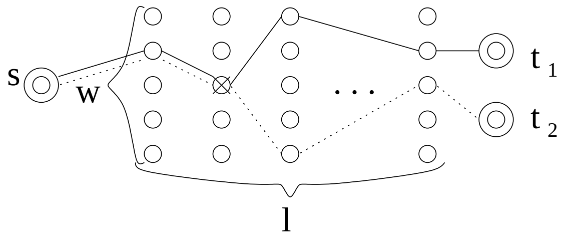

Figure 1. Player 1's sequence is solid. Player 2's sequence is dotted. The fork point is marked by an x.

Obviously, such an i always exists, since the sequences start at the same vertex (s), and end at different vertices ($t_1 \ne t_2$). We think of the sequence of Player 1 also as a path from s to t_1 and we think of the sequence of Player 2 also as a path from s to t_2. The *FORK* game is somewhat easier to deal with than the game G, because of the symmetry between the players in the *FORK* game. We will first show that the deterministic communication complexity of *FORK* is a lower bound for the deterministic communication complexity of the game G.

Proposition 7.1.1.

$$CC(FORK) \le CC(G).$$

Proof. The proof is by a reduction. Assume that we have a protocol P for the game G. We will show that P can be used also as a protocol for the game *FORK*. Recall that in the game G, Player 1 gets a directed path D from s to t, and Player 2 gets a coloring C of all vertices by 0 and 1, such that, s is colored 0 and t is colored 1. To use the protocol P, the players need to convert their inputs $(x_0, x_1, \ldots, x_l, x_{l+1})$ and $(y_0, y_1, \ldots, y_l, y_{l+1})$ (for the game *FORK*) to inputs for the game G.

The players convert the inputs in the following way. Define the vertex s in the game G to be the same as s in *FORK*. Define the vertex t (in the game G) to be the same as t_1 (in *FORK*). All other vertices stay the same (the vertex t_2 can actually be omitted). Player 1 uses the path $(x_0, x_1, \ldots, x_l, x_{l+1})$ as the path D for the game G. Player 2 generates a coloring C in the following way: a vertex is colored 0 iff it is one of the vertices in $(y_0, y_1, \ldots, y_l, y_{l+1})$. (Note that s is colored 0 because $y_0 = s$, and t is colored 1 because $t = t_1$ and $t_1 \ne t_2$).

The players can now use the protocol P to get an edge $(u, v) = (x_i, x_{i+1}) \in D$, such that, $C(u) = 0$ and $C(v) = 1$. Hence, $u = x_i$ is in $(y_0, y_1, \ldots, y_l, y_{l+1})$ and $v = x_{i+1}$ is not. That is, $x_i = y_i$ and $x_{i+1} \neq y_{i+1}$. ■

We will prove that

$$CC(FORK) = \Omega(\log(l) \cdot \log(w)).$$

The lower bound for G follows by setting $l = w = \sqrt{n}$. But first, let us show an upper bound for the communication complexity of $FORK$.

Proposition 7.1.2.

$$CC(FORK) = O(\log(l) \cdot \log(w)).$$

Proof. The proof is similar to the proof for the upper bound for the game G (described in the previous lecture). The players basically perform a binary search on the path $(x_0, x_1, \ldots, x_l, x_{l+1})$. In the first step, Player 1 sends the middle vertex in the path. Denote that vertex by x_i. Player 2 responds by one bit, indicating whether $x_i = y_i$ or not. If $x_i = y_i$ then the players can continue with the second half of the path. If $x_i \neq y_i$ then the players can continue with the first half of the path. The players continue to perform the binary search until they are left with a path of length 1. Altogether, the number of steps needed is $O(\log(l))$ and in each step the number of bits communicated is $O(\log(w))$. ■

Our goal will be to show that this upper bound is tight.

7.2. Restricted $FORK$ Protocols

Note that we can think of the inputs for the two players in the $FORK$ game as elements in $\{1, \ldots, w\}^l$. This is true since the start vertex s and the two end vertices t_1, t_2 are fixed and hence they are not part of the input. Other vertices x_i, y_i are known to be in layer L_i, and hence we can think of x_i, y_i as numbers in $\{1, ..., w\}$, (indicating the index of the vertex in layer L_i). From now on we think of each x_i and each y_i as elements of $\{1, ..., w\}$, and we think of the input for each player as an element of $\{1, \ldots, w\}^l$.

In order to prove a lower bound for $FORK$, we will consider restricted protocols that work on subsets of all inputs. Let us first introduce some notations. We denote the set $\{1, ..., w\}$ by $[w]$. We think of $[w]^l$ as a set of paths. When an element $(z_1, ..., z_l) \in [w]^l$ is associated with Player 1, we think of it as the path $(s, z_1, ..., z_l, t_1)$. When the same element is associated with Player 2, we think of it as the path $(s, z_1, ..., z_l, t_2)$. We will denote by S a subset of $[w]^l$. The density of a set $S \subset [w]^l$ is defined in the following way.

Definition 2. *(Density):* The density of a set $S \subset [w]^l$ is defined by,

$$\alpha(S) = \frac{|S|}{w^l}.$$

For any $S \subset [w]^l$, we consider the game $FORK^S$, which is the game $FORK$ restricted to inputs in S. That is, Player 1 gets an input $x \in S$ and Player 2 gets an input $y \in S$. The goal is the same as in the $FORK$ game. We say that a protocol P works on S if it gives a correct answer for every $x, y \in S$, that is, if it is a protocol for the game $FORK^S$. We say that a protocol P is an (α, l)-protocol if it works on some set S of density α.

Definition 3. *((α, l)-Protocol):* A communication protocol is an (α, l)-protocol if it works on some set $S \subset [w]^l$ with density $\alpha(S) = \alpha$.

Definition 4. *($CC(\alpha, l)$):* Denote by $CC(\alpha, l)$ the minimal communication complexity of an (α, l)-protocol.

That is, if $CC(\alpha, l) = d$ then there exists a set $S \subset [w]^l$ of density $\alpha(S) = \alpha$, and a protocol P with communication complexity d that works on S. Thus, we have two parameters: α is the density of the set and l is the length of the paths. The additional parameter, w, is hidden. We omit w from the notation because it will be fixed and will not change during the proof. We will prove that

$$CC(FORK) = CC(1, l) = \Omega(\log(l) \cdot \log(w)).$$

7.3. Trivial Lower Bound of $\Omega(\log(w))$

What can we say about the communication complexity of (α, l)-protocols ? Our first lemma (which is quite easy to prove) states that for any (α, l), if there exists an (α, l)-protocol with communication complexity $c > 0$ then there exists an $(\alpha/2, l)$-protocol with communication complexity $c - 1$. That is, if there exists a protocol with communication complexity $c > 0$ that works on a certain set S then there exists a protocol with communication complexity $c - 1$ that works on a set of size $|S|/2$. A similar lemma can be stated and proved for any communication game. (The proof doesn't use any property special to the $FORK$ game).

Lemma 7.3.1. For any α, l, if $CC(\alpha, l) > 0$ then

$$CC(\alpha, l) \geq CC(\alpha/2, l) + 1.$$

Proof. Let $c = CC(\alpha, l)$. Thus, there exists a set $S \subset [w]^l$ of density $\alpha(S) = \alpha$, and a protocol P with communication complexity c that works on S. Assume w.l.o.g. that Player 1 sends the first bit in the protocol P. Let $S_0 \subset S$ be the set of inputs for which Player 1 sends the bit 0, and similarly let $S_1 \subset S$ be the set of inputs for which Player 1 sends the bit 1. Since $S_0 \cup S_1 = S$, either S_0 or S_1 contain at least half of the inputs in S. Assume w.l.o.g. that $|S_0| \geq |S_1|$. Then, $\alpha(S_0) \geq \alpha/2$. Let P_0 be the rest of the protocol P, after assuming that the first bit (sent by Player 1) was 0. That is, P_0 works exactly like P, but without sending the first bit, and continuing as if the value of the first bit was 0. The communication complexity of P_0 is $c-1$. Obviously, P_0 works for any input (x, y), such that, $x \in S_0$ and $y \in S$. In particular, P_0 works on S_0, and hence, P_0 is an $(\alpha/2, l)$-protocol. ■

Lemma 7.3.1 can be applied as long as $CC(\alpha, l) > 0$. The next lemma (which is also easy to prove) shows that as long as α is large enough $CC(\alpha, l) > 0$. That is, if $CC(\alpha, l) = 0$ then the density α is relatively small.

Lemma 7.3.2. For any l and any $\alpha > 1/w$,

$$CC(\alpha, l) > 0.$$

Proof. Assume for a contradiction that $CC(\alpha, l) = 0$. Thus, there exists a set $S \subset [w]^l$ of density $\alpha(S) = \alpha$, and a protocol P with communication complexity 0 that works on S. We will show that $\alpha(S) \leq 1/w$.

Let $x, y \in S$ be the inputs for the two players. Since the communication complexity of P is 0, an answer is known without any communication. The answer must be correct also for the case $x = y$. In this case, for every $0 \leq i \leq l$, we have $x_i = y_i$, and hence the answer must be l (the answer l is correct since $x_l = y_l$ and $x_{l+1} = t_1 \neq t_2 = y_{l+1}$).

Since the answer is l for every $x, y \in S$, it must be that for every $x, y \in S$, we have $x_l = y_l$. Hence, for some $z \in [w]$, for every $x \in S$, we have $x_l = z$. Thus, $\alpha(S) \leq 1/w$. ■

Using Lemma 7.3.1 and Lemma 7.3.2, we can prove the (trivial) lower bound

$$CC(FORK) = CC(1, l) \geq \log_2(w).$$

This is proved by repeatedly applying Lemma 7.3.1, as long as the density is larger than $1/w$. (Lemma 7.3.2 guarantees that indeed we can apply Lemma 7.3.1 as long as the density is larger than $1/w$). That is,

$$CC(1, l) \geq CC(1/2, l) + 1 \geq CC(1/4, l) + 2 \geq ... \geq CC(1/w, l) + \log_2(w) \geq \log_2(w),$$

(where we assume for simplicity that w is a power of 2). In the next lecture, we will prove the stronger (and tight) lower bound

$$CC(FORK) = CC(1, l) = \Omega(\log(l) \cdot \log(w)).$$