DSC180A FA25: Rao-Yehudayoff Book Notes

Ciro, Darren, Ivy, Ryan

November 24, 2025

Contents

1 Deterministic Protocols

Notes on Ch1 of the Rao-Yehudayoff book from the second meeting, Monday Oct 6 2025. Speakers: Ciro and Ivy. This chapter covers deterministic protocols; rectangles; richness; fooling sets. Basic lower bound methods.

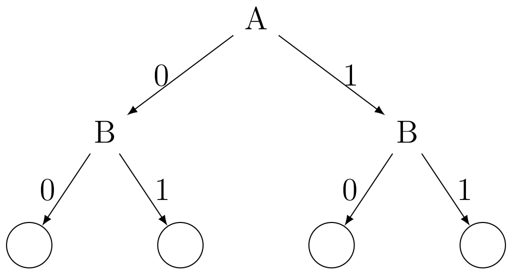
1.1 Rectangles

A **deterministic protocol** is a pre-determined procedure between how patries commnicate for whatever problem we give it. The goal of this communication is to compute something deterministic based on the input.
One way to model this is as a tree. Consider two parties, Alice and Bob. Alice has input x, Bob has y, and they wish to compute something based on x, y.

Example 1.1 (**Eq** example) Suppose Alice and Bob work to find if $x = y$, equivalently compute

$$\mathrm{Eq}(x,y) = \begin{cases} 1 & x = y, \\ 0 & x \neq y. \end{cases}$$

Suppose further x, y are two bits. We can represent this communication as a **tree**:



Remark 1.2 Notice that communication complexity presents a combinatorial setting to solve problems, which have much more mathematical tools compared to general complexity. Therefore,

Lemma 1.3 A protocol that sends c bits has a protocol tree of depth c. This implies you can partition your matrix (whose rows are the possibilities of x and columns y) into 2^c monochromatic rectangles. (Recall that a partition is a disjoint collection whose union is the whole thing.)

Definition 1.4 A **monochromatic rectangle** is a subset of the matrix $R \subseteq X \times Y$ such that if $(x, y), (x', y') \in R$, then $(x, y'), (x', y) \in R$.

Example 1.5 Consider the following matrix:

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

This cannot be one monochromatic rectangle. However,

$$\begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix}$$

can be expressed as one monochromatic rectangle.

Example 1.6 Notice that the matrix for Eq from Example 1.1 is just the identity matrix. By 1.4 there must be at least n monochromatic rectangles, one for each 1 on the diagonal. Therefore, no protocol exists that does better than n bits communicated for Eq.

Remark 1.7 Notice that since we care about the worst case, that is all x, y, we care more about the number of rectangles, not the size of the rectangles; it is possible that our matrix is covered mostly by 1 rectangle, but many (hard) inputs can only be covered by very small rectangle.

1.2 Fooling Sets

Definition 1.8 TODO; A **fooling set** is a subset $F \subseteq X \times Y$ such that if $(x, y), (x', y') \in F$ and $f : X \times Y \to 0, 1$ is the evaluation function, then either $f(x, y') \neq f(x, y)$ or $f(x', y) \neq f(x, y)$ such that it cannot be a single monochromatic rectangle.

1.3 Richness

Definition 1.9 Define **richness** as a tuple (u, v) describing the matrix of a protocol, particularly when $|X| < |Y|$ or vice versa. u refers to the number of rows that have v 1s. Together with the size of X, Y, and the number of 1s in the matrix, you can give an upper bound on the maximum size of a 1-monochromatic-rectangle and therefore a lower bound on the number of monochromatic rectangles (and thus communication complexity in general).

Example 1.10 Consider the following matrix:

$$\begin{matrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

The above has richness $(3, 1), (2, 2), (1, 3)$. This tells us a maximum size of a 1-monochormatic rectangle (in particular, 2 by 2), which just happens to be tight here.

Remark 1.11 Note that rectangles and richness easily extend to relations, where there may be more than one possible input (for example, consider indexed sets X, Y and finding i where $X_i = Y_i$).

1.4 Computing Functions and Relations

Definition 1.12 Define the **computational complexity** of $g : X \times Y \to \{0, 1\}$ denoted $\mathrm{CC}(g)$ is the minimum depth of the protocol tree of g. Define $P(g)$ as the minimum number of leaves, equivalently, the number of monochromatic triangles of the protocol tree.

Lemma 1.13 $P(g) \le 2^{\mathrm{CC}(g)}$. Also, $\sqrt{\mathrm{CC}(g)} \le \log_2 P(g) \le \mathrm{CC}(g)$.

Example 1.14 Suppose $g : X \times Y \to \{0, 1\}$ and g requires c bits of communication. Suppose we wish to compute g^m. If g is a function, $\mathrm{CC}(g^m) = \mathrm{CC}(g)^m$ is an open problem. If g is a relation, you cannot do this. TODO; See book why.

Theorem 1.15 $P(g^m) = P(g)^m$.

Remark 1.16 We can study the number of leaves but not the depth...

2 Rank

2.1 What is Rank?

Definition 2.1 The **rank** of a matrix is the number of linearly independent variables (or columns/rows) in that matrix. Essentially, rank measures the number of variables not accounted for by other variables.

This idea connects closely to communication complexity: the communication complexity represents the number of independent pieces of data that must be sent. Dependent variables need not be transmitted—only the independent ones. Hence, rank correlates with communication complexity.

2.2 Properties of Rank

Lemma 2.2 Given matrices A and B:

$$\text{If } A \subseteq B, \quad \mathrm{rank}(A) \le \mathrm{rank}(B),$$
$$\mathrm{rank}\begin{pmatrix} A & C \\ 0 & B \end{pmatrix} \ge \mathrm{rank}(A) + \mathrm{rank}(B),$$
$$|\mathrm{rank}(A) - \mathrm{rank}(B)| \le \mathrm{rank}(A + B) \le \mathrm{rank}(A) + \mathrm{rank}(B).$$

- Constants used in a function $g(i, j)$ do not change the rank.
- All monochromatic rectangles are submatrices with rank $= 1$.
- The largest possible rank of an $m \times n$ matrix is $\min(m, n)$.
- $\mathrm{rank}(M) \le$ (number of rank-1 matrices).

2.3 Tensors

Definition 2.3 The **tensor product** of two matrices is the projection of one matrix onto another, resulting in an $n \times n'$-dimensional space.

The tensor product multiplies ranks:

$$\text{If } C = A \otimes B, \quad \text{then } \operatorname{rank}(C) = \operatorname{rank}(A) \cdot \operatorname{rank}(B).$$

This makes intuitive sense—projecting onto additional dimensions increases rank multiplicatively.

2.4 Lower Bounds Using Rank

Lemma 2.4 The logarithm of the rank gives a lower bound on communication complexity:

$$2^c \geq \operatorname{rank}(M) \quad \Rightarrow \quad c \geq \log_2 \operatorname{rank}(M).$$

Since 2^c represents the maximum number of monochromatic rectangles that c bits can encode, $\log_2(\operatorname{rank}(M))$ is a fundamental lower bound.

2.5 Non-Negative Rank

Definition 2.5 For a non-negative matrix M, the following are equivalent:

1. The smallest number of **non-negative rank-1 matrices** that sum to M.
2. The smallest integer r such that $M = UV$ where U and V are non-negative matrices of sizes $m \times r$ and $r \times n$, respectively.
3. The smallest number r such that every column of M is a non-negative linear combination of r non-negative vectors of size m.

Example 2.6 The matrix

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

has $\operatorname{rank}(M) = 3$ but $\operatorname{rank}_+(M) = 4$.

Lemma 2.7 If a matrix is 3×3 or smaller, $\operatorname{rank}_+(M) = \operatorname{rank}(M)$.

Lemma 2.8 (Fact 2.12.) $min\{m, n\} \geq \operatorname{rank}_+(M) \geq \operatorname{rank}(M)$.

Remark 2.9 Non-negative rank equals regular rank when $\operatorname{rank}(M) \leq 2$, but may diverge for larger matrices.

Conjecture 2.10 Determining $\operatorname{rank}(M)$ is polynomial-time solvable, being $(O(n^3))$ (e.g., via SVD or Gaussian elimination), while determining $\operatorname{rank}_+(M)$ or checking whether $\operatorname{rank}_+(M) = \operatorname{rank}(M)$ is NP-hard.

2.6 Non-Negative Rank Bounds

Remark 2.11 Despite computational difficulty, non-negative rank often yields tighter lower or upper bounds on communication complexity.

2.7 Log-Rank Conjecture

Conjecture 2.12 There exists a constant c such that for every non-constant matrix M,

$$\mathrm{CC}(M) \leq (\log \mathrm{rank}(M))^c.$$

2.8 Upper Bounds and Proof Sketches

Theorem 2.13 (2.14) The communication complexity of a Boolean matrix M is at most $O(\mathrm{rank}(M) \log^2 \mathrm{rank}^+(M))$.

Proof. Partition M as $M = [A\ B]$, where A is a 1-rectangle. Without loss of generality, if Alice's input is in A, she sends 1 bit to Bob. Each iteration halves the matrix, akin to a binary search, so the process requires $O(\log n)$ rounds.

Lemma 2.14 (2.18) Any $n \times n$ matrix with r ones and rank k contains a monochromatic submatrix of size at least $2^{-O(\sqrt{r}\log(r))}$.

Theorem 2.15 (2.17) If $\mathrm{rank}(M) = r$, then $\mathrm{CC}(M) = O(\sqrt{r}\log^2(r)))$.

Proof. Follows from iterative binary partitioning, reducing the matrix size by a constant factor at each step. A tighter bound of $O(\sqrt{\mathrm{rank}(M)} \log \mathrm{rank}(M))$ was proven by Lovett (2014).

2.9 Properties of Boolean Matrices

Claim (2.19) If at least half of the entries in M are 0s, then there is a submatrix T of size at least $mn * 2^{-O(\sqrt{r} log(r))}$ such that the fraction of 1s in T is at most $\frac{1}{r^3}$.

Claim (2.20) If M is a 0/1 matrix of rank $r \geq 2$ where at most $\frac{1}{r^3}$ entries are 1s, then there exists a submatrix with at least half the rows and half the columns of M that contains only 0s.

Lemma 2.16 (2.22) Any Boolean matrix M of rank r can be expressed as $M = UV$, where U is an $m \times r$ matrix with vectors of lenght at most 1, and V an $r \times n$ matrix with each row vector of length at most $\sqrt{r}$.

Theorem 2.17 (2.21) Let $K \subseteq R^T$ be a symmetric convex body such that the unit ball is the largest-volume ellipsoid contained in K. Then every element of K has Euclidean length at most $\sqrt{r}$.

Proof. Since scaling preserves convexity, we can assume K forms a convex polytope. By Theorem 2.21, all rows of U have norm $\leq \sqrt{r}$. Because M is Boolean, for each pair (u_i, v_j), the dot product satisfies $u_i^\top v_j \in \{0, 1\}$. Since $|row(A)|_2 \leq 1$ with equality achieved when r = 1, we get the triangle inequality that $|col(B)|_2 \leq 1$, as desired. $\square$

3 Randomized Protocols

3.1 Defining Randomness

Definition 3.1 To talk about randomness in communication protocols, we begin by giving a careful definition of it. A protocol uses **public randomness** or **public coins** when all parties have access to a common shared random string. We denote this string by R.

A protocol uses **private randomness** or **private coins** if each party privately samples an independent random string. In the case of Alice and Bob, we call these strings R_a and R_b respectively.

We require R, R_a, R_b, and the input (X, Y) as independent uniform and the only source of randomness in the protocol. Thus, a random protocol is simply a distribution over deterministic protocols.

Lemma 3.2 Every private coin protocol can be simulated by a public coin protocol.

Proof. Simply use the public coin as the private coin. □

Lemma 3.3 Every public coin protocol can be simulated by a private coin protocol, albeit with a small increase in the length of the protocol.

Proof. Deferred. □

Definition 3.4 By introducting randomness, we also allow **error** into our functions. There are two ways to measure the probability that a random protocol makes an error:

A random protocol has error ϵ in the **worst-case error** if for every input, the probability that the protocol makes an error is at most ϵ. That is, $\forall x, y$,

$$\underset{R,R_a,R_b}{Pr}[\pi_{R,R_a,R_b}(x, y) \text{ is wrong}] \leq \epsilon,$$

where π_{R,R_a,R_b} is the deterministic protocol obtained by fixing said variables.

Given a distribution on inputs μ, we say that a protocol has error ϵ with respect to μ if the probability that the protocol makes an error is at most ϵ when the inputs are sampled from μ. That is,

$$\underset{R,R_a,R_b,(X,Y)}{Pr}[\pi_{R,R_a,R_b}(X, Y) \text{ is wrong}] \leq \epsilon.$$

The **length** of a protocol is defined to be the maximum depth of all the deterministic protocol tress the random protocol may generate.

Lemma 3.5 If a randomized protocol never errors, then we can fix the randomness to obtain a deterministic protocol that is always correct.

Corollary If a randomized protocol has less length than its deterministic counterparts, then it must introduce some form of error to the output of the protocol.

For any random protocal, we can convert it easily into a determanistic protocal, simply by fixing a coin or input, and then consiter how it behaves. Further, Public and Private coin protocals can be cycled between. When converting from a private to a public protocal, we mearely need to had both parties agree to share thier private coins. For public to private, it can be shown that this conversion can be done with only an additional $\log(\frac{n}{\epsilon^2}) + \mathcal{O}(1)$ bits.

3.2 Yaos minimax Principle

Theorem 3.6 Yao's Minimax principle is a very helpful principle for proving lowerbounds on randomized protocals. Formaly, it is an extention of the general minimax principle, $\min_{x\in\mathbb{A}} \max_{y\in b} xMy = \max_{y\in B} \min_{x\in A}$, to expectations over random variables. Informally, it states that the optimal performance of any random protocal is equal to that of any determanistic protocal who's input is sampled from a worst case distribution, chosen to be as hard as possible for the algorithm to solve. Thus, for many random protocals,the process of finding a lower bound becomes that of finding such a worste case input, and then proving the lower bound for that input.

3.3 Equality

In equality, Alice and Bob are given n bit strings x, y and wish to know if these strings are the same or not. We know its determinstic communciation complexity is $n + 1$.

Example 3.7 **Simple Random Equality Protocol:** Let Alice and Bob have a shared random coin of length $k2^n$. This allows Alice and Bob to recover a random function

$$h : \{0,1\}^n \to \{0,1\}^k$$

which we can treat as a hashmap. Then Alice can send $h(x)$ to Bob and Bob responds with a bit indicating if $h(x) = h(y)$.

Here $k+1$ bits are communicated and the probability of making an error is at most 2^{-k}; the probability $h(x) = h(y)$ yet $x \neq y$ is 2^{-k}, the probability k bits of x, y are the same (when hashed).

Remark 3.8 Though the protocol is efficient it requires a large number of shared random bits. We can try to reduce the number of bits, introducing a new dimension to optimize in our protocols.

Example 3.9 **Error-correcting Equality Protocol** An (example of an) **error correcting code** is a function

$$C : \{0,1\}^n \to [2k]^m$$

such that if $x \neq y$, then $C(x)$ and $C(y)$ differ in all but $m/2^{-\Omega(k)}$ coordinates. (When $m = 10n$, for any k most functions C will be error-correcting codes.)

Given a code, Alice can pick a random coordinate of $C(x)$ and send it to Bob, and Bob can check whether $C(x) = C(y)$. Alice takes $\log_2 m$ bits to send the index and $\log_2 2^k = k$ bits to send the value. Thus the protocol takes $O(\log n + k)$ bits of communication.

The probability of making an error is at most $2^{-\Omega(k)}$, by the definition of the error-correcting code.

Example 3.10 **Greater-than Random Protocol** Deterministic protocols take $\log n$ bits. A randomized protocol taking $O(\log \log n)$ bits exists.

A simpler random protocol in $O(\log \log n \cdot \log \log \log n)$ can be done by using the above equality protocol to check if the first $n/2$ most significant bits of x, y are the same. This allows us to perform a binary search on which bit differs first.

3.4 Disjointness

Example 3.11 **k-Disjoint Random Protocol** Suppose Alice and Bob are given sets $X, Y \subseteq [n]$, each of size at most k. They wish to check for intersection. By the rank method, at least $\log \binom{n}{k}$ bits are required.

When $k \ll n$, the deterministic protocol of length $O(k)$ is more efficient than any deterministic protocol. Here is how it goes:

Shared randomness is used to generate $R_1, R_2, \ldots \subseteq [n]$. Alice announces the smallest index i with $X \subseteq R_i$ and Bob does the symmetry with j. This takes $2(\log i + \log j + 2)$ bits. Alice replaces her set with $X \cap R_j$ and Bob with $Y \cap R_i$. Notice that if $X \cap Y = \varnothing$ this remains true, and same if $X \cap Y \neq \varnothing$.

Repeat the above until $O(k)$ bits are communicated. If either set becomes empty, X, Y are disjoint. If both sets are non-empty, they conclude X, Y intersect.

Theorem 3.12 In the above, Alice and Bob arrive at the correct conclusion with probability at least $2/3$.

Lemma 3.13 The expected length of the first step is at most $2(|X| + |Y| + 2)$.

Proof. The probability R_1 contains $|X|$ is $2^{-|X|}$. If it does not, then

$$\mathbb{E}[i] = 2^{-|X|} \cdot 1 + (1 - 2^{-|X|})(\mathbb{E}[i] + 1) \implies \mathbb{E}[i] = 2^{|X|}.$$

Same for Y. Then

$$\mathbb{E}[2 \log i + 2 \log j + 4] \leq 2(\log \mathbb{E}[i] + \log \mathbb{E}[j] + 2) = 2(|X| + |Y| + 2). \qquad \square$$

Lemma 3.14 In the second round, the expectation is that the sets are half as large. Formally, for $i \in [n]$ let $X_{i,s}$ be the indicator variable for if $i \in X$ before step s on the above process. Define $Y_{i,s}$ similarly. If X, Y are disjoint,

$$\mathbb{E}\left[\sum_{i \in [n]} \sum_{s=1}^{\infty} X_{i,s} + Y_{i,s}\right] \leq 4k.$$

Proof. Notice

$$\mathbb{E}\left[\sum_{i\in[n]}\sum_{s=1}^{\infty} X_{i,s} + Y_{i,s}\right] = \sum_{i\in[n]}\sum_{s=1}^{\infty} \mathbb{E}[X_{i,s}] + \mathbb{E}[Y_{i,s}].$$

If $X_{i,1} = 0$, then $X_{i,s} = 0$ for all s. If $X_{i,1} = 1$, the probability $X_{i,s} = 1$ is 2^{-s+1} so

$$\sum_{i\in X}\sum_{s=1}^{\infty} \mathbb{E}[X_{i,s}] = \sum_{i\in X}\sum_{s=1}^{\infty} 2^{-s+1} \leq 2k.$$

Applying the same to $Y_{i,s}$ gives $\leq 4k$. □

Theorem 3.15 (Markov's Inequality) For any random variable $X \geq 0$,

$$\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}[X]}{a}.$$

Lemma 3.16 The previous lemma implies that the parties communicate at most $16k$ bits. By the Markov inequality, the probability the protocol communicates more than $3\cdot 16k = 48k$ bits is at most $1/3$.

3.5 Hamming Distance

Definition 3.17 If Alice and Bob are given two strings $x, y \in \{\pm 1\}^n$, define the **Hamming distance** as

$$H(x,y) = \Delta(x,y) = |i \in [n] \mid x_i \neq y_i| = \frac{n - \langle x, y\rangle}{2}.$$

Definition 3.18 Say that a protocol π approximates Δ up to m if

$$|\pi(x,y) - \Delta(x,y)| \leq m$$

for all inputs x, y. For $alpha < 1/2$, approximating Δ up to αn requires deterministic CC $\Omega(n)$.

Theorem 3.19 **Hamming Distance Random Protocol** Suppose Alice and Bob wish to estimate Δ up to m. Both use shared randomness to sample $i_1, \ldots, i_k \in [n]$ independently and uniformly. They communicate $2k$ bits the **empirical distance**

$$\gamma = (1/k) \cdot |j \in [k] \mid x_{i_j} \neq y_{i_j}|$$

and output γn.

Lemma 3.20 The probability of making an error is at most $1/e$.

Proof. Deferred. □

4 Number on Foreheads

4.1 Multi-Party Communication

Definition 4.1 In **nunber on forehead** problems, we have k parties where each party can see the input of all other parties but not themselves.

Remark 4.2 Why study this weird setup? Normally we consider number in hand setups, where each party only knows their own number. The reason is that many of these tools are used to prove lower bounds. (Opposed to algorithms which focus more on lower bounds). Because number of foreheads is a much more powerful model than number in hand, we can show lower bounds for many more problems that do not even look like communciation.

In particular, this model is applicable even when we allow shared information, but number in hand would not.

Example 4.3 Consider the **pointer chasing problem**. There are three players. The first player holds a pointer in $\log n$ bits to n nodes. The second holds a permutation of the n nodes. The third has labels for for the n permuted nodes. If we have the number in hand model, this would take $2\log n$ bits. However, with number for forehead, a trivial protocol takes $O(n)$ bits and a non-trivial one takes $O(n/\log n)$ bits.

Example 4.4 Consider **equality with k players**. Then you only need $k-1$ bits of communication. Consider the case with 3 players. Alice can see if Bob's number if equal to Charlie and send a 1 if so. Bob can then do the same with Alice and Charlie.

Example 4.5 Consider the **intersection with k parties problem**, and each party has a set $X_i \subseteq [n]$ as input on their forehead. The goal is to compute

$$|X_1 \cap X_2 \cap \cdots \cap X_k|$$

We can map this as a matrix problem: Consisder a $k \times n$ matrix where entry (i, j) means player i has number j. Thus we want to count the number of columns with all 1s, such as the middle column of the following matrix $k \times n$ matrix:

$$\begin{bmatrix} 1 & \cdots & 1 & \cdots & 0 \\ \vdots & \ddots & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & \ddots & \vdots \\ 0 & \cdots & 1 & \cdots & 1 \end{bmatrix} \in \mathbb{F}_2^{k\times n}.$$

A simple $O(n)$ algorithm would have one player count all the other parties' numbers, then have another party verify it out. A more clever algorithm is below.

For each party i, count C_{i0}, C_{i1}, and so on, where C_{ij} is the number of columns containing exatcly j ones visible to the ith party. The parties announce $C_{i,j}$ for all i, j. Thus the communciation complexity is at most $O(k^2 \log n)$.

The above example only works if given all C_{ij} for all i, j, there is a unique possible count for the number of all 1s columns. We show that now.

Proof. Let Z_r denote the number of column with r ones. We show there is a unique tuple $Z = (Z_0, \ldots, Z_k)$ consistent with the C_{ij}s.

LEMMA 4.6 Suppose $Z = A$, $Z = B$ are two solutions that are both consistent with all C_{ij}s. Then

$$|A_r - B_r| = \binom{k}{r}|A_0 - B_0|$$

for each $r \in \{0, 1, \ldots, k\}$.

□

EXAMPLE 4.7 Consider exactly 3 parties. Each party has a number from $[n]$ on their forehead, and they want to know if these sum to n. The trivial protocol takes $O(\log n)$ by announcing one number and having the party whose number was announced compute the sum.

There is another protocol that takes $O(\sqrt{\log n})$. Take $\Delta = A + B + C - n$. One does $-A + 2(n - A - C)$. Another does $(n - B - C) + 2B$.

THEOREM 4.8 **Berhend's Coloring**: One can color the set $[m]$ with $2^{O(\sqrt{\log m})}$ colors with no monochromatic 3-term arithmetic progression. Namely, for each $a, b \in [m]$, if all three numbers $a, a + b, a + 2b$ are in $[m]$, they do not have the same color.

Just by comparing the colors, they will know whether $\Delta = 0$ which is equivalent to knowing if the sum is equal to n.

REMARK 4.9 Recent results have allowed us to show we need $2^{(\log m)^c}$ where $0 \leq c \leq 1$ is a small constant. The paper is very technical.

4.2 Cylinders

DEFINITION 4.10 Any set $S \subseteq X_1 \times \cdots \times X_k$ can be described by its **characteristic function**

$$F_S(x_1, \ldots, x_k) = \begin{cases} 1 & (x_1, \ldots, x_k) \in S \\ 0 & \text{otherwise} \end{cases}$$

DEFINITION 4.11 A **cylinder** is a set constant in a dimension X_s.

Remark 4.12 Let us take a step back. In the 2 player case, the inputs where the output is 1 form 2^c rectangles. Cylinders are the greater than 2 dimension counterpart. Consider a decision tree with three players:

$$\begin{tikzcd}
& & f(B,C) \arrow[dl, "0"'] \arrow[dr, "1"] & \\
& g(A,C) \arrow[dl, "0"'] \arrow[dr, "1"] & & \cdots \\
\cdots & & f(B,C) \arrow[dl] \arrow[dr] & \\
& h(A,B) \arrow[d] & & \cdots \\
& \text{Result} & &
\end{tikzcd}$$

Then the intersection forms at most 2^c cylinder intersections where the output is 1.

Remark 4.13 TODO: Ramsey theory

Term Index

Example Index