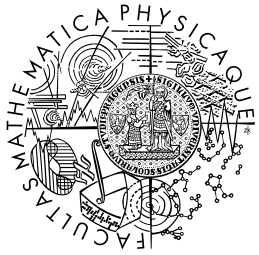

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Mgr. Pavel Dvořák

Limits of Data Structures, Communication, and Cards

Computer Science Institute of Charles University

Supervisor of the doctoral thesis: prof. Mgr. Michal Koucký, Ph.D.

Study programme: Computer Science

Study branch: Theory of Computing, Discrete Models and Optimization

Prague 2021

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank my supervisor, Michal Koucký, who had enough patience with me and guided me through the whole Ph.D. study. Further, I would like to thank all my coauthors, without them this thesis would not exist, especially Bruno Loff, with whom I spent hours discussing information theory; and I would like to thank Anna Koucká, my high school teacher, who sparked my interest in mathematics. Last but not least, I would like to thank my family who supports me the whole time, even though they do not understand very much, what I actually do.

Title: Limits of Data Structures, Communication, and Cards

Author: Mgr. Pavel Dvořák

Institute: Computer Science Institute of Charles University

Supervisor: prof. Mgr. Michal Koucký, Ph.D., Computer Science Institute of Charles University

Abstract: In this thesis, we study several aspects of computational complexity. One of the main topics is the complexity of data structures, which are algorithms for efficient storing data supporting efficient queries to the data. In the case of dynamic data structures, they also allow modification of the data before querying. A long-standing open problem in this area is to prove an unconditional polynomial lower bound of the trade-off between the update time and the query time of an adaptive dynamic data structure computing some explicit function.

We provide an unconditional polynomial lower bound for a restricted class of semi-adaptive dynamic data structures computing functions of large corruption bound, that generalizes the result by Ko and Weinstein [FOCS '20] who provided such a lower bound for data structures computing the Disjointness function. Further, we provide conditional lower bounds for certain static data structures computing permutation inversion, and polynomial evaluation and inversion. These lower bounds beat the best-known unconditional lower bounds for the problems of interest.

Further, we study the communication complexity of the elimination problem, which is a problem closely related to the direct sum. In the elimination problem, Alice and Bob get k n-bit strings each, $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$, respectively. Their goal is to communicate as few bits as possible and then output a string $o \in \{0,1\}^k$ which is different from the string z, where $z_i = f(x_i, y_i)$ for some fixed boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ – thus they eliminate one incorrect answer. We consider the elimination problem for f being the Greater-Then (GT) function, i.e., $\mathrm{GT}(x, y) = 1$ if and only if $x \geq y$, where x and y are considered as n-bits numbers. We prove the trivial protocol for this problem (where Alice and Bob compute if $x_i \geq y_i$ for some i) is optimal.

The last topic of this thesis is so-called card-based protocols. In this model, Alice and Bob represent their inputs by cards and they want to compute a given function without revealing their inputs. We identify a class of card-based protocols that correspond to the circuit class NC^1. Further, we study a connection between card-based protocols and Turing machines, and we propose new card encodings of the input.

Keywords: Lower Bound, Data Structure, Communication Complexity, Card-based Protocol

Contents

Funding. *Chapter 3:* The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement n. 616787. *Chapters 2, 4, 5:* The projects were supported by Czech Science Foundation GAČR (grant #19-27871X).

Introduction

The typical question of a computational complexity scientist is: Is this algorithm optimal, or can we design a better one? Without an answer to this question, we do not know whether we should spend any effort to improve the algorithm or we should rather focus on some other problem. If we suspect that some algorithm is actually optimal, then it would be nice if we can also prove such a statement, for example, prove that any algorithm that solves a problem P needs at least time t. To state such a lower bound, we first need to define a computational model, which formalizes what is an algorithm, what resource we actually measure (usually computational time or space), etc. There are many models which are studied. Favorite models for describing general algorithms are Turing machines or circuits [8]. However, to prove (non-trivial) unconditional lower bound for these models seems like a really hard task. On the other hand, improvements of some best-known algorithms are long-standing open problems. Thus for many problems, there is a substantial gap between the lower bounds and upper bounds (given by the algorithms).

One way how to deal with this gap is to find at least conditional lower bounds, in an ideal scenario, such a lower bound matches the upper bound. For a conditional lower bound, one proves that too efficient algorithm for solving a problem of interest would contradict some standard complexity assumption (such as Exponential Time Hypothesis which asserts that deciding a given formula satisfiability can not be solved in sub-exponential time [47]). Thus, if we have a conditional lower bound that matches the upper bound, we provide at least some evidence that we have the optimal algorithm (if we believe the underlying complexity assumption). Also, it indicates that proving the unconditional lower bound for the problem is hard, as popular complexity assumptions, that are used for the conditional lower bounds, are usually based on some long-standing open problems. In Chapter 4, we provide a conditional lower bound based on the Network Coding Conjecture (NCC) by Li and Li [70], which recently got a lot of attention in the area of conditional lower bounds.

Another way to overcome the difficulty of proving unconditional lower bounds is to somehow restrict the class of possible algorithms and try to prove unconditional lower bounds for them. A favorite model for unconditional lower bounds is a communication model introduced by Yao [105]. There are two parties – Alice and Bob. Alice gets $x \in \{0,1\}^n$ and Bob gets $y \in \{0,1\}^n$ and their goal is to compute $f(x,y)$ where f is a Boolean function $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. They have unlimited computational power and we measure how many bits they have to exchange to compute $f(x,y)$. In randomized models, they can use random bits – shared or private. The (randomized) communication complexity of f is the length of the shortest protocol computing f. This model is weak in the sense that we are able to prove unconditional lower bounds in it. However, it is still strong enough that there are some non-trivial protocols. A trivial protocol of length n for a function f is the following: Alice sends her input to Bob and he computes the output (or another way around). As mentioned above, there are some more efficient protocols. For example, let $\mathrm{EQ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function which equals to 1 if and only if $x = y$. There is a randomized protocol using

hashing for EQ [65] of length $O(\log n)$ and error probability smaller than $1/n$. Although the communication model is quite abstract, it is closely related to data structures, which are studied in several chapters of this thesis.

Data Structures. Informally, data structures are algorithms for efficiently storing and retrieving data. Since data structures form a restricted class of algorithms (but a very important class), we are able to prove some non-trivial unconditional lower bounds for them (unlike for general algorithms).

A static data structure is a pair of algorithms where the first one encodes a data $D \in \{0,1\}^n$ into a memory consisting of c cells each of w bits (so-called cell probe model, which was introduced by Yao [105]). Thus, the data structure uses space $s = c \cdot w$. The second algorithm is a query algorithm that should answer queries from a fixed set Q, where formally each query is a function $r : \{0,1\}^n \to \{0,1\}^k$. The query time t_q of the data structure is the maximum number of cells accessed (so-called cell probes) by the query algorithm in order to compute $r(D)$, where the maximum is over $r \in Q$. The probes may be completely adaptive, i.e., the next cell probe may depend on the values of the probed cells so far. There are two trivial solutions. The first one is to store the answers for all queries in Q, thus $s = |Q| \cdot k$ and $t_q = k/w$ (it needs to read k bits in total). The second one is to store the data D in a raw form and compute the whole answer for a query during the query phase, thus $s = n$ and $t_q = n/w$ (it needs to read n bits in total). We are interested in whether we can have a better trade-off between s and t_q or the trivial solutions are optimal.

Lower bounds in the cell probe model are quite strong in a sense they also apply for computations on real machines (as we charge only the access to the memory and not the computation). By a counting argument, it can be shown [71] that for most functions the trivial static data structures are optimal. However, it is hard to prove it for some explicit function. One of the key tools in data structure lower bounds is communication complexity as one can use an efficient static data structure computing a function f to design an efficient protocol for f [65, 73]. Despite the great interest in lower bounds for static data structures (e.g. [6, 106, 72, 32, 40]), only little is known: The best known lower bound for a static data structure computing an explicit function is $t_q \geq \Omega(\log n)$ for data structures of linear space and polynomial many queries ($|Q| = \mathit{poly}(n)$), e.g. [83, 89, 67]. Improving such lower bound to $t_q \geq \omega(\log^2 n)$ even for non-adaptive data structures seems like a really hard problem as shown by Dvir et al. [35]. They proved that such an improvement would solve long-standing problems in circuit complexity and matrix rigidity [100]. In Chapter 4, we provide an $\Omega(\log n/\log\log n)$ conditional lower bounds for certain static data structures for permutation inversion, and polynomial evaluation and interpolation, based on the Network coding conjecture. Although these bounds do not cross the logarithmic barrier, they still beat the best-known lower bounds for the problems of interest.

A slightly better situation is in the area of dynamic data structures. In a dynamic data structure problem, we wish to encode the data in such a way that allows us also to update the data. There is a set of updates U where each update is a function $p : \{0,1\}^n \to \{0,1\}^n$. Thus, if there is an encoding of data D stored in the memory after an update $p \in U$ we want to have an encoding of data $p(D)$ in the memory. The update time t_u of the data structure is the maximum number

of cells accessed during the update, where the maximum is over $p \in U$. The query time t_q is defined in the same way as for the static data structures.

Again, there are two trivial solutions in a similar way as for the static data structures. A long-standing open problem in this area is to prove an unconditional polynomial lower bound on the trade-off between the update time and the query time (with an assumption that the data structure uses polynomial space). In contrast to static data structures, here we are interested in lower bounds for adaptive data structures, as polynomial lower bounds for non-adaptive dynamic data structures can be obtained by a simple coding argument [19]. The logarithmic query time barrier was crossed for dynamic data structures. There are lower bounds $\max\{t_q, t_u\} \geq \log^{1+\delta} n$ for several dynamic data structures problems [84, 85, 66, 68], where δ is between 1/2 and 1. In a recent paper, Ko and Weinstein [64] provided a polynomial lower bound for a semi-adaptive dynamic data structure that solves the Pătraşcu's multiphase problem [88]. In Chapter 2, we generalize their result to a similar lower bound for all functions of a large corruption bound.

Direct Sum and Related Problems. Another important question in complexity is the so-called direct sum question. It asks if we compute k independent instances of f at once, do we need k times to run the optimal algorithm for f or can we achieve some savings? In some models, we can not achieve any savings – like decision trees [80, 50]. On the other hand, the direct sum is not true in various communication models [53, 94, 63]. The understanding of the direct sum question would shed light on various aspects of complexity. A direct sum theorem for communication complexity would imply the separation of NC^1 from NC^2 [54], where NC^i is the class of circuits of polynomial size and depth $O(\log^i n)$. The direct sum for information complexity is used to prove lower bounds in communication complexity (see for example [11, 51, 12, 17]).

There are several problems related to the direct sum questions. Beimal et al. [15] introduced a problem where Alice and Bob get k inputs $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$, respectively, and they have to compute one instance $f(x_i, y_i)$ according to their choice. Provided that the instances are independent there is a hope of picking some easy instance among the k instances. Another problem is to select a positive instance among k independent instances – this problem is studied in structural complexity theory, starting by Selman [92]; or the problem of distinguishing k positive instances from k negative instances [15].

The computationally least difficult problem among those is the problem of *elimination* [7, 15]. Let $f : X \times Y \to \{0, 1\}$ be a function, Alice and Bob get again k inputs $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$, respectively, and they have to output a string different from $f(x_1, y_1) \ldots f(x_k, y_k)$, i.e., they have to eliminate one incorrect answer among all 2^k possible strings. Trivial protocol is to compute one instance $f(x_i, y_i)$ and output a string where the i-th entry is $1 - f(x_i, y_i)$. In Chapter 3, we study communication complexity of the elimination problem when f is the Greater-Than function, and we discuss that the trivial protocol is optimal for a large class of functions (of a small discrepancy).

Relations of Models and Security. Since there are many computational models, other questions arise, which ask what are the relations between those

models. Considering two specific models, which one is more powerful? If we have an efficient algorithm in one model, do we have also an efficient algorithm in another model? And many others. As mentioned above there is such a relation between data structures and circuits (e.g. [100, 101, 102, 28, 35]), and between the communication model and data structures [73]. Another interesting connection, which we use in Chapter 5, is Barrington's theorem [13] that characterize the circuit class NC^1 by branching programs.

An important aspect of computation is security. A standard cryptographic problem is how to secure communication in such a way that no eavesdropper can recover the messages [57]. Further, how to convince somebody we have a proof of some claim without telling him the actual proof (so-called zero-knowledge proofs [8]); how to make a computation oblivious, thus making it impossible to extract any information about data from memory accesses, e.g. [86, 43]; how to share a secret among many parties in such a way that the secret can be revealed only if all parties cooperate [95]; and many others. In this thesis, we study a problem where two parties have private inputs, say x and y in $\{0,1\}^n$, and they want to compute a value $f(x,y)$ for a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. However, they do not want to reveal any information about their inputs to the other party (except for what is implied by the output value $f(x,y)$). This problem was for example considered by Feige et al. [37]. There is interest in a model for such protocols where the inputs are represented by cards and the computation is done by moving, turning, and shuffling the cards. The study of such protocols was initiated by den Boer [33] and since then a solid theory of card-based protocols was built, e.g. [30, 79, 76]. In Chapter 5, we study connections between card-based protocols and other computational models, like circuits and Turing machines, in particular, we identify a class of card-based protocols that corresponds to NC^1 circuits.

Organization of the Thesis. In this thesis, we study several complexity questions, which were indicated above. In Chapter 1, we give necessary notions of information theory, which are used in several chapters. In Chapter 2, we provide an unconditional polynomial lower bound for semi-adaptive dynamic data structures computing functions of large corruption bound. In Chapter 3, we provide an unconditional lower bound for the elimination problem in the communication model. In Chapter 4, we provide a conditional lower bound (based on Network Coding Conjecture [70]) for certain static data structures. In Chapter 5, we study complexity of card-based protocols.

This thesis is based on the following papers (the citation references are into List of Publications):

Chapter 2: Lower Bounds for Semi-adaptive Data Structures via Corruption. P. Dvořák, and B. Loff [5].

Chapter 3: Lower Bounds for Elimination via Weak Regularity. A. Chattopadhyay, P. Dvořák, M. Koucký, B. Loff, and S. Mukhopadhyay [12].

Chapter 4: Data Structures Lower Bounds and Popular Conjectures. P. Dvořák, M. Koucký, K. Král, and V. Slívová [2].

Chapter 5: Barrington Plays Cards: The Complexity of Card-Based Protocols. P. Dvořák, M. Koucký [3].

1. Preliminaries

In this section, we provide necessary notions of information theory that are used in this thesis. For more details, we refer to the book by Cover and Thomas [29].

We use the following notational convention. Sets are denoted by uppercase letters, such as X and Y. Random variables are denoted by uppercase boldface letters, such as $\mathbf{X}$ and $\mathbf{Y}$. By $\mu_{\mathbf{X}}$ we denote a distribution of the random variable $\mathbf{X}$. Thus, if $\mathbf{X}$ and $\mathbf{Y}$ are distributed by a joint distribution $\mu_{\mathbf{X},\mathbf{Y}}$, then the distributions $\mu_{\mathbf{X}}$ and $\mu_{\mathbf{Y}}$ are marginal distributions of $\mathbf{X}$ and $\mathbf{Y}$. By $\mu_{\mathbf{X}}(x)$ we denote the probability $\Pr[\mathbf{X} = x]$ and by $\mu_{\mathbf{X}}(x \mid \mathcal{E})$ we denote the probability of $\mathbf{X} = x$ conditioned on an event $\mathcal{E}$, i.e., $\Pr[\mathbf{X} = x \mid \mathcal{E}]$. For a distribution μ, we let $supp(\mu)$ denote the *support* of μ, i.e., the set of x with $\mu(x) > 0$. Let $\mathbf{X}$ be a random variable. The *entropy* of $\mathbf{X}$ is

$$H(\mathbf{X}) = \sum_x \mu_{\mathbf{X}}(x) \cdot \log \frac{1}{\mu_{\mathbf{X}}(x)}.$$

Informally, the entropy of $\mathbf{X}$ measures how many bits in expectation we need to encode a sample of $\mathbf{X}$. The conditional entropy $H(\mathbf{X} \mid \mathbf{Y})$ for two random variables $\mathbf{X}$ and $\mathbf{Y}$ is defined as

$$\begin{aligned} H\Big(\mathbf{X} \mid \mathbf{Y}\Big) &= \sum_y \mu_{\mathbf{Y}}(y) \cdot H\Big(\mathbf{X} \mid \mathbf{Y} = y\Big) \\ &= \sum_y \mu_{\mathbf{Y}}(y) \sum_x \mu_{\mathbf{X}}\Big(x \mid \mathbf{Y} = y\Big) \cdot \log \frac{1}{\mu_{\mathbf{X}}\Big(x \mid \mathbf{Y} = y\Big)}. \end{aligned}$$

With the entropy, we can define the mutual information $I(\mathbf{X} : \mathbf{Y})$ of two random variables.

$$I\Big(\mathbf{X} : \mathbf{Y}\Big) = H(\mathbf{X}) - H\Big(\mathbf{X} \mid \mathbf{Y}\Big)$$

Informally, the mutual information of $\mathbf{X}$ and $\mathbf{Y}$ measures how many bits we know about unknown sample of $\mathbf{X}$ if we know a sample of $\mathbf{Y}$ – or vice versa; it is not clear from the definition, however it holds that $I(\mathbf{X} : \mathbf{Y}) = I(\mathbf{Y} : \mathbf{X})$. Similarly to the entropy, we can define the conditional mutual information

$$I\Big(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z}\Big) = \sum_z \mu_{\mathbf{Z}}(z) \cdot I\Big(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z} = z\Big).$$

Let μ, μ' be two distributions such that $supp(\mu) \subseteq supp(\mu')$, then the *Kullback-Leibler divergence* of μ from μ' is

$$D_{KL}\Big(\mu \parallel \mu'\Big) = \sum_{x \in supp(\mu)} \mu(x) \cdot \log \frac{\mu(x)}{\mu'(x)}.$$

A product $\mu \times \mu'$ of two distributions μ and μ' is such distribution over $supp(\mu) \times supp(\mu')$ that for each pair $(x, y) \in supp(\mu) \times supp(\mu')$ holds that

$$\mu \times \mu'(x, y) = \mu(x) \cdot \mu'(y).$$

With the Kullback-Leibler divergence, we can have an equivalent definition of the mutual information.

$$I\Big(\mathbf{X} : \mathbf{Y}\Big) = D_{KL}\Big(\mu_{\mathbf{X},\mathbf{Y}} \parallel \mu_{\mathbf{X}} \times \mu_{\mathbf{Y}}\Big).$$

Thus, the mutual information also measures how close (according to the Kullback-Leibler divergence) is a joint distribution to the product of its marginals.

Fact 1.1 (Chain Rule)**.** *For any random variables* $\mathbf{X}_1, \mathbf{X}_2, \mathbf{Y}$ *and* $\mathbf{Z}$ *holds that*

$$H\Big(\mathbf{X}_1, \mathbf{X}_2 \mid \mathbf{Z}\Big) = H\Big(\mathbf{X}_1 \mid \mathbf{Z}\Big) + H\Big(\mathbf{X}_2 \mid \mathbf{X}_2\mathbf{Z}\Big),$$
$$I\Big(\mathbf{X}_1\mathbf{X}_2 : \mathbf{Y} \mid \mathbf{Z}\Big) = I\Big(\mathbf{X}_1 : \mathbf{Y} \mid \mathbf{Z}\Big) + I\Big(\mathbf{X}_2 : \mathbf{Y} \mid \mathbf{Z}, \mathbf{X}_1\Big).$$

Since the mutual information is never negative, we have the following corollary.

Corollary 1.2. *For any random variables* $\mathbf{X}, \mathbf{Y}$ *and* $\mathbf{Z}$ *holds that*

$$I\Big(\mathbf{X} : \mathbf{Y}\Big) \leq I\Big(\mathbf{X} : \mathbf{Y}\,\mathbf{Z}\Big).$$

The ℓ_1-distance between two distributions μ and μ' is defined as

$$\Big\|\mu - \mu'\Big\|_1 = \sum_x \Big|\mu(x) - \mu'(x)\Big|.$$

We use two observations about the ℓ_1-distance.

Observation 1.3. *Let* $\mathbf{X}, \mathbf{Y}$ *be two random variables and* $\mathcal{E}_1, \ldots, \mathcal{E}_t$ *be mutually exclusive events. Then,*

$$\Big\|\mu_{\mathbf{X}} - \mu_{\mathbf{Y}}\Big\|_1 \leq \sum_{i \leq t} \Pr[\mathcal{E}_i] \cdot \Big\|\mu_{\mathbf{X}|\mathcal{E}_i} - \mu_{\mathbf{Y}|\mathcal{E}_i}\Big\|_1.$$

Observation 1.4. *Let* $\mathbf{X}$ *be a random variable taking values in a set* S*. Let* $\mathbf{U}$ *be a uniform variable over* S *and* $\pi, \pi' : S \to S$ *be permutations. Then,*

$$\Big\|\mu_{\pi(\mathbf{X})} - \mu_{\pi'(\mathbf{X})}\Big\|_1 \leq 2 \cdot \Big\|\mu_{\mathbf{X}} - \mu_{\mathbf{U}}\Big\|_1.$$

There is a relation between ℓ_1-distance and Kullback-Leibler divergence.

Fact 1.5 (Pinsker's Inequality [31])**.** *For any two distributions* μ *and* μ' *such that* $supp(\mu) \subseteq supp(\mu')$*, it holds that*

$$\Big\|\mu - \mu'\Big\|_1 \leq \sqrt{2 \cdot D_{KL}\Big(\mu \parallel \mu'\Big)}$$

By applying the Pinsker's inequality for μ' being a uniform distribution we have the following corollary.

Corollary 1.6. *Let* $\mathbf{X}$ *be a random variable taking values in a set* S*, and let* $\mathbf{U}$ *be a random variable uniformly distributed over* S*. Then,*

$$\Big\|\mu_{\mathbf{X}} - \mu_{\mathbf{U}}\Big\|_1 \leq \sqrt{2 \cdot \Big(\log|S| - H(\mathbf{X})\Big)}.$$

2. Lower Bounds for Semi-adaptive Data Structures via Corruption

2.1 Introduction

As mentioned above, the major unsolved question in the area of dynamic data structures is to prove a polynomial lower bound for $t = \max\{t_{\mathrm{u}}, t_{\mathrm{q}}\}$, where t_{u} is update time and t_{q} is query time of an adaptive dynamic data structure computing an explicit function. In this chapter, we provide an unconditional polynomial lower bound for semi-adaptive dynamic data structure computing a broad class of functions (in particular, for all functions of large corruption bound).

For example, consider the dynamic reachability problem, where we wish to maintain a directed n-vertex graph in memory, under edge insertions and deletions, while being able to answer reachability queries ("*is vertex i connected to vertex j?*"). Is it true that any scheme for the dynamic reachability problem requires $t = \Omega(n^{\delta})$, for some constant $\delta > 0$? Indeed, such a lower bound is known under various complexity-theoretic assumptions [88, 1]. Strictly speaking, these conditional lower bounds only work if the preprocessing time, which is the time taken to encode the data into memory, is also bounded. But we will ignore this distinction. The question is whether such a lower bound may be proven unconditionally.

In an influential paper [88], Mihai Pătraşcu proposed an approach to this unsolved question. He defined a data structure problem, called the *multiphase problem*. Let us represent partial functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as total functions $f' : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}$ where $f'(x,y) = *$ if $f(x,y)$ is not defined. Then associated with a partial Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}$, and a natural number $k \geq 1$, we may define a corresponding *multiphase problem of f* as the following dynamic process:

Phase I - Initialization. We are given k inputs $x_1, \ldots, x_k \in \{0,1\}^n$, and are allowed to preprocess this input in time $nk \cdot t_{\mathrm{p}}$.

Phase II - Update. We are then given another input $y \in \{0,1\}^n$, and we have time $n \cdot t_{\mathrm{u}}$ to read and update the memory locations from the data structure constructed in Phase I.

Phase III - Query. Finally, we are given a query $i \in [k]$, we have time t_{q} to answer the question whether $f(x_i, y) = 1$. If $f(x_i, y)$ is not defined, the answer can be arbitrary.

Typically we will have $k = \mathrm{poly}(n)$. Let us be more precise, and consider randomized solutions to the above problem.

Scheme for the multiphase problem of f. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}$ be a partial Boolean function. A *scheme* for the multiphase problem of

f with preprocessing time t_{p}, update time t_{u} and query time t_{q} is a triple

$$\mathcal{D} = \Big(E, \{U_y\}_{y\in\{0,1\}^n}, \{Q_i\}_{i\in[k]}\Big),$$

where:

- $E : (\{0,1\}^n)^k \to (\{0,1\}^w)^s$ maps the input x to the memory contents $E(x)$, where each of the s memory locations holds w bits. The function E must be computed in time $nk \cdot t_{\mathrm{p}}$ by a Random-Access Machine.
- For each $y \in \{0,1\}^n$, $U_y : (\{0,1\}^w)^s \to (\{0,1\}^w)^u$ is a decision-tree of depth $\leq n \cdot t_{\mathrm{u}}$, which reads $E(x)$ and produces a sequence $U_y(E(x))$ of u *updates*.
- For each $i \in [k]$, $Q_i : (\{0,1\}^w)^s \times (\{0,1\}^w)^u \to \{0,1\}$ is a decision-tree of depth $\leq t_{\mathrm{q}}$.
- For all $x \in (\{0,1\}^n)^k$, $y \in \{0,1\}^n$, and $i \in [k]$,

$$f(x_i, y) \neq * \implies Q_i\Big(E(x), U_y(E(x))\Big) = f(x_i, y).$$

Note that the functions E, U_y's and Q_i's consequently correspond to the phases of the multiphase problem. In a *randomized scheme* for the multiphase problem of f, each U_y and Q_i are distributions over decision trees, and the scheme can make an error (outputs a wrong answer) only with a small probability. Formally, it must hold that for all $x \in (\{0,1\}^n)^k$, $y \in \{0,1\}^n$, and $i \in [k]$,

$$f(x_i, y) \neq * \implies \Pr_{Q_i, U_y}\Big[Q_i\Big(E(x), U_y(E(x))\Big) = f(x_i, y)\Big] \geq 1 - \varepsilon.$$

The value ε is called the *error probability* of the scheme.

Remark. In the usual way of defining the update phase (functions U_y's), we have a read/write decision-tree U_y which changes the very same cells that it reads. But when $w = \Omega(\log s)$, this can be seen to be equivalent, up to constant factors, to the definition we present here, where we have a decision-tree U_y that writes the updates on a separate location. In order to simulate a scheme that uses a read/write decision tree, we may use a hash table with $O(1)$ worst-case lookup time, such as cuckoo hashing. Then, we have a read-only decision-tree $U'_y(E(x))$ whose output is the hash table containing all the $i \in [s]$ which were updated by $U_y(E(x))$, associated with their final value in the execution of $U_y(E(x))$. Note that the hash table itself is static.

All our results will hold even if a decision tree Q_i is allowed to depend arbitrarily on x_i, as the scheme was defined by Ko and Weinstein [64] in that way. However, this makes for a less natural model, so we omit this from the definitions.

Pătraşcu [88] considered the multiphase problem where $f = \textsc{Disj}$ is the Disjointness function:

$$\textsc{Disj}(x, y) = \begin{cases} 0 & \text{if there exists } i \in [n] \text{ such that } x_i = y_i = 1 \\ 1 & \text{otherwise} \end{cases}$$

He conjectured that any scheme for the multiphase problem of Disj will necessarily have $\max\{t_{\mathrm{p}}, t_{\mathrm{u}}, t_{\mathrm{q}}\} \geq n^{\delta}$ for some constant $\delta > 0$.

Pătraşcu shows that such lower bounds for the multiphase problem of DISJ would imply polynomial lower bounds for various dynamic data structure problems. For example such lower bounds would imply that dynamic reachability requires $t = \Omega(n^\delta)$. He also shows that these lower bounds hold under the assumption that there is no sub-quadratic algorithm for the 3SUM problem – given n numbers, decide if there are 3 of them that sum to 0.

Finally, Pătraşcu then defines a 3-player Number-On-Forehead (NOF) communication game, such that lower bounds on this game imply matching lower bounds for the multiphase problem. The game associated with a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is as follows:

1. Alice is given $x_1, \ldots, x_k \in \{0,1\}^n$ and $i \in [k]$, Bob gets $y \in \{0,1\}^n$ and $i \in [k]$ and Charlie gets $x_1, \ldots, x_k$ and y.

2. Charlie sends a private message of ℓ_1 bits to Bob and then he is silent.

3. Alice and Bob communicate ℓ_2 bits and want to compute $f(x_i, y)$.

Again, the steps of this communication game correspond to the phases of the multiphase problem. Pătraşcu [88] conjectured that if ℓ_1 is $o(k)$, then ℓ_2 has to be bigger than the communication complexity of f. However, this conjecture turned out to be false. The randomized communication complexity of DISJ is $\Omega(n)$ [91, 52, 11], but Chattopadhyay et al. [23] constructed a protocol for $f = \text{DISJ}$ where both $\ell_1, \ell_2 = O(\sqrt{n} \cdot \log k)$. They further show that any randomized scheme in the above model can be derandomized. The conjecture remains that if $\ell_1 = o(k)$, then ℓ_2 has to be larger than the maximum distributional communication complexity of f under a product distribution. This is $\tilde{\Theta}(\sqrt{n})$ for Disjointness [10].

Thus, the above communication model is more powerful than it appears at first glance. However, a recent paper by Ko and Weinstein [64] succeeds in proving lower bounds for a simpler version of the multiphase problem, which translate to lower bounds for a restricted class of dynamic data structure schemes. They manage to prove a lower bound of $\Omega(\sqrt{n})$ for the simpler version of the multiphase problem which is associated with the Disjointness function $f = \text{DISJ}$. We generalize their result:

- We generalize their lower bound to any function f having large complexity according to the smooth corruption bound, under a product distribution. Disjointness is such a function [10], but so is Inner Product [65], Gap Orthogonality, and Gap Hamming Distance [96].

- The new lower-bounds we obtain (for Inner-product, Gap Orthogonality, and Gap Hamming Distance) are stronger – $\Omega(n)$ instead of the lower-bound $\Omega(\sqrt{n})$ for Disjointness. As far as was known before our result, it could well have been that every function had a scheme for the simpler version of the multiphase problem using only $O(\sqrt{n})$ communication.

- Ko and Weinstein derived their lower bound via a cut-and-paste lemma which works specifically for Disjointness. This cut-and-paste lemma is a more robust version of the one appearing in a paper by Bar-Yossef et al. [11], made to work not only for protocols, where the inputs $\mathbf{X}$ and $\mathbf{Y}$ are independent given the transcript $\mathbf{Z}$ of the protocol, but also for random-variables

that are "protocol-like", namely any $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ where $I(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z})$ is close to 0. Instead, we directly derive the existence of a large nearly-monochromatic rectangle, from the existence of such protocol-like random variables, which is what then allows us to use the smooth corruption bound. This result is a core technical contribution of this chapter and may be of independent interest.

All of the above lower bounds will be shown to hold also for randomized schemes, and not just for deterministic schemes.

2.1.1 Semi-adaptive Multiphase Problem

Let us provide rigorous definitions.

Definition 2.1 (Semi-adaptive random data structure [64])**.** *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}$ be a partial function. A scheme $\mathcal{D} = \Big(E, \{U_y\}_{y\in\{0,1\}^n}, \{Q_i\}_{i\in[k]}\Big)$ for the multiphase problem of f is called* semi-adaptive *if any path on the decision-tree*

$$Q_i : \Big(\{0,1\}^w\Big)^s \times \Big(\{0,1\}^w\Big)^u \to \{0,1\}$$

first queries the first part of the input (the $E(x)$ part), and then queries the second part of the input (the $U(E(x))$ part). If $\mathcal{D}$ is randomized, then this property must hold for every randomized choice of Q_i.

We point out that the reading of the cells in each part is completely adaptive. The restriction is only that the data structure cannot read cells of $E(x)$ if it already started to read cells of $U(E(x))$. Ko and Weinstein state their result for deterministic data structures, i.e., $\varepsilon = 0$, thus the data structure always returns the correct answer.

Theorem 2.2 (Theorem 4.9 of Ko and Weinstein [64])**.** *Let $k \geq \omega(n)$. Any semi-adaptive deterministic data structure that solves the multiphase problem of the Disj function, must have either $t_{\mathrm{u}} \cdot n \geq \Omega(k/w)$ or $t_{\mathrm{q}} \geq \Omega(\sqrt{n}/w)$.*

To prove the lower bound they reduce the semi-adaptive data structure into a low correlation random process.

Theorem 2.3 (Reformulation of Lemma 4.1 of Ko and Weinstein [64])**.** *Let $\mathbf{X}_1, \ldots, \mathbf{X}_k$ be random variables over $\{0,1\}^n$ and each of them is independently distributed according to the same distribution μ_1 and let $\mathbf{Y}$ be a random variable over $\{0,1\}^n$ distributed according to μ_2 (independently of $\mathbf{X}_1, \ldots, \mathbf{X}_k$). Let $\mathcal{D}$ be a randomized semi-adaptive scheme for the multiphase problem of a partial function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}$ with error probability bounded by ε. Then, for any $p \leq o(k)$ there is a random variable $\mathbf{Z} \in \{0,1\}^m$ and $i \in [k]$ such that:*

1. $\Pr\Big[f(\mathbf{X}_i, \mathbf{Y}) \neq *,\ \mathbf{Z}_m \neq f(\mathbf{X}_i, \mathbf{Y})\Big] \leq \varepsilon$.
2. $I\Big(\mathbf{X}_i : \mathbf{Y}\,\mathbf{Z}\Big) \leq t_{\mathrm{q}} \cdot w + o(t_{\mathrm{q}} \cdot w)$.
3. $I\Big(\mathbf{Y} : \mathbf{Z}\Big) \leq t_{\mathrm{q}} \cdot w$.
4. $I\Big(\mathbf{X}_i : \mathbf{Y} \mid \mathbf{Z}\Big) \leq O\Big(\frac{t_{\mathrm{u}} \cdot n \cdot w}{p}\Big)$.

The random variable $\mathbf{Z}$ consists of some $\mathbf{X}_j$'s and transcripts of query phases of $\mathcal{D}$ for some $j \in [k]$. The theorem can be interpreted as saying that the last bit of $\mathbf{Z}$ predicts $f(\mathbf{X}_i, \mathbf{Y})$, $\mathbf{Z}$ has little information about $\mathbf{X}_i$ and $\mathbf{Y}$, and the triple $(\mathbf{X}_i, \mathbf{Y}, \mathbf{Z})$ is "protocol-like", in the sense that $\mathbf{X}_i$ and $\mathbf{Y}$ are close to being independent given $\mathbf{Z}$. Ko and Weinstein [64] proved Theorem 2.3 for the deterministic schemes for the DISJ function and in the case where $\mu_1 = \mu_2$. However, their proof actually works for any (partial) function f and any two, possibly distinct distributions μ_1 and μ_2. Moreover, their proof also works for randomized schemes. The resulting statement for randomized schemes for any function f is what we have given above. To complete the proof of their lower bound, Ko and Weinstein proved that if we set p (and k) large enough so that $I(\mathbf{X}_i : \mathbf{Y} \mid \mathbf{Z}) \leq o(1)$ then such random variable $\mathbf{Z}$ cannot exist when f is the DISJ function. It is this second step that we generalize.

Let $f : X \times Y \to \{0, 1, *\}$ be a partial function and μ be a distribution over $X \times Y$. A set $R \subseteq X \times Y$ is a *rectangle* if there exist sets $A \subseteq X$ and $B \subseteq Y$ such that $R = A \times B$. For $b \in \{0, 1\}$ and $0 \leq \rho \leq 1$, we say the rectangle R is *ρ-error b-monochromatic for f under μ* if

$$\mu\Big(R \cap f^{-1}(1-b)\Big) \leq \rho \cdot \mu(R).$$

For $0 \leq \alpha \leq \frac{1}{2}$, the distribution μ is α-balanced according to f if $\mu(f^{-1}(0))$, $\mu(f^{-1}(1)) \geq \alpha$. We will prove that the existence of a random variable $\mathbf{Z}$ given by Theorem 2.3 implies that, for any $b \in \{0, 1\}$, any balanced product distribution μ and any function g which is "close" to f, there is a large (according to μ) ρ-error b-monochromatic rectangle for g in terms of t_{q}. This technique is known as smooth corruption bound [14, 21] or smooth rectangle bound [49]. We denote the smooth corruption bound of f as $\mathsf{scb}_{\mu}^{\rho,\lambda}$. Informally, $\mathsf{scb}_{\mu}^{\rho,\lambda}(f) \geq s$ if there is $b \in \{0, 1\}$ and a partial function $g : X \times Y \to \{0, 1, *\}$ which is close (measured by the parameter $\lambda \in \mathbb{R}$) to f such that any ρ-error b-monochromatic rectangle $R \subseteq X \times Y$ for g has size (under μ) at most 2^{-s}. We will define smooth corruption bound formally in the next section. Thus, if we use Theorem 2.3 as a black box we generalize Theorem 2.2 for any function of large corruption bound.

Theorem 2.4. *Let $\lambda, \tilde{\varepsilon}, \tilde{\alpha} \geq 0$ such that $\alpha \geq 2\varepsilon$ for $\varepsilon = \tilde{\varepsilon} + \lambda, \alpha = \tilde{\alpha} - \lambda$. Let μ be a product distribution over $\{0, 1\}^n \times \{0, 1\}^n$ such that μ is $\tilde{\alpha}$-balanced according to a partial function $f : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1, *\}$. Any semi-adaptive randomized scheme for the multiphase problem of f, with error probability bounded by $\tilde{\varepsilon}$, must have either $t_{\mathrm{u}} \cdot n \geq \Omega(k/w)$, or*

$$t_{\mathrm{q}} \cdot w \geq \Omega\left(\alpha \cdot \mathsf{scb}_{\mu}^{O(\varepsilon/\alpha),\lambda}(f)\right).$$

We point out that Ω and O in the bound given above hide absolute constants independent of α, ε, and λ.

As a consequence of the main result (Theorem 2.4), and of previously-known bounds for the corruption, we are able to show new lower-bounds for $t_{\mathrm{q}} = \Omega(\frac{n}{w})$ against semi-adaptive schemes for the multiphase problem of the Inner Product, Gap Orthogonality and Gap Hamming Distance functions (where the gap is $\sqrt{n}$). These lower-bounds hold assuming that $t_{\mathrm{u}} = o(\frac{k}{wn})$. They follow from the small discrepancy of Inner Product, and from a bound shown by Sherstov for the corruption of Gap Orthogonality, followed by a reduction to Gap Hamming Distance

[96]. This result also gives an alternative proof of the same lower bound proven by Ko and Weinstein [64], for the Disjointness function, of $t_{\mathrm{q}} = \Omega(\frac{\sqrt{n}}{w})$. This follows from the bound for the corruption of Disjointness under a product distribution, shown by Babai et al. [10].

Organization. In Section 2.2 we give basic definitions from communication complexity. The proof of Theorem 2.4 appears in Section 2.3. The various applications appear in Section 2.4.

2.2 Preliminaries

We now formally define the smooth corruption bound and related measures from communication complexity and refer to the book by Kushilevitz and Nisan [65] for more details. Let $f : X \times Y \to \{0, 1, *\}$ be a partial function, where $f(x, y) = *$ means f is not defined on (x, y). Let μ be a distribution over $X \times Y$. We say that f is λ-close to a partial function $g : X \times Y \to \{0, 1, *\}$ under μ if

$$\Pr_{(x,y)\sim\mu}\Big[f(x, y) \neq g(x, y)\Big] \leq \lambda.$$

For $b \in \{0, 1\}$, $\rho \in [0, 1]$, let

$$\mathcal{R}_{\mu}^{\rho,b}(f) = \Big\{R \subseteq X \times Y \text{ rectangle} \mid \mu\Big(R \cap f^{-1}(1-b)\Big) \leq \rho \cdot \mu\Big(R\Big)\Big\}$$

be the set of ρ-error b-monochromatic rectangles for f under μ. The complexity measure mono quantifies how large almost b-monochromatic rectangles can be for both $b \in \{0, 1\}$:

$$\mathsf{mono}_{\mu}^{\rho}(f) = \min_{b\in\{0,1\}} \max_{R\in\mathcal{R}_{\mu}^{\rho,b}(f)} \mu(R)$$

Using mono we can define the *corruption bound* cb and the *smooth corruption bound* scb of a function as

$$\mathsf{cb}_{\mu}^{\rho}(f) = \log \frac{1}{\mathsf{mono}_{\mu}^{\rho}(f)},$$
$$\mathsf{scb}_{\mu}^{\rho,\lambda}(f) = \max_{g\text{: } \lambda\text{-close to } f \text{ under } \mu} \mathsf{cb}_{\mu}^{\rho}(g).$$

Thus, if $\mathsf{scb}_{\mu}^{\rho,\lambda}(f) \geq s$ then there is a $b \in \{0, 1\}$ and a function g which λ-close to f under μ such that for any ρ-error b-monochromatic rectangle for g under μ it holds that $\mu(R) \leq 2^{-s}$.

The notion $\mathsf{mono}_{\mu}^{\rho}$ is related to the *discrepancy* of a function:

$$\mathsf{disc}_{\mu}(f) = \max_{R \text{ : rectangle of } X\times Y} \left|\mu\Big(R \cap f^{-1}(0)\Big) - \mu\Big(R \cap f^{-1}(1)\Big)\right|.$$

It is easy to see that for a total function f holds that $\mathsf{disc}_{\mu}(f) \geq (1-2\rho)\cdot\mathsf{mono}_{\mu}^{\rho}(f)$ for any ρ. Thus, Theorem 2.4 will give us lower bounds also for functions of small discrepancy.

Remark. In Razborov's paper where an $\Omega(n)$ lower-bound for disjointness is first proven [91], the (implicitly given) definition of a ρ-error b-monochromatic rectangle is

$$\mu\Big(R \cap f^{-1}(1-b)\Big) \leq \rho \cdot \mu\Big(R \cap f^{-1}(b)\Big).$$

Later, a strong direct product theorem for the corruption (under product distributions) was proven by Beame et al. [14], which uses instead the condition that

$$\mu\Big(R \setminus f^{-1}(b)\Big) \leq \rho \cdot \mu(R).$$

The definition we present above comes from Sherstov [96], where the condition is

$$\mu\Big(R \cap f^{-1}(1-b)\Big) \leq \rho \cdot \mu(R).$$

Thus, we have three different definitions of ρ-error b-monochromatic rectangle, and thus three different corruption bounds. Now, if the distribution μ is supported on the domain of f, all these three definitions result in (roughly) equivalent complexity measures. But if μ attributes some mass to inputs where f is undefined (which is sometimes useful if μ is a product distribution and f is a partial function, as in our case), then the definitions are no longer equivalent. Our lower bound will hold for any of the definitions, but the proof is somewhat simpler for the definition used in Sherstov's paper [96], which is the only corruption-based lower bound we use, where μ attributes mass to undefined inputs.

2.3 Finding Large Almost Monochromatic Rectangles

In this section, we prove Theorem 2.4, i.e., we prove that any function g, which is λ-close to f, has a large almost monochromatic rectangle for both $b = 0$ and $b = 1$. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}$ be a partial function. Suppose there is a semi-adaptive random scheme $\mathcal{D}$ for the multiphase problem of f with error probability bounded by $\tilde{\varepsilon}$ such that $t_{\mathrm{u}} \cdot n \leq o(k/w)$. Let $\mu = \mu_{\mathbf{X},\mathbf{Y}}$ be a product distribution over $\{0,1\}^n \times \{0,1\}^n$ of two distribution μ_1 and μ_2, such that μ is $\tilde{\alpha}$-balanced according to f. Let $b \in \{0,1\}$ and $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}$ be a partial function which is λ-close to f under μ. We will prove there is a large almost b-monochromatic rectangle for g.

Let $\mathbf{X}_1, \ldots, \mathbf{X}_k$ be independent random variables each of them distributed according to μ_1 and $\mathbf{Y}$ be an independent random variable distributed according to μ_2. Let the random variable $\mathbf{Z} \in \{0,1\}^m$ and the index $i \in [k]$ be given by Theorem 2.3 applied to the random variables $\mathbf{X}_1, \ldots, \mathbf{X}_k, \mathbf{Y}$ and the function f. Thus, we will work with the the joint distribution of $(\mathbf{X}_1, \ldots, \mathbf{X}_k, \mathbf{Y}, \mathbf{Z})$. We will then need to keep in mind that $\mu_{\mathbf{Z}}$ is the $\mathbf{Z}$-marginal of the joint distribution of $(\mathbf{X}_1, \ldots, \mathbf{X}_k, \mathbf{Y}, \mathbf{Z})$. For simplicity we denote $\mathbf{X} = \mathbf{X}_i$.

By $f(\mathbf{X}, \mathbf{Y}) \neq^* \mathbf{Z}_m$ we denote the event that the random variable $\mathbf{Z}_m$ gives us the wrong answer on an input from the support of f, i.e. $f(\mathbf{X}, \mathbf{Y}) \neq *$ and $f(\mathbf{X}, \mathbf{Y}) \neq \mathbf{Z}_m$ hold simultaneously. By Theorem 2.3 we know that

$$\Pr\Big[f(\mathbf{X}, \mathbf{Y}) \neq^* \mathbf{Z}_m\Big] \leq \tilde{\varepsilon}.$$

Since f and g are λ-close under μ, we have that μ is still balanced according to g and $g(\mathbf{X},\mathbf{Y}) \neq^* \mathbf{Z}_m$ with small probability, as stated in the next observation.

Observation 2.5. *Let $\alpha = \tilde{\alpha} - \lambda$ and $\varepsilon = \tilde{\varepsilon} + \lambda$. For the function g it holds that*

1. *The distribution $\mu_{\mathbf{X},\mathbf{Y}}$ is α-balanced according to g.*
2. $\Pr\Big[g(\mathbf{X},\mathbf{Y}) \neq^* \mathbf{Z}_m\Big] \leq \varepsilon$.

Proof. Let $b' \in \{0,1\}$. We will bound $\mu\Big(g^{-1}(b')\Big)$.

$$\begin{aligned}\tilde{\alpha} \leq \Pr\Big[f(\mathbf{X},\mathbf{Y}) = b'\Big] &= \Pr\Big[f(\mathbf{X},\mathbf{Y}) = b', f(\mathbf{X},\mathbf{Y}) = g(\mathbf{X},\mathbf{Y})\Big] \\ &\quad + \Pr\Big[f(\mathbf{X},\mathbf{Y}) = b', f(\mathbf{X},\mathbf{Y}) \neq g(\mathbf{X},\mathbf{Y})\Big] \\ &\leq \Pr\Big[g(\mathbf{X},\mathbf{Y}) = b'\Big] + \lambda.\end{aligned}$$

Thus, by rearranging we get $\mu(g^{-1}(b')) \geq \tilde{\alpha} - \lambda = \alpha$. The proof of the second bound is similar:

$$\begin{aligned}\Pr\Big[g(\mathbf{X},\mathbf{Y}) \neq^* \mathbf{Z}_m\Big] &= \Pr\Big[f(\mathbf{X},\mathbf{Y}) \neq^* \mathbf{Z}_m, f(\mathbf{X},\mathbf{Y}) = g(\mathbf{X},\mathbf{Y})\Big] \\ &\quad + \Pr\Big[g(\mathbf{X},\mathbf{Y}) \neq^* \mathbf{Z}_m, f(\mathbf{X},\mathbf{Y}) \neq g(\mathbf{X},\mathbf{Y})\Big] \leq \tilde{\varepsilon} + \lambda = \varepsilon.\end{aligned}$$

□

Let c be the bound on $I(\mathbf{X} : \mathbf{Y}\,\mathbf{Z})$ and $I(\mathbf{Y} : \mathbf{Z})$ given by Theorem 2.3. Since $I(\mathbf{X} : \mathbf{Z}) \leq I(\mathbf{X} : \mathbf{Y}\,\mathbf{Z})$, we have $I(\mathbf{X} : \mathbf{Z}), I(\mathbf{Y} : \mathbf{Z}) \leq t_{\mathrm{q}} \cdot w + o(t_{\mathrm{q}} \cdot w) = c$. We will prove that if we assume that $t_{\mathrm{u}} \cdot n < o(k/w)$ and we choose p large enough (p of Theorem 2.3) then we can find a rectangle $R \subseteq \{0,1\}^n \times \{0,1\}^n$ such that R is $O(\varepsilon/\alpha)$-error b-monochromatic for g and $\mu(R) \geq \frac{1}{2^{c'}}$ for $c' = O(\frac{t_{\mathrm{q}} \cdot w}{\alpha})$. Thus, we have $\mathsf{mono}_{\mu}^{O(\varepsilon/\alpha)}(g) \geq 2^{-c'}$ and consequently

$$\mathsf{scb}_{\mu}^{O(\varepsilon/\alpha),\lambda}(f) \leq O\left(\frac{t_{\mathrm{q}} \cdot w}{\alpha}\right).$$

By rearranging, we get the bound of Theorem 2.4.

Let us sketch the proof of how we can find such a rectangle R. Let $\mu_{\mathbf{X}|z}$ be a distribution of $\mathbf{X}$ conditioned on $\mathbf{Z} = z$ and similarly for distributions $\mu_{\mathbf{Y}|z}$ and $\mu_{\mathbf{X},\mathbf{Y}|z}$. We will first fix the random variable $\mathbf{Z}$ to z such that $\mathbf{X}$ and $\mathbf{Y}$ are not very correlated conditioned on $\mathbf{Z} = z$, i.e., the joint distribution $\mu_{\mathbf{X},\mathbf{Y}|z}$ is very similar to the product distribution of the marginals $\mu_{\mathbf{X}|z} \times \mu_{\mathbf{Y}|z}$. Moreover, we will pick z in such a way the probability of error $\Pr\Big[g(\mathbf{X},\mathbf{Y}) \neq^* z_m | \mathbf{Z} = z\Big]$ is still small. Then, since $\mu_{\mathbf{X},\mathbf{Y}|z}$ is close to $\mu_{\mathbf{X}|z} \times \mu_{\mathbf{Y}|z}$, the probability of error under the latter distribution will be small as well, i.e., if $(\mathbf{X}',\mathbf{Y}') \sim \mu_{\mathbf{X}|z} \times \mu_{\mathbf{Y}|z}$, then $\Pr\Big[g(\mathbf{X}',\mathbf{Y}') \neq^* z_m\Big]$ will also be small. Finally, we will find subsets $A \subseteq \mathit{supp}(\mu_{\mathbf{X}|z})$, $B \subseteq \mathit{supp}(\mu_{\mathbf{Y}|z})$ of large mass (under the original distributions μ_1 and μ_2), while keeping the probability of error on the rectangle $R = A \times B$ sufficiently small.

Let us then proceed to implement this plan. Let $\beta = \alpha - \varepsilon$. We will show that β is a lower bound for the probability that $\mathbf{Z}_m$ is equal to b. Let γ be the bound on $I(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z})$ given by Theorem 2.3, i.e., $I(\mathbf{X}_i : \mathbf{Y} \mid \mathbf{Z}) \leq \gamma = O(\frac{t_{\mathrm{u}} \cdot n \cdot w}{p})$.

Lemma 2.6. *There exists $z \in supp(\mu_{\mathbf{Z}})$ such that*

1. $z_m = b$.
2. $I\Big(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z} = z\Big) \leq \frac{5}{\beta} \cdot \gamma$.
3. $D_{KL}\Big(\mu_{\mathbf{X}|z} \parallel \mu_{\mathbf{X}}\Big), D_{KL}\Big(\mu_{\mathbf{Y}|z} \parallel \mu_{\mathbf{Y}}\Big) \leq \frac{5}{\beta} \cdot c$.
4. $\Pr\Big[g(\mathbf{X}, \mathbf{Y}) \neq^* z_m \mid \mathbf{Z} = z\Big] \leq \frac{5}{\beta} \cdot \varepsilon$.

Proof. Since μ is α-balanced according to g, we find that

$$\begin{aligned}\alpha &\leq \Pr\Big[g(\mathbf{X}, \mathbf{Y}) = b\Big] \\ &= \Pr\Big[g(\mathbf{X}, \mathbf{Y}) = b, \mathbf{Z}_m = b\Big] + \Pr\Big[g(\mathbf{X}, \mathbf{Y}) = b, \mathbf{Z}_m \neq b\Big] \leq \Pr\Big[\mathbf{Z}_m = b\Big] + \varepsilon.\end{aligned}$$

Thus, by rearranging we get $\Pr[\mathbf{Z}_m = b] \geq \alpha - \varepsilon = \beta$. By expanding the information $I(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z})$ we find

$$\gamma \geq I\Big(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z}\Big) = \mathop{\mathbb{E}}_{z \sim \mu_{\mathbf{Z}}}\left[I\Big(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z} = z\Big)\right]$$

and by the Markov's inequality we get that

$$\Pr_{z \sim \mu(\mathbf{Z})}\left[I\Big(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z} = z\Big) \geq \frac{5}{\beta} \cdot \gamma\right] \leq \frac{\beta}{5}.$$

Similarly, for the information $I(\mathbf{X} : \mathbf{Z})$:

$$c \geq I\Big(\mathbf{X}\,\mathbf{Y} : \mathbf{Z}\Big) \geq I\Big(\mathbf{X} : \mathbf{Z}\Big) = \mathop{\mathbb{E}}_{z \sim \mu_{\mathbf{Z}}}\left[D_{KL}\Big(\mu_{\mathbf{X}|z} \parallel \mu_{\mathbf{X}}\Big)\right]$$

and so

$$\Pr_{z \sim \mu_{\mathbf{Z}}}\left[D_{KL}\Big(\mu_{\mathbf{X}|z} \parallel \mu_{\mathbf{X}}\Big) \geq \frac{5}{\beta} \cdot c\right] \leq \frac{\beta}{5}.$$

The bound for $D_{KL}(\mu_{\mathbf{Y}|z} \parallel \mu_{\mathbf{Y}})$ is analogous. Let $e_z = \Pr_{\mu}\Big[g(\mathbf{X}, \mathbf{Y}) \neq^* z_m | \mathbf{Z} = z\Big]$. Then,

$$\varepsilon \geq \Pr\Big[g(\mathbf{X}, \mathbf{Y}) \neq^* \mathbf{Z}_m\Big] = \sum_{z \in supp(\mu_{\mathbf{Z}})} \mu_{\mathbf{Z}}(z) \cdot e_z = \mathop{\mathbb{E}}_{z \sim \mu_{\mathbf{Z}}}\Big[e_z\Big]$$

$$\Pr_{z \sim \mu_{\mathbf{Z}}}\left[e_z \geq \frac{5}{\beta} \cdot \varepsilon\right] \leq \frac{\beta}{5}.$$

Thus, by a union bound we may infer the existence of the sought $z \in \mu_{\mathbf{Z}}$.

□

Let us now fix $z \in supp(\mu_{\mathbf{Z}})$ from the previous lemma. Let $\mu_z = \mu_{\mathbf{X},\mathbf{Y}|z}$ be the distribution μ conditioned on $\mathbf{Z} = z$, and let $\mu'_z = \mu_{\mathbf{X}|z} \times \mu_{\mathbf{Y}|z}$ be the product of its marginals. Let S be the support of μ_z, and let $S_{\mathbf{X}}$ and $S_{\mathbf{Y}}$ be the supports of $\mu_{\mathbf{X}|z}$ and $\mu_{\mathbf{Y}|z}$, respectively, i.e., $S_{\mathbf{X}}$ and $S_{\mathbf{Y}}$ are the projections of S into input sets of Alice and Bob, respectively. Then, Pinsker's inequality will give us that μ_z and μ'_z are very close. Let $\delta = \sqrt{\frac{10}{\beta} \cdot \gamma}$.

Lemma 2.7. $\left\|\mu_z - \mu'_z\right\|_1 \leq \delta$

Proof. Indeed, by Pinsker's inequality,

$$\left\|\mu_z - \mu'_z\right\|_1 \leq \sqrt{2 \cdot D_{KL}\big(\mu_z \parallel \mu'_z\big)}.$$

The right-hand side is $\sqrt{2 \cdot D_{KL}\big(\mu_{\mathbf{X},\mathbf{Y}|z} \parallel \mu_{\mathbf{X}|z} \times \mu_{\mathbf{Y}|z}\big)}$, which by the alternative definition of mutual information equals to $\sqrt{2 \cdot I(\mathbf{X} : \mathbf{Y} \mid \mathbf{Z} = z)}$, and by Lemma 2.6 this is $\leq \sqrt{\frac{10}{\beta} \cdot \gamma} = \delta$.

□

Recall that $(\mathbf{X}', \mathbf{Y}') \sim \mu'_z$ are random variables chosen according to $\mu'_z = \mu_{\mathbf{X}|z} \times \mu_{\mathbf{Y}|z}$. Let $\varepsilon' = \frac{5}{\beta} \cdot \varepsilon + \delta$. It follows from Lemma 2.6 and Lemma 2.7 that:

Lemma 2.8. $\Pr\Big[g(\mathbf{X}', \mathbf{Y}') \neq^* z_m\Big] \leq \varepsilon'$.

Proof. We prove that

$$\left|\Pr\Big[g(\mathbf{X}, \mathbf{Y}) \neq^* z_m \mid \mathbf{Z} = z\Big] - \Pr\Big[g(\mathbf{X}', \mathbf{Y}') \neq^* z_m\Big]\right| \leq \delta.$$

Since $\Pr\Big[g(\mathbf{X}, \mathbf{Y}) \neq^* z_m \mid \mathbf{Z} = z\Big] \leq \frac{5}{\beta} \cdot \varepsilon$ by Lemma 2.6, the lemma follows. Let

$$B = \Big\{(x, y) \in S_{\mathbf{X}} \times S_{\mathbf{Y}} : g(x, y) \neq z_m, g(x, y) \neq *\Big\}.$$

$$\begin{aligned}
&\left|\Pr\Big[g(\mathbf{X}, \mathbf{Y}) \neq^* z_m \mid \mathbf{Z} = z\Big] - \Pr\Big[g(\mathbf{X}', \mathbf{Y}') \neq^* z_m\Big]\right| \\
&\quad = \Big|\sum_{(x,y) \in B} \mu_z(x, y) - \mu'_z(x, y)\Big| \\
&\quad \leq \sum_{(x,y) \in B} \Big|\mu_z(x, y) - \mu'_z(x, y)\Big| \leq \delta \text{ by the triangle inequality and Lemma 2.7}
\end{aligned}$$

□

Let $c' = \frac{5}{\beta} \cdot c$. We will prove the ratio between $\mu_{\mathbf{X}|z}(x)$ and $\mu_{\mathbf{X}}(x)$ is larger than $2^{O(c')}$ with only small probability (when $x \sim \mu_{\mathbf{X}|z}$). The same holds for $\mu_{\mathbf{Y}|z}$ and $\mu_{\mathbf{Y}}$.

Lemma 2.9.

$$\Pr_{x \sim \mu_{\mathbf{X}|z}}\Big[\mu_{\mathbf{X}|z}(x) \geq 2^{6c'} \cdot \mu_{\mathbf{X}}(x)\Big], \Pr_{y \sim \mu_{\mathbf{Y}|z}}\Big[\mu_{\mathbf{Y}|z}(y) \geq 2^{6c'} \cdot \mu_{\mathbf{Y}}(y)\Big] \leq \frac{1}{6}.$$

Proof. We prove the lemma for $\mu_{\mathbf{X}|z}$, the proof for $\mu_{\mathbf{Y}|z}$ is analogous. By Lemma 2.6 we know that $D_{KL}(\mu_{\mathbf{X}|z} \parallel \mu_{\mathbf{X}}) \leq c'$. We expand the Kullback-Leibler divergence:

$$c' \geq D_{KL}\big(\mu_{\mathbf{X}|z} \parallel \mu_{\mathbf{X}}\big) = \sum_{x \in S_{\mathbf{X}}} \mu_{\mathbf{X}|z}(x) \log \frac{\mu_{\mathbf{X}|z}(x)}{\mu_{\mathbf{X}}(x)} = \mathop{\mathbb{E}}_{x \sim \mu_{\mathbf{X}|z}}\left[\log \frac{\mu_{\mathbf{X}|z}(x)}{\mu_{\mathbf{X}}(x)}\right],$$

and then use the Markov's inequality:

$$\Pr_{x\sim\mu_{\mathbf{X}|z}}\Big[\mu_{\mathbf{X}|z}(x) \geq 2^{6c'}\cdot\mu_{\mathbf{X}}(x)\Big] = \Pr_{x\sim\mu_{\mathbf{X}|z}}\left[\log\frac{\mu_{\mathbf{X}|z}(x)}{\mu_{\mathbf{X}}(x)} \geq 6c'\right] \leq \frac{1}{6}.$$

□

We now split $S_{\mathbf{X}}$ and $S_{\mathbf{Y}}$ into buckets C^{x}_ℓ and C^{y}_ℓ (for $\ell \geq 1$), where the ℓ-th buckets are

$$C^{\mathrm{x}}_\ell = \left\{x \in S_{\mathbf{X}} \;\middle|\; \frac{(\ell-1)}{2^{c'}} < \frac{\mu_{\mathbf{X}|z}(x)}{\mu_{\mathbf{X}}(x)} \leq \frac{\ell}{2^{c'}}\right\},$$
$$C^{\mathrm{y}}_\ell = \left\{y \in S_{\mathbf{Y}} \;\middle|\; \frac{(\ell-1)}{2^{c'}} < \frac{\mu_{\mathbf{Y}|z}(y)}{\mu_{\mathbf{Y}}(y)} \leq \frac{\ell}{2^{c'}}\right\}.$$

In a bucket C^{x}_ℓ there are elements of $S_{\mathbf{X}}$ such that their probability under $\mu_{\mathbf{X}|z}$ is approximately $\frac{\ell}{2^{c'}}$-times bigger than their probability under $\mu_{\mathbf{X}}$. By Lemma 2.9, it holds that with high probability the elements $x \in S_{\mathbf{X}}, y \in S_{\mathbf{Y}}$ are in the buckets $C^{\mathrm{x}}_{\ell_1}$ and $C^{\mathrm{y}}_{\ell_2}$ for $\ell_1, \ell_2 \leq 2^{7c'}$. Thus, if we find a bucket $C^{\mathrm{x}}_{\ell_1}$ for $\ell_1 \leq 2^{7c'}$ which has probability at least $\frac{1}{2^{O(c')}}$ under $\mu_{\mathbf{X}|z}$, then it has also probability at least $\frac{1}{2^{O(c')}}$ under $\mu_{\mathbf{X}}$. The same holds also for buckets C^{y}_ℓ. In the next lemma we will show that there are buckets $C^{\mathrm{x}}_{\ell_1}$ and $C^{\mathrm{y}}_{\ell_2}$ of large probability under μ'_z such that the probability of error on $C^{\mathrm{x}}_{\ell_1} \times C^{\mathrm{y}}_{\ell_2}$ is still small.

Lemma 2.10. *There exist buckets $C^{\mathrm{x}}_{\ell_1}$ and $C^{\mathrm{y}}_{\ell_2}$ such that*

1. $1 < \ell_1, \ell_2 \leq 2^{7c'}$.
2. $\Pr\Big[\mathbf{X}' \in C^{\mathrm{x}}_{\ell_1}\Big], \Pr\Big[\mathbf{Y}' \in C^{\mathrm{y}}_{\ell_2}\Big] \geq \frac{1}{6\cdot 2^{7c'}}$.
3. $\Pr\Big[g(\mathbf{X}', \mathbf{Y}') \neq^* z_m, (\mathbf{X}', \mathbf{Y}') \in C^{\mathrm{x}}_{\ell_1} \times C^{\mathrm{y}}_{\ell_2}\Big] \leq 6\varepsilon' \cdot \Pr\Big[(\mathbf{X}', \mathbf{Y}') \in C^{\mathrm{x}}_{\ell_1} \times C^{\mathrm{y}}_{\ell_2}\Big]$.

Proof. We prove that ℓ_1, ℓ_2 exist via the probabilistic method. Let $\mathbf{L}_1$ and $\mathbf{L}_2$ be bucket indices of $\mathbf{X}'$ and $\mathbf{Y}'$, respectively. Thus, $\Pr[\mathbf{L}_1 = \ell] = \Pr[\mathbf{X}' \in C^{\mathrm{x}}_\ell]$ and $\Pr[\mathbf{L}_2 = \ell] = \Pr[\mathbf{Y}' \in C^{\mathrm{y}}_\ell]$.

Let $B_1, B_2 \subseteq L' = \{1, \ldots, 2^{7c'}\}$ be sets of indices of small probability, i.e., for $i \in \{1, 2\}$

$$B_i = \left\{\ell \in L' \;\middle|\; \Pr[\mathbf{L}_i = \ell] \leq \frac{1}{6\cdot 2^{7c'}}\right\}.$$

We will prove that with high probability we have $2^{7c'} \geq \mathbf{L}_1 > 1$ and $\mathbf{L}_1 \notin B_1$. The proof for $\mathbf{L}_2$ is analogous.

$$\Pr\Big[\mathbf{L}_1 = 1\Big] = \Pr\Big[\mathbf{X}' \in C^{\mathrm{x}}_1\Big] = \sum_{x\in C^{\mathrm{x}}_1} \mu_{\mathbf{X}|z}(x) \leq \frac{\sum_{x\in C^{\mathrm{x}}_1}\mu_{\mathbf{X}}(x)}{2^{c'}} \leq \frac{1}{2^{c'}}$$

By Lemma 2.9, we get $\Pr[\mathbf{L}_1 > 2^{7c'}] = \Pr_{x\sim\mu_{\mathbf{X}|z}}\Big[\mu_{\mathbf{X}|z}(x) \geq 2^{6c'}\cdot\mu_{\mathbf{X}}(x)\Big] \leq \frac{1}{6}$. There is only small probability that $\mathbf{L}_1$ is in B_1.

$$\Pr\Big[\mathbf{L}_1 \in B_1\Big] = \sum_{\ell\in B_1}\Pr[\mathbf{L}_1 = \ell] \leq \frac{|L'|}{6\cdot 2^{7c'}} = \frac{1}{6}$$

Thus, we have that $\mathbf{L}_i \in B_i$ or $\mathbf{L}_i = 1$ or $\mathbf{L}_i > 2^{7c'}$ with probability at most $\frac{2}{3} + \frac{2}{2^{c'}}$.

By Lemma 2.8, we have that $\Pr\Big[g(\mathbf{X}', \mathbf{Y}') \neq^* z_m\Big] \leq \varepsilon'$. By expanding the probability and by Markov's inequality we will now get the last inequality for $C^{\mathrm{x}}_{\ell_1}$ and $C^{\mathrm{y}}_{\ell_2}$. Let

$$e(\ell_1, \ell_2) = \Pr\Big[g(\mathbf{X}', \mathbf{Y}') \neq^* z_m \mid \mathbf{X}' \in C^{\mathrm{x}}_{\ell_1}, \mathbf{Y}' \in C^{\mathrm{y}}_{\ell_2}\Big].$$

We will prove there is ℓ_1 and ℓ_2 such that $e(\ell_1, \ell_2) \leq 6\varepsilon'$. This is equivalent to the third bound of the lemma. We have: $\varepsilon' \geq \Pr\Big[g(\mathbf{X}', \mathbf{Y}') \neq^* z_m\Big] = \mathbb{E}\left[e(\mathbf{L}_1, \mathbf{L}_2)\right]$ and thus, by Markov, $\Pr\Big[e(\mathbf{L}_1, \mathbf{L}_2) > 6\varepsilon'\Big] \leq \frac{1}{6}$. By a union bound we conclude that there must exist $1 < \ell_1, \ell_2 \leq 2^{7c'}$ such that $\Pr[\mathbf{L}_1 = \ell_1], \Pr[\mathbf{L}_2 = \ell_2] \geq \frac{1}{6 \cdot 2^{7c'}}$ and $e(\ell_1, \ell_2) \leq 6\varepsilon'$.

□

As a corollary, we will prove that the rectangle $C^{\mathrm{x}}_{\ell_1} \times C^{\mathrm{y}}_{\ell_2}$ (given by the previous lemma) is a good rectangle under the original distribution μ. We remark that the proof of the following corollary is the only place where we use the fact that $\mathbf{X}$ and $\mathbf{Y}$ are independent.

Corollary 2.11. *There exists a rectangle $R \subseteq S_{\mathbf{X}} \times S_{\mathbf{Y}}$ such that*

1. $\Pr\Big[(\mathbf{X}, \mathbf{Y}) \in R\Big] \geq \frac{1}{36 \cdot 2^{26c'}}$.
2. $\Pr\Big[g(\mathbf{X}, \mathbf{Y}) \neq^* z_m, (\mathbf{X}, \mathbf{Y}) \in R\Big] \leq 24\varepsilon' \cdot \Pr\Big[(\mathbf{X}, \mathbf{Y}) \in R\Big]$.

Proof. Let $R = C^{\mathrm{x}}_{\ell_1} \times C^{\mathrm{y}}_{\ell_2}$ where $C^{\mathrm{x}}_{\ell_1}$ and $C^{\mathrm{y}}_{\ell_2}$ are buckets given by Lemma 2.10. By Lemma 2.10, we get

$$\frac{1}{6 \cdot 2^{7c'}} \leq \Pr\Big[\mathbf{X}' \in C^{\mathrm{x}}_{\ell_1}\Big] = \sum_{x \in C^{\mathrm{x}}_{\ell_1}} \mu_{\mathbf{X}|z}(x) \leq \sum_{x \in C^{\mathrm{x}}_{\ell_1}} \frac{\ell_1 \cdot \mu_{\mathbf{X}}(x)}{2^{c'}} = \Pr\Big[\mathbf{X} \in C^{\mathrm{x}}_{\ell_1}\Big] \cdot \frac{\ell_1}{2^{c'}}.$$

By rearranging we get

$$\Pr\Big[\mathbf{X} \in C^{\mathrm{x}}_{\ell_1}\Big] \geq \frac{2^{c'}}{6\ell_1 \cdot 2^{7c'}} \geq \frac{1}{6 \cdot 2^{13c'}}$$

The bound for $\Pr[\mathbf{Y} \in C^{\mathrm{y}}_{\ell_2}]$ is analogous, thus we have $\Pr[(\mathbf{X}, \mathbf{Y}) \in R] \geq \frac{1}{36 \cdot 2^{26c'}}$. (Here and below, we crucially use the fact that $\mathbf{X}, \mathbf{Y}$ are given by a product distribution.) Now we prove the second bound for R. Let

$$B = \Big\{(x, y) \in R : g(x, y) \neq z_m, g(x, y) \neq *\Big\}.$$

$$\begin{aligned}
6\varepsilon' \cdot \Pr\Big[(\mathbf{X}, \mathbf{Y}) \in R\Big] \cdot \frac{\ell_1 \ell_2}{2^{2c'}} &\geq 6\varepsilon' \cdot \Pr\Big[(\mathbf{X}', \mathbf{Y}') \in R\Big] && \text{(by definition of buckets)} \\
&\geq \Pr\Big[(\mathbf{X}', \mathbf{Y}') \in B\Big] && \text{(by Lemma 2.10)} \\
&\geq \Pr\Big[(\mathbf{X}, \mathbf{Y}) \in B\Big] \cdot \frac{(\ell_1 - 1)(\ell_2 - 1)}{2^{2c'}} \\
& && \text{(by definition of buckets)}
\end{aligned}$$

Thus, by rearranging we get

$$\Pr\Big[(\mathbf{X},\mathbf{Y}) \in B] \leq 6\varepsilon' \cdot \Pr\Big[(\mathbf{X},\mathbf{Y}) \in R\Big] \cdot \frac{\ell_1\ell_2}{(\ell_1-1)(\ell_2-1)} \leq 24\varepsilon' \cdot \Pr\Big[(\mathbf{X},\mathbf{Y}) \in R\Big],$$

as $\frac{\ell_1\ell_2}{(\ell_1-1)(\ell_2-1)} \leq 4$ for $\ell_1, \ell_2 > 1$ by Lemma 2.10.

□

Now, we are ready to prove the main theorem of this chapter.

Theorem 2.4. *Let $\lambda, \tilde{\varepsilon}, \tilde{\alpha} \geq 0$ such that $\alpha \geq 2\varepsilon$ for $\varepsilon = \tilde{\varepsilon} + \lambda, \alpha = \tilde{\alpha} - \lambda$. Let μ be a product distribution over $\{0,1\}^n \times \{0,1\}^n$ such that μ is $\tilde{\alpha}$-balanced according to a partial function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}$. Any semi-adaptive randomized scheme for the multiphase problem of f, with error probability bounded by $\tilde{\varepsilon}$, must have either $t_{\mathrm{u}} \cdot n \geq \Omega(k/w)$, or*

$$t_{\mathrm{q}} \cdot w \geq \Omega\left(\alpha \cdot \mathsf{scb}_{\mu}^{O(\varepsilon/\alpha),\lambda}(f)\right).$$

Proof. Suppose that $t_{\mathrm{u}} \cdot n \leq o(k/w)$. Let R be the rectangle given by Corollary 2.11. It holds that the rectangle R is $24\varepsilon'$-error b-monochromatic for g under μ. Therefore, for the function g holds that

$$\mathsf{mono}_{\mu}^{24\varepsilon'}(g) \geq \Pr\Big[(\mathbf{X},\mathbf{Y}) \in R\Big] \geq \frac{1}{36 \cdot 2^{26c'}}. \tag{2.1}$$

We need to argue that ε' is $O(\varepsilon/\alpha)$. By definition,

$$\varepsilon' = \frac{5}{\alpha - \varepsilon} \cdot \varepsilon + \delta.$$

We recall that

$$\delta = O\left(\sqrt{\frac{t_{\mathrm{u}} \cdot n \cdot w}{p}}\right) \leq \sqrt{\frac{o(k)}{p}}.$$

Thus, we can set p to be large enough so that δ is smaller than an arbitrary constant and still $p \leq o(k)$. By the assumption we have $2\varepsilon < \alpha$. Thus, $\frac{\varepsilon}{\alpha-\varepsilon} \leq \frac{2\varepsilon}{\alpha}$ and we conclude that ε' is $O(\varepsilon/\alpha)$. Since $c' = O(\frac{t_{\mathrm{q}} \cdot w}{\alpha \cdot (1-\varepsilon)}) = O(\frac{t_{\mathrm{q}} \cdot w}{\alpha})$, we get the result by rearranging Inequality (2.1).

□

2.4 Applications

In this section we apply Theorem 2.4 to derive lower bounds for several explicit functions – Inner Product (IP), Disjointness (Disj), Gap Orthogonality (ORT) and Gap Hamming Distance (GHD):

$$\mathrm{IP}(x,y) = \sum_{i \in n} x_i \cdot y_i \mod 2,$$

$$\mathrm{GHD}_n(x,y) = \begin{cases} 1 & \text{if } \Delta_H(x,y) \geq \frac{n}{2} + \sqrt{n}, \\ 0 & \text{if } \Delta_H(x,y) \leq \frac{n}{2} - \sqrt{n}. \end{cases}$$

The function Δ_H is the Hamming Distance of two strings, i.e., $\Delta_H(x,y)$ is a number of indices $i \in [n]$ such that $x_i \neq y_i$. For $\mathrm{IP}_{\mathbb{R}}(x,y) = \sum_{i\in[n]}(-1)^{x_i+y_i}$ we define

$$\mathrm{ORT}_{n,d}(x,y) = \begin{cases} 1 & \text{if } \left|\mathrm{IP}_{\mathbb{R}}(x,y)\right| \geq 2d\cdot\sqrt{n} \\ 0 & \text{if } \left|\mathrm{IP}_{\mathbb{R}}(x,y)\right| \leq d\cdot\sqrt{n}. \end{cases}$$

The standard value for d is 1, thus we denote $\mathrm{ORT}_n = \mathrm{ORT}_{n,1}$. Note that $\Delta_H(x,y) = \frac{n-\mathrm{IP}_{\mathbb{R}}(x,y)}{2}$ and $\mathrm{IP}_{\mathbb{R}}(x,y)$ is the Inner Product of x', y' over $\mathbb{R}$ where x' and y' arise from x and y by replacing 0 by 1 and 1 by -1. We present previous results with bounds for measures of interest under hard distributions.

Theorem 2.12 ([65]). *Let μ_1 be a uniform distribution on $\{0,1\}^n \times \{0,1\}^n$. Then,*

$$\mathsf{disc}_{\mu_1}(IP) \leq \frac{1}{2^{n/2}}.$$

Theorem 2.13 (Babai et al. [10]). *Let $\rho < 1/100$ and μ_2 be a a uniform distribution over $S \times S$, where S consists of n-bit strings containing exactly $\sqrt{n}$ 1's. Then,*

$$\mathsf{mono}^{\rho}_{\mu_2}(\textsc{Disj}) \leq \frac{1}{2^{\Omega(\sqrt{n})}}.$$

Sherstov [96] provided a lower bound of communication complexity of GHD by lower bound of corruption bound of $\mathrm{ORT}_{n,\frac{1}{8}}$ following by reduction to GHD.

Theorem 2.14 (Sherstov [96]). *Let $\rho > 0$ be sufficiently small and μ_3 be a uniform distribution over $\{0,1\}^n \times \{0,1\}^n$. Then,*

$$\mathsf{cb}^{\rho}_{\mu_3}(ORT_{n,\frac{1}{8}}) \geq \rho \cdot n.$$

By this theorem and Theorem 2.4 we get a lower bound for data structures for $\mathrm{ORT}_{n,\frac{1}{8}}$. By reductions used by Sherstov [96] we also get a lower bounds for ORT and GHD.

$$\begin{aligned} \mathrm{ORT}_{n,\frac{1}{8}}(x,y) = {} & \mathrm{ORT}_{64n}\Big(x^{64}, y^{64}\Big) \\ \mathrm{ORT}_{n}(x,y) = {} & \mathrm{GHD}_{10n+15\sqrt{n}}\Big(x^{10}1^{15\sqrt{n}}, y^{10}0^{15\sqrt{n}}\Big) \\ & \wedge \neg\mathrm{GHD}_{10n+15\sqrt{n}}\Big(x^{10}0^{15\sqrt{n}}, y^{10}0^{15\sqrt{n}}\Big) \end{aligned}$$

Where s^i denote i copies of s concatenated together. Let $\mathcal{D}$ be a semi-adaptive random scheme for the multiphase problem of the presented functions with sufficiently small error probability. By the theorems presented in this section and by Theorem 2.4, we can derive the following lower bounds for $t_{\mathrm{q}} \cdot w$, assuming that $t_{\mathrm{u}} \cdot n \leq o\Big(k/w\Big)$.

Function f	Ballancedness of the hard distribution	Lower bound of $t_{\mathrm{q}} \cdot w$
IP	$\frac{1}{2}$	$\Omega(n)$
Disj	$\sim \frac{1}{e}$	$\Omega(\sqrt{n})$
ORT_n	$\Theta(1)$	$\Omega(n)$
GHD_n	N/A (lower-bound is via reduction)	$\Omega(n)$

3. Lower Bound for Elimination of Greater-Than via Weak Regularity

3.1 Introduction

In this chapter, we study a problem called *elimination*, which is closely related to the direct sum question. Let $f : \{0,1\} \times \{0,1\} \to \{0,1\}$ be a two-player boolean function, the elimination problem $elim \circ f^k$ gives Alice and Bob k n-bit strings each and asks them to find a k-bit vector z of answers to the k instances which differs from the correct answer for at least one of the instances (i.e., $z_i \neq f(x_i, y_i)$ for some i). Hence, one eliminates a single incorrect vector of answers. In the context of communication complexity, this problem was posed by Ambainis et al. [7] and further studied by Beimel et al. [15]. A trivial solution for the elimination problem is to solve one of the k instances and negate its answer. Both papers [7, 15] provided examples where one can achieve some savings, typically $\ll k$ bits of communication, and they also provide lower bounds for particular functions.

Ambainis et al. established $\Omega(n/\log n \log\log n)$ lower bound on randomized communication complexity of elimination for the Inner Product – $elim \circ \mathrm{IP}^k$ – and Disjointness – $elim \circ \textsc{Disj}^k$ – for constant k. Their result can be extended to slightly growing k, but for $k \geq \Omega(\log n)$ the lower bound becomes trivial. Beimel et al. [15] establish a general relationship, showing that the (public-coin) randomized communication complexity of elimination for f^k is lower-bounded by the (public-coin) randomized complexity of f with error roughly $\frac{1}{2} - 2^{-k}$. Due to this large allowed error, the lower bound also becomes trivial for large k.

In this chapter, we consider the problem of elimination for the Greater-Than function in the setting of (public-coin) randomized communication complexity. The Greater-Than ($\mathrm{GT} : \{0,1\} \times \{0,1\} \to \{0,1\}$) function is defined as follows:

$$\mathrm{GT}(x, y) = 1 \Leftrightarrow x \geq y,$$

where x and y are considered as n-bits numbers. In this chapter, we will prove that a trivial protocol for $elim \circ \mathrm{GT}^k$ is optimal, as randomized communication complexity of GT is $\Theta(\log n)$ [65].

Theorem 3.1. *Let $k < o(n^{1/9})$. Then,*

$$\mathsf{R}^{\varepsilon}\left(elim \circ GT^k\right) \geq \Omega(\log n).$$

Above, R stands for randomized communication complexity of a given function. In our paper [22], we actually proved a more general result, which connects the communication complexity of $elim \circ f^k$ and the discrepancy of f.

Theorem 3.2. *For any boolean function f and a distribution μ on inputs of f,*

$$\mathsf{D}^{\varepsilon}_{\mu^k}\left(elim \circ f^k\right) \geq \log \frac{1}{\mathsf{disc}_{\mu}(f)} - \log k + \log(1 - \varepsilon \cdot 2^k) - O(1).$$

Above, D stands for distributional communication complexity of a given function. Theorem 3.2 implies Theorem 3.1, as by Yao's principle [65] we have that for any boolean function $f : \{0,1\} \times \{0,1\} \to \{0,1\}$ holds that

$$\mathsf{R}^{\varepsilon}(f) = \max_{\mu} \mathsf{D}^{\varepsilon}_{\mu}(f).$$

Since the discrepancy of GT is $O(1/\sqrt{n})$ [18], Theorem 3.2 implies a lower bound of $\Omega(\log n)$ for communication complexity of the elimination of GT^k, for $k < o(n^{1/2})$. Previous results by Beimel et al. [15] yielded interesting bounds only for $k \leq \log n$. Another corollary is a lower bound of $\Omega(n)$ for elimination of IP^k (Inner Product) for k as large as exponential in n. The best-known result before our work [15] does not give any non-trivial lower bound for $k \geq n$.

To prove Theorem 3.2 we used a XOR lemma for the discrepancy by Lee et al. [69], which is considered as "heavy machinery". In this chapter, we provide a direct proof of Theorem 3.1 from the first principles of information theory. Although we get a worse bound for k (but still polynomial) than from the proof via Theorem 3.2, the elementary proof is interesting on its own and it shed some light on the structure of the elimination problem.

To prove Theorem 3.1, we identify a property of boolean functions that is closely related to the randomized communication complexity of elimination: the *weak-regularity* (this property was also used for the proof of Theorem 3.2). The proof of weak-regularity of GT designs a hard distribution for GT. Our distribution borrows ideas from the work of Viola [103] but extends them in a novel way.

Organization. In the next section, we formally define the notion of weak-regularity and prove that if a function f has small weak-regularity, then the elimination of f is a hard problem in the communication model. In Section 3.3, we prove that GT has small weak-regularity.

3.2 The Elimination Problem

To solve the elimination problem of a boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, Alice and Bob are given k strings $\bar{x} = x_1, \ldots, x_k$, and $\bar{y} = y_1, \ldots, y_k$ respectively, of n-bits each; they must then communicate in order to agree on an *output* string $\mathsf{out} \in \{0,1\}^k$, such that $\mathsf{out} \neq f^k(\bar{x}, \bar{y})$.

Other problems of a similar flavor are studied in previous works – enumeration and selection in [7], choice and agreement in [15]. However, proving lower bounds for the elimination will give the same lower bounds for these other problems, thus we will not describe the other problems in further detail.

Instead, we will focus on proving lower bounds against *randomized* protocols for the elimination problem. In this setting, we say that Alice and Bob *succeed* when they output a string $\mathsf{out} \in \{0,1\}^k$ other than $f^k(\bar{x}, \bar{y})$, as above; otherwise we say that they *made an error*. There are two related concepts of randomness in communication complexity.

Definition 3.3. *The* randomized communication complexity *of elim$\circ f^k$, denoted* $\mathsf{R}^{\varepsilon}(\mathit{elim} \circ f^k)$*, is the length of the smallest randomized protocol with shared randomness, which on every input $\bar{x}, \bar{y}$ will succeed except with* error probability $\leq \varepsilon$.

Definition 3.4. *Let μ be a distribution over $\{0,1\} \times \{0,1\}$, and $f : \{0,1\} \times \{0,1\} \to \{0,1\}$ be a boolean function. The* distributional communication complexity *of $elim \circ f^k$ over μ, denoted $\mathsf{D}^{\varepsilon}_{\mu}(elim \circ f^k)$, is the length of the smallest deterministic protocol which succeeds with error probability $\leq \varepsilon$ on inputs drawn from μ.*

3.2.1 Basic Observations

By the trivial protocol, both players can compute f for a single instance, then they can output any vector that negates f at that coordinate. Thus, clearly $\mathsf{R}^{\varepsilon}(elim \circ f^k) \leq \mathsf{R}^{\varepsilon}(f)$.

The next thing to notice is that the randomized task becomes trivial for $\varepsilon \geq 2^{-k}$. Indeed, both players can use their shared randomness to choose a uniformly random $\mathsf{out} \in \{0,1\}^k$, and outputting this out causes them to succeed with probability $1 - 2^{-k}$. Hence we will assume from this point onward that $\varepsilon \leq 2^{-k}$.

As usual, we will make use of Yao's principle [65] to prove our lower bounds. Thus to prove Theorem 3.1, we provide a hard input distribution μ for which $\mathsf{D}^{\varepsilon}_{\mu}(elim \circ f^k)$ is high. In preparation for this, let us think what it means when we say $\mathsf{D}^{\varepsilon}_{\mu}(elim \circ f^k) \leq C$. It means that we can partition the space $\{0,1\}^{nk} \times \{0,1\}^{nk}$ into $\leq 2^C$ combinatorial rectangles $R = A \times B$, and in each rectangle we will output a single $\mathsf{out} = \mathsf{out}(R) \in \{0,1\}^k$. Now, each rectangle R is itself naturally partitioned into 2^k *pieces*, one piece for each $z \in \{0,1\}^k$:

$$R^z = \Big\{(\bar{x}, \bar{y}) \in R \mid f^k(\bar{x}, \bar{y}) = z\Big\}.$$

These pieces are possibly non-rectangular and some of them might be empty. Then, if the success probability is high, it must happen that on the μ-large rectangles, R^{out} has very little mass. Indeed, the error probability is the sum of every $\mu(R^{\mathsf{out}(R)})$ over the various rectangles R induced by the protocol.

This (vague) condition is both necessary and sufficient, because if we do have a (protocol-induced) partition of $\{0,1\}^{nk} \times \{0,1\}^{nk}$ into rectangles, and on every such rectangle R there is a piece R^z with very little mass (less than $2^{-k}\mu(R)$, say), we may simply output $\mathsf{out} = z$ on this rectangle, and we will have a non-trivial protocol for elimination.

3.2.2 Regularity

A natural way of proving a lower bound would be to show that, under some carefully chosen hard distribution μ, every rectangle R gets split into pieces R^z all of which are *non-vanishing*. We may eventually come to the following natural definition:

Definition 3.5. *Let $n, k \geq 1$ be natural numbers, and $\delta \in [0,1]$; let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a boolean function, and μ be a distribution over $\{0,1\}^{nk} \times \{0,1\}^{nk}$.*

Then f is said to be δ-weakly-regular *with respect to μ if for every R and z,*

$$\mu(R^z) \geq 2^{-k}\Big(\mu(R) - \delta\Big),$$

where R ranges over combinatorial rectangles in $\{0,1\}^{nk} \times \{0,1\}^{nk}$ and z ranges over $\{0,1\}^k$.

In this definition, if $\mu(R) < \delta$ then the condition is satisfied trivially so δ bounds from below the mass of rectangles for which each R^z should have a non-trivial mass.

Theorem 3.6. *If f is δ-weakly-regular with respect to μ, then*

$$\mathsf{D}^{\varepsilon}_{\mu}\left(\mathit{elim} \circ f^k\right) \geq \log \frac{1 - \varepsilon \cdot 2^k}{\delta}.$$

Proof. Let π be an optimal protocol computing $\mathit{elim} \circ f^k$ and communicating $C = \mathsf{D}^{\varepsilon}_{\mu}(\mathit{elim} \circ f^k)$ bits. Let R range over the monochromatic rectangles induced by π. Since f is δ-weakly-regular w.r.t. μ, π's error probability is

$$\varepsilon = \sum_R \mu\left(R^{\mathsf{out}(R)}\right) \geq \sum_R 2^{-k} \cdot \left(\mu(R) - \delta\right) \geq 2^{-k} \cdot \left(1 - \delta \cdot 2^C\right).$$

By rearranging, we get the following.

$$\mathsf{D}^{\varepsilon}_{\mu}\left(\mathit{elim} \circ f^k\right) = C \geq \log \frac{1 - \varepsilon \cdot 2^k}{\delta}$$

□

Remark. The notion of *weak* regularity naturally arises from first considerations of the elimination problem. Interestingly, it is the weak version of a stronger notion, which says that f is *regular* with respect to ρ if every large rectangle R is partitioned into rectangles R^z of roughly equal mass (i.e., each R^z comprises approximately 2^{-k} fraction of all the ρ-mass of R). This strong regularity first appeared in a paper of Raz and Wigderson [90] on randomized communication complexity of Karchmer-Wigderson games [55], and also later – in disguise – in related work on simulation theorems [44].

3.3 Lower Bound for $\mathit{elim} \circ \mathrm{GT}^k$ from First Principles

In this section we prove that there is a "hard" distribution μ such that GT is $n^{-1/9}$-weakly-regular with respect to μ^k. This implies Theorem 3.1. The distribution μ on $\{0,1\}^n \times \{0,1\}^n$ is defined as follows. Let m and ℓ be integers such that $n = m\ell$ (think of $m = \sqrt{n}$). We split each n-bit output into m blocks of ℓ bits each. We set $\mathbf{X} = \mathbf{X}_1 \dots \mathbf{X}_m$, where each block $\mathbf{X}_i$ is uniformly and independently selected from the set $\mathcal{B}_\ell = \{0,1\}^\ell - \{0^\ell, 1^\ell\}$ (i.e. we forbid the all-0's and all-1's strings). Then, we pick a uniformly random block index $\mathbf{J} \in [m]$, and a uniformly-random bit $\mathbf{Z} \in \{-1, 1\}$, and set $\mathbf{Y} = \mathbf{X}_1 \dots \mathbf{X}_{\mathbf{J}-1}(\mathbf{X}_{\mathbf{J}} + \mathbf{Z})0 \dots 0$, where $\mathbf{X}_j$ is interpreted as an integer. Note that $\mathbf{X} \geq \mathbf{Y}$ (i.e., $\mathrm{GT}(\mathbf{X}, \mathbf{Y}) = 1$) if and only if $\mathbf{Z} = -1$. Let $\bar{\mu}$ denote the distribution of $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{J})$ generated by this process; then μ is the projection of $\bar{\mu}$ onto $(\mathbf{X}, \mathbf{Y})$.

Lemma 3.7. *For $x \geq 2$, it holds* $\log(2^x - 2) \geq x - \frac{1}{2^{x-2}}$.

Proof. We will prove an equivalent inequality

$$\log(2^x - 2) - \log(2^x) \geq -\frac{1}{2^{x-2}}.$$

By convexity of the exponential function we have $1 - y \geq 2^{-2y}$ for $y \in [0; \frac{1}{2}]$. Then

$$\log(2^x - 2) - \log(2^x) = \log\Big(1 - 2^{1-x}\Big) \geq \log\Big(2^{-2^{2-x}}\Big) = -\frac{1}{2^{x-2}}.$$

□

In this section, we prove the following theorem, which, together with Theorem 3.6, implies Theorem 3.1. Our proof is inspired by a paper of Viola [103] who proved lower bound on the randomized communication complexity of GT.

Theorem 3.8. *GT is $n^{-1/9}$-weakly-regular with respect to μ^k, provided $k \leq n^{1/4}$.*

Proof. Let the random variables $\mathbf{X} = \mathbf{X}^1 \dots \mathbf{X}^k$, $\mathbf{Y} = \mathbf{Y}^1 \dots \mathbf{Y}^k$, $\mathbf{J} = \mathbf{J}^1 \dots \mathbf{J}^k$ and $\mathbf{Z} = \mathbf{Z}^1 \dots \mathbf{Z}^k$ be drawn according to the distribution $\bar{\mu}^k$, by the process given above, so that $(\mathbf{X}, \mathbf{Y})$ is distributed according to μ^k.

Fix a large rectangle $R = R_1 \times R_2$ – i.e., a rectangle such that $\mu^k(R) \geq \frac{1}{n}$. Let $\mathcal{X} = (\mathcal{B}_\ell)^{mk}$ be the support of $\mathbf{X}$. Since μ^k has zero mass outside of $\mathcal{X} \times \{0,1\}^{kn}$, assume without loss of generality that $R_1 \subset \mathcal{X}$.

Let $\mathbf{W}$ denote a random variable distributed as $\mathbf{Y}$ conditioned on $\mathbf{X} \in R_1$, and $\mathbf{W}_z$ be distributed as $\mathbf{Y}$ conditioned on $\mathbf{X} \in R_1$ and $\mathbf{Z} = z$. We slightly abuse a notation and for two random variables $\mathbf{A}$ and $\mathbf{B}$ we denote $\|\mathbf{A} - \mathbf{B}\|_1$ the ℓ_1 distance of their distribution, i.e., the value $\|\mu_{\mathbf{A}} - \mu_{\mathbf{B}}\|_1$. We will prove that, for any $z \in \{0,1\}^k$,

$$\Big\|\mathbf{W} - \mathbf{W}_z\Big\|_1 \leq \frac{1}{n^{1/9}}. \tag{3.1}$$

This implies Theorem 3.8, because

$$\begin{aligned}
2^k \mu^k(R^z) &= 2^k \cdot \Pr\Big[(\mathbf{X}, \mathbf{Y}) \in R^z\Big] \\
&= 2^k \cdot \Pr\Big[\mathbf{X} \in R_1, \mathbf{Y} \in R_2, \mathbf{Z} = z\Big] \\
&= \Pr[\mathbf{X} \in R_1] \cdot \Pr\Big[\mathbf{Y} \in R_2 \mid \mathbf{X} \in R_1, \mathbf{Z} = z\Big] \\
&\geq \Pr[\mathbf{X} \in R_1] \cdot \Big(\Pr[\mathbf{Y} \in R_2 \mid \mathbf{X} \in R_1] - \frac{1}{n^{1/9}}\Big) \\
&\geq \Pr[\mathbf{X} \in R_1] \cdot \Pr\Big[\mathbf{Y} \in R_2 \mid \mathbf{X} \in R_1\Big] - \frac{1}{n^{1/9}} \\
&= \mu^k(R) - \frac{1}{n^{1/9}}
\end{aligned}$$

Note that the same property will trivially hold if $\mu^k(R) < \frac{1}{n}$. To prove Inequality (3.1), it suffices bounding $\|\mathbf{W}_b - \mathbf{W}_a\|_1 \leq \frac{1}{n^{1/9}}$ for arbitrary $a, b \in \{1, -1\}^k$,

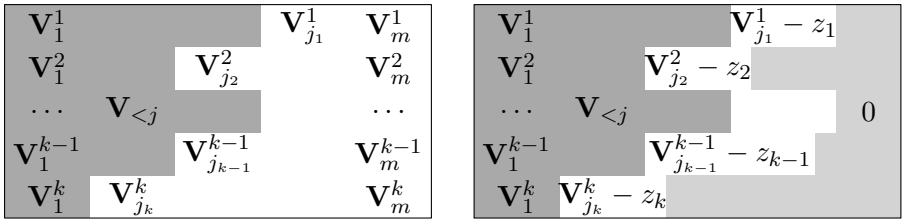

Figure 3.1: Sketch of $\mathbf{V}$ and $\mathbf{W}_z$. $\mathbf{V}_{<j}$ appears in dark gray; light gray marks zeros.

because:

$$\begin{aligned}
\left\|\mathbf{W} - \mathbf{W}_a\right\|_1 &= \sum_{\bar{y}} \left|\Pr[\mathbf{W} = \bar{y}] - \Pr[\mathbf{W}_a = \bar{y}]\right| \\
&= \sum_{\bar{y}} \left|\sum_{b \in \{1,-1\}^k} \Pr[\mathbf{Z} = b] \cdot \Pr[\mathbf{W}_b = \bar{y}] - \Pr[\mathbf{W}_a = \bar{y}]\right| \\
&\leq \sum_{b \in \{1,-1\}^k} \frac{1}{2^k} \sum_{\bar{y}} \left|\Pr[\mathbf{W}_b = \bar{y}] - \Pr[\mathbf{W}_a = \bar{y}]\right| \\
&= \sum_{b \in \{1,-1\}^k} \frac{1}{2^k} \left\|\mathbf{W}_b - \mathbf{W}_a\right\|_1
\end{aligned}$$

Then, let $\mathbf{V}$ denote a random variable distributed as $\mathbf{X}$ conditioned on $\mathbf{X} \in R_1$. Since $R_1 \subset \mathcal{X}$ and $\mathbf{X}$ is uniform over $\mathcal{X}$, $\mathbf{V}$ itself is drawn uniformly from R_1. First we prove that

$$H(\mathbf{V}) \geq nk - o(1) - \log n \tag{3.2}$$

Since $\mathbf{V}$ is uniform on R_1, $H(\mathbf{V}) = \log |R_1|$. As we assumed R was large,

$$\frac{|R_1|}{|\mathcal{X}|} = \Pr[\mathbf{X} \in R_1] \geq \frac{1}{n}.$$

Thus, by Lemma 3.7:

$$\begin{aligned}
H(\mathbf{V}) \;&\geq\; \log \frac{|\mathcal{X}|}{n} = \log \left[(2^{n/m} - 2)^{mk}\right] - \log n \\
&\geq\; mk \left(\frac{n}{m} - \frac{1}{2^{\frac{n}{m}-2}}\right) - \log n \;=\; nk - o(1) - \log n.
\end{aligned}$$

The variable $\mathbf{V}$ can be divided into the same blocks as the variable $\mathbf{X}$. Thus, $\mathbf{V} = \mathbf{V}^1, \ldots, \mathbf{V}^k$ where each $\mathbf{V}^i$ is an n-bit number. Each $\mathbf{V}^i$ is $\mathbf{V}_1^i \ldots \mathbf{V}_m^i$ where each $\mathbf{V}_j^i$ is an ℓ-bit number. Let j be a vector in $[m]^k$. By $\mathbf{V}_{<j}$ we denote all blocks of $\mathbf{V}$ such that $\mathbf{V}_s^i$ for $i \in [k]$ and $s < j_i$. The support of $\mathbf{V}_{<j}$ is denoted by $\mathcal{V}_{<j}$. For $j \in [m]^k$, $v \in \mathcal{V}_{<j}$ and $z \in \{-1, 1\}^k$, let $\mathbf{V}_{j,v}$ be the random variable $\mathbf{V}_{j_1}^1, \ldots, \mathbf{V}_{j_k}^k$ conditioned on $\mathbf{V}_{<j} = v$, and $\mathbf{V}_{j,v,z}$ be the random variable $\mathbf{V}_{j_1}^1 - z_1, \ldots, \mathbf{V}_{j_k}^k - z_k$, conditioned on $\mathbf{V}_{<j} = v$. This is the same distribution as the projection of $\mathbf{Y}$ onto blocks given by j, conditioned on $\mathbf{J} = j, \mathbf{Z} = z$ and $\mathbf{V}_{<j} = v$. Figure 3.1 illustrates the notation. Since $\mathbf{W}_z$ contains only zeroes in blocks "after $\mathbf{J}$" and by Observation 1.3, we get,

$$\left\|\mathbf{W}_b - \mathbf{W}_a\right\|_1 \leq \sum_{j \in [m]^k} \sum_{v \in \mathcal{V}_{<j}} \Pr[\mathbf{J} = j, \mathbf{V}_{<j} = v] \cdot \left\|\mathbf{V}_{j,v,b} - \mathbf{V}_{j,v,a}\right\|_1.$$

Now notice that $\mathbf{V}_{j,v,a} = \pi(\mathbf{V}_{j,v,b}) = \pi'(\mathbf{V}_{j,v})$ for some permutations π and π' of the domain $\{0,1\}^{\frac{nk}{m}}$. By Observation 1.4, we get for $\mathbf{U}$ uniform over $\{0,1\}^{\frac{nk}{m}}$,

$$\left\|\mathbf{V}_{j,v,b} - \mathbf{V}_{j,v,a}\right\|_1 \leq 2 \cdot \left\|\mathbf{V}_{j,v} - \mathbf{U}\right\|_1.$$

Then by Pinsker's inequality (Corollary 1.6),

$$\left\|\mathbf{V}_{j,v} - \mathbf{U}\right\|_1 \leq \sqrt{2 \cdot \left(\frac{nk}{m} - H\Big(\mathbf{V}^1_{j_1}, \ldots, \mathbf{V}^k_{j_k} \mid \mathbf{V}_{<j} = v\Big)\right)}.$$

We may thus bound

$$\begin{aligned}
\left\|\mathbf{W}_b - \mathbf{W}_a\right\|_1 &\leq \sum_{j,v} 2 \cdot \frac{1}{m^k} \cdot \Pr[\mathbf{V}_{<j} = v] \sqrt{2 \cdot \left(\frac{nk}{m} - H\Big(\mathbf{V}^1_{j_1} \ldots \mathbf{V}^k_{j_k} \mid \mathbf{V}_{<j} = v\Big)\right)} \\
&\leq 2 \cdot \sqrt{2 \cdot \sum_{j,v} \frac{1}{m^k} \Pr[\mathbf{V}_{<j} = v] \left(\frac{nk}{m} - H\Big(\mathbf{V}^1_{j_1} \ldots \mathbf{V}^k_{j_k} \mid \mathbf{V}_{<j} = v\Big)\right)} \\
&\qquad \text{(by concavity of } \sqrt{\cdot}\text{)} \\
&= \sqrt{8 \cdot \left(\frac{nk}{m} - \frac{1}{m^k} \sum_j H\Big(\mathbf{V}^1_{j_1} \ldots \mathbf{V}^k_{j_k} \mid \mathbf{V}_{<j}\Big)\right)} \\
&\qquad \text{(by definition of conditional entropy)}
\end{aligned}$$

Now we need to bound the sum $\sum_j H\Big(\mathbf{V}^1_{j_1} \ldots \mathbf{V}^k_{j_k} \mid \mathbf{V}_{<j}\Big)$. The sum is over all vectors j in $[m]^k$. We will divide the summands into parts that allow us to use the chain rule. We call a vector $p \in [m]^k$ a *pattern* if p contains 1 in some coordinate. We denote the set of all patterns by $\mathcal{P}$. For a pattern $p \in \mathcal{P}$ we define a width $w(p)$ of pattern p as the maximum of entries of p:

$$w(p) = \max_{i \in [k]} p_i.$$

Denote the set of all patterns of width w by $\mathcal{P}_w$. For an integer s, we denote $(p+s) = (p_1 + s, p_2 + s, \ldots, p_k + s)$. We can rewrite the sum of entropies:

$$\sum_{j \in [m]^k} H\Big(\mathbf{V}^1_{j_1} \ldots \mathbf{V}^k_{j_k} \mid \mathbf{V}_{<j}\Big) = \sum_{w \in [m]} \sum_{p \in \mathcal{P}_w} \sum_{s=0}^{m-w} H\Big(\mathbf{V}^1_{p_1+s} \ldots \mathbf{V}^k_{p_k+s} \mid \mathbf{V}_{<(p+s)}\Big). \tag{3.3}$$

We fix some pattern $p \in \mathcal{P}_w$ and bound the last sum

$$M^p = \sum_{s=0}^{m-w} H\Big(\mathbf{V}^1_{p_1+s} \ldots \mathbf{V}^k_{p_k+s} \mid \mathbf{V}_{<(p+s)}\Big).$$

Let $p' = p + (m - w)$. Let $\mathbf{L}$ be blocks of $\mathbf{V}$ "to the left" of p and $\mathbf{R}$ be blocks "to the right" of p'. Formally,

$$\mathbf{L}^p = \mathbf{V}^1_1 \ldots \mathbf{V}^1_{p_1-1} \;\; \ldots \;\; \mathbf{V}^k_1 \ldots \mathbf{V}^k_{p_k-1} \qquad \mathbf{R}^{p'} = \mathbf{V}^1_{p'_1+1} \ldots \mathbf{V}^1_m \;\; \ldots \;\; \mathbf{V}^k_{p'_k+1} \ldots \mathbf{V}^k_m.$$

The variables $\mathbf{L}^p$ and $\mathbf{R}^{p'}$ are chosen in a way that they, together with the blocks used in the sum M^p, "cover" all blocks of $\mathbf{V}$. For a better understanding see Figure 3.2.

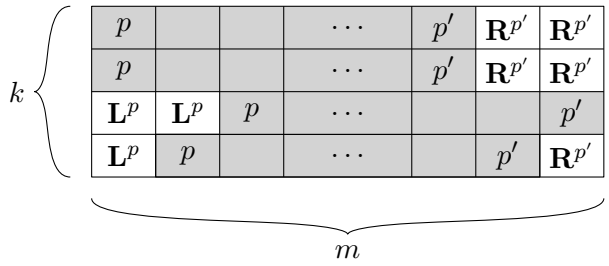

Figure 3.2: An example how to cover all blocks in the sum M^p (gray blocks) and the variable $\mathbf{L}^p$ and $\mathbf{R}^{p'}$. Each rectangle represents a block of $\frac{n}{m}$ bits.

Note that variables $\mathbf{L}^p$ and $\mathbf{R}^{p'}$ contains together $(w-1)k$ blocks (i.e., $\frac{n}{m}(w-1)k$ bits) independently of the choice of $p \in \mathcal{P}_w$. The chain rule then says that

$$\begin{aligned} H(\mathbf{V}) &= H(\mathbf{L}^p) + M^p + H(\mathbf{R}^{p'}|\mathbf{V}_{\leq p'}), \\ M^p &= H(\mathbf{V}) - H(\mathbf{L}^p) - H(\mathbf{R}^{p'}|\mathbf{V}_{\leq p'}) \geq H(\mathbf{V}) - \frac{n}{m}(w-1)k. \end{aligned}$$

Now, we are ready to bound the sums from Equation (3.3).

$$\begin{aligned} \sum_{w\in[m]} \sum_{p\in\mathcal{P}_w} \sum_{s=0}^{m-w} H\Big(\mathbf{V}^1_{p_1+s} \dots \mathbf{V}^k_{p_k+s} \mid \mathbf{V}_{<(p+s)}\Big) &\geq \sum_{w\in[m]} \sum_{p\in\mathcal{P}_w} H(\mathbf{V}) - \frac{n}{m}(w-1)k \\ &\geq \sum_{w\in[m]} \sum_{p\in\mathcal{P}_w} nk - o(1) - \log n - \frac{n}{m}(w-1)k \qquad \text{(by Equation (3.2))} \\ &= \frac{nk}{m}\left(\sum_{w\in[m]} \sum_{p\in\mathcal{P}_w} m - w + 1\right) - |\mathcal{P}|\Big(o(1) + \log n\Big) \\ &= m^k \frac{nk}{m} - |\mathcal{P}|\Big(o(1) + \log n\Big) \end{aligned}$$

Note that the last equality is given by that $\sum_{w\in[m]} \sum_{p\in\mathcal{P}_w} m - w + 1$ equals to the number of all $j \in [m]^k$. Bounding the number of all patterns by $|\mathcal{P}| \leq km^{k-1}$, we can also bound

$$\begin{aligned} \Big\|\mathbf{W}_a - \mathbf{W}_b\Big\|_1 &\leq \sqrt{8 \cdot \left(\frac{nk}{m} - \frac{1}{m^k} \sum_j H\Big(\mathbf{V}^1_{j_1} \dots \mathbf{V}^k_{j_k} \mid \mathbf{V}_{<j}\Big)\right)} \\ &\leq \sqrt{8 \cdot \left(\frac{nk}{m} - \frac{1}{m^k}\left(m^k \frac{nk}{m} - |\mathcal{P}|\Big(o(1) + \log n\Big)\right)\right)} \\ &\leq \sqrt{8 \cdot \frac{km^{k-1}}{m^k}\Big(\log n + o(1)\Big)} \\ &\leq \sqrt{8 \cdot \frac{1}{n^{\frac{1}{4}}}\Big(\log n + o(1)\Big)} \leq \frac{1}{n^{\frac{1}{9}}} \\ &\qquad \text{(for } m = \sqrt{n}, k \leq n^{1/4} \text{ and } n \text{ large enough)} \end{aligned}$$

□

4. Network Coding Conjecture Implies Data Structure Lower Bounds

4.1 Introduction

In this chapter, we study lower bounds for static data structures. As mentioned above, proving polynomial lower bounds on query time for certain static data structure problems seems out of reach. To deal with this situation researchers developed various conjectures which if true would imply the sought-after lower bounds. For that aim, we use the Network Coding Conjecture (NCC) by Li and Li [70]. This conjecture attracted recently a lot of attention and it was used to prove various lower bounds such as lower bounds on the size of circuits computing multiplication [5] or sorting [9], and the number of IO operations needed for external memory sorting [36].

We study a certain data structure type problem for function inversion [46], which is popular in cryptography. We show that the Network Coding Conjecture implies certain weak lower bounds for the inversion data structure problem. We show that similar results apply to a host of other data structure problems such as the well-studied polynomial evaluation problem or the Finite Field Fourier transform problem.

The data structures considered in this chapter are static, non-adaptive, and systematic, i.e., a very restricted class of data structures for which lower bounds should perhaps be easier to obtain. Such data structure problems have the following structure: Given the input data described by N bits, create a data structure of size s. Then, we receive a single query from a set of permissible queries and we are supposed to answer the query while non-adaptively inspecting at most t locations in the data structure and in the original data.

We show that logarithmic lower bounds on t for the data structures can be derived from the Network Coding Conjecture even in the more generous setting of $s \geq \varepsilon N$ and when inspecting locations in the data structure is for free. Our technique seems applicable to data structure problems that are *involutions* that are inverses of themselves. Although we use a lot of the same technical machinery as the previous papers on the Network Coding Conjecture our proofs involve new ideas. An interesting aspect of our proofs is that they apply the hypothesized data structure twice in the reductions. This is reminiscent of many quantum algorithms that use Hadamard transform twice in a row (like Shor's quantum algorithm for the Satisfiability problem [8]).

Organization. In the next section, we review the data structure problems we consider. Then we provide a precise definition of Network Coding Conjecture in Section 4.3. The statement and the proof of the main results of this chapter are in Section 4.4.

4.2 Data Structure Problems

In this chapter, we study lower bounds on *systematic data structures* for various problems – function inversion, polynomial evaluation, and polynomial interpolation. We are given an input $I = \{x_0, \ldots, x_{n-1}\}$, where each $x_i \in [n] = \{0, \ldots, n-1\}$ or each x_i is an element of some field $\mathbb{F}$. First, a data structure algorithm can preprocess I to produce an advice string $\tilde{a}_I$ of s bits (we refer to the parameter s as *space* of the data structure $\mathcal{D}$). Then, we are given a query q and the data structure should produce a correct answer (what is a correct answer depends on the problem). To answer a query q, the data structure $\mathcal{D}$ has access to the whole advice string $\tilde{a}_I$ and can make t probes to the input I, i.e., read at most t elements from I. We refer to the parameter t as the query time of the data structure.

4.2.1 Function Inversion

In the function inversion problem, we are given a function $f : [n] \to [n]$ and a point $y \in [n]$ and we want to find $x \in [n]$ such that $f(x) = y$. This is a central problem in cryptography as many cryptographic primitives rely on the existence of a function that is hard to invert. To sum up we are interested in the following problem.

Function Inversion	
Input:	A function $f : [n] \to [n]$ as an oracle.
Preprocessing:	Using f, prepare an advice string $\tilde{a}_f \in \{0,1\}^s$.
Query:	Point $y \in [n]$.
Answer:	Compute the value $f^{-1}(y)$, with a full access to $\tilde{a}_f$ and using at most t probes to the oracle for f.

We want to design an efficient data structure, i.e., make s and t as small as possible. There are two trivial solutions. The first one is that the whole function f^{-1} is stored in the advice string $\tilde{a}_f$, thus $s = O(n \log n)$ and $t = 0$. The second one is that the whole function f is probed during answering a query $y \in [n]$, thus $t = O(n)$ and $s = 0$. Note that the space s of the data structure is the length of the advice string $\tilde{a}_f$ in bits but with one oracle-probe x_i the data structure reads the whole $f(x_i)$, thus with n oracle-probes we read the whole description of f, i.e., $n \log n$ bits.

The question is whether we can design a data structure with $s, t \leq o(n)$. Hellman [46] gave the first non-trivial solution and introduced a randomized systematic data structure which inverts a function with a constant probability (over the uniform choice of the function f and the query $y \in [n]$) and $s = O(n^{2/3} \log n)$ and $t = O(n^{2/3} \log n)$. Fiat and Naor [38] improved the result and introduced a data structure that inverts any function at any point, however with a slightly worse trade-off: $s^3 t = O(n^3 \log n)$. Hellman [46] also introduced a more efficient data structure for inverting a permutation – it inverts any permutation at any point and $st = O(n \log n)$. Thus, it seems that inverting a permutation is an easier problem than inverting an arbitrary function.

In this chapter, we are interested in lower bounds for the inversion problem. Yao [106] gave a lower bound that any systematic data structure for the inversion problem must have $st \geq \Omega(n \log n)$, however, the lower bound is applicable only

if $t \leq O(\sqrt{n})$. Since then, only slight progress was made. De et al. [32] improved the lower bound of Yao [106] to be applicable for the full range of t. Abusalah et al. [3] improved the trade-off, that for any k it must hold that $s^k t \geq \Omega(n^k)$. Seemingly, their result contradicts Hellman's trade-off $s = t = O(n^{2/3} \log n)$ as it implies $s = t \geq n^{k/k+1}$ for any k. However, for Hellman's attack [46] we need that the function can be efficiently evaluated and the functions introduced by Abusalah et al. [3] cannot be efficiently evaluated. There is also a series of papers [42, 99, 34, 27] that studied how the probability of successful inversion depends on the parameters s and t. However, none of these results yields a better lower bound than $st \geq \Omega(n \log n)$. Hellman's trade-off is still the best-known upper bound trade-off for the inversion problem. Thus, there is still a substantial gap between the lower and upper bounds.

Another caveat of all known data structures for the inversion is that they heavily use adaptivity during answering queries $y \in [n]$. I.e., probes to the oracle depend on the advice string $\tilde{a}$ and answers to the oracle probes which have been already made. We are interested in non-adaptive data structures. We say a systematic data structure is *non-adaptive* if all oracle probes depend only on the query $y \in [n]$.

As non-adaptive data structures are weaker than adaptive ones, there is a hope that for non-adaptive data structures we could prove stronger lower bounds. Moreover, the non-adaptive data structure corresponds to circuits computation [100, 101, 102, 28]. Thus, we can derive a circuit lower bound from a strong lower bound for a non-adaptive data structure. Non-adaptive data structures were considered by Corrigan-Gibbs and Kogan [28]. They proved that improvement by a polynomial factor of Yao's lower bound [106] for non-adaptive data structures would imply the existence of a function $F : \{0,1\}^N \to \{0,1\}^N$ for $N = n \log n$ that cannot be computed by a linear-size and logarithmic-depth circuit. A connection similar to Corrigan-Gibbs and Kogan between data structures and circuits was made also by Viola [104].

4.2.2 Evaluation and Interpolation of Polynomials

In this section, we describe two natural problems connected to polynomials. We consider our problems over a finite field $\mathbb{F}$ to avoid issues with encoding reals.

Polynomial Evaluation over $\mathbb{F}$	
Input:	Coefficients of a polynomial $p \in \mathbb{F}[x]$: $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{F}$ (i.e., $p(x) = \sum_{i \in [n]} \alpha_i x^i$)
Preprocessing:	Using the input, prepare an advice string $\tilde{a}_p \in \{0,1\}^s$.
Query:	A number $x \in \mathbb{F}$.
Answer:	Compute the value $p(x)$, with a full access to $\tilde{a}_p$ and using at most t probes to the coefficients of p.

Polynomial Interpolation over $\mathbb{F}$	
Input:	Point-value pairs of a polynomial $p \in \mathbb{F}[x]$ of degree at most $n-1$: $(x_0, p(x_0)), \ldots, (x_{n-1}, p(x_{n-1})) \in \mathbb{F}\times\mathbb{F}$ where $x_i \neq x_j$ for any two indices $i \neq j$
Preprocessing:	Using the input, prepare an advice string $\tilde{a}_p \in \{0,1\}^s$.
Query:	An index $j \in [n]$.
Answer:	Compute the j-th coefficient of the polynomial p, i.e., the coefficient of x^j in p, with a full access to $\tilde{a}_p$ and using at most t probes to the oracle for point-value pairs.

We often use a version of polynomial interpolation where the points $x_0, \ldots, x_{n-1}$ are fixed in advance and the input consists just of $p(x_0), \ldots, p(x_{n-1})$. Since we are interested in lower bounds, this makes our results slightly stronger.

Let $\mathbb{F} = \mathrm{GF}(p^k)$ denote the Galois Field of p^k elements. Let n be a divisor of $p^k - 1$. It is a well-known fact that for any finite field $\mathbb{F}$ its multiplicative group $\mathbb{F}^*$ is cyclic (see e.g. Serre [93]). Thus, there is a primitive n-th root of unity $\sigma \in \mathbb{F}$ (that is an element σ such that $\sigma^n = 1$ and for each $1 \leq j < n$, $\sigma^j \neq 1$). Pollard [87] defines the *Finite Field Fourier transform* (FFFT) (with respect to σ) as a linear function $\mathrm{FFFT}_{n,\sigma} \colon \mathbb{F}^n \to \mathbb{F}^n$ which satisfies:

$$\begin{aligned}\mathrm{FFFT}_{n,\sigma}(\alpha_0, \ldots, \alpha_{n-1}) &= (\beta_0, \ldots, \beta_{n-1}) \text{ where} \\ \beta_i &= \sum_{j\in[n]} \alpha_j \sigma^{ij} \qquad \text{for any } i \in [n]\end{aligned}$$

The inversion $\mathrm{FFFT}^{-1}_{n,\sigma}$ is given by:

$$\begin{aligned}\mathrm{FFFT}^{-1}_{n,\sigma}(\beta_0, \ldots, \beta_{n-1}) &= (\alpha_0, \ldots, \alpha_{n-1}) \text{ where} \\ \alpha_i &= \frac{1}{n} \sum_{j\in[n]} \beta_j \sigma^{-ij} \qquad \text{for any } i \in [n]\end{aligned}$$

Here, $\frac{1}{n} = \left(\sum_{i=1}^{n} 1\right)^{-1}$ over $\mathbb{F}$. In our theorems we always set n to be a divisor of $|\mathbb{F}| - 1 = p^k - 1$ thus n modulo p is non-zero and the inverse exists. Observe, that $\mathrm{FFFT}^{-1}_{n,\sigma} = \frac{1}{n}\mathrm{FFFT}_{n,\sigma^{-1}}$.

FFFT is the finite field analog of Discrete Fourier transform (DFT) which works over complex numbers. The FFT algorithm by Cooley and Tukey [26] can be used for the case of finite fields as well (as observed by Pollard [87]) to get an algorithm using $O(n \log n)$ field operations (addition or multiplication of two numbers). Thus we can compute $\mathrm{FFFT}_{n,\sigma}$ and its inverse in $O(n \log n)$ field operations.

It is easy to see that $\mathrm{FFFT}_{n,\sigma}$ is actually evaluation of a polynomial in multiple special points (specifically in $\sigma^0, \ldots, \sigma^{n-1}$). We can also see that it is a special case of interpolation by a polynomial in multiple special points since $\mathrm{FFFT}^{-1}_{n,\sigma} = \frac{1}{n}\mathrm{FFFT}_{n,\sigma^{-1}}$. We provide an NCC-based lower bound for data structures computing the polynomial evaluation. However, we use the data structure only for evaluating a polynomial in powers of a primitive root of unity. Thus, the same proof yields a lower bound for data structures computing the polynomial interpolation.

There is a great interest in data structures for polynomial evaluation in a cell probe model. In this model, some representation of a polynomial $p = \sum_{i\in[n]} \alpha_i x^i \in$

$\mathbb{F}[x]$ is stored in a table $\mathcal{T}$ of s_{cell} cells, each of w bits. Usually, w is set to $O(\log|\mathbb{F}|)$, that we can store an element of $\mathbb{F}$ in a single cell. On a query $x \in \mathbb{F}$ the data structure should output $p(x)$ making at most t_{cell} probes to the table $\mathcal{T}$. A difference between data structures in the cell probe model and systematic data structures is that a data structure in the cell probe model is charged for any probe to the table $\mathcal{T}$ but a systematic data structure is charged only for probes to the input (the coefficients α_i), reading from the advice string $\tilde{a}_p$ is for free. Note that, the coefficients α_i of p do not have to be even stored in the table $\mathcal{T}$. There are again two trivial solutions. The first one is that we store a value $p(x)$ for each $x \in \mathbb{F}$ and on a query $x \in \mathbb{F}$ we probe just one cell. Thus, we would get $t_{\mathsf{cell}} = 1$ and $s_{\mathsf{cell}} = |\mathbb{F}|$ (we assume that we can store an element of $\mathbb{F}$ in a single cell). The second one is that we store the coefficients of p and on a query $x \in \mathbb{F}$ we probe all cells and compute the value $p(x)$. Thus, we would get $t_{\mathsf{cell}} = s_{\mathsf{cell}} = n$.

Let $k = \log|\mathbb{F}|$. Kedlaya and Umans [58] provided a data structure for the polynomial evaluation that uses space $n^{1+\varepsilon} \cdot k^{1+o(1)}$ and query time $\log^{O(1)} n \cdot k^{1+o(1)}$. Note that, $n \cdot k$ is the size of the input and k is the size of the output.

The first lower bound for the cell probe model was given by Miltersen [72]. He proved that for any cell probe data structure for the polynomial evaluation it must hold that $t_{\mathsf{cell}} \geq \Omega(k/\log s_{\mathsf{cell}})$. This was improved by Larsen [67] to $t_{\mathsf{cell}} \geq \Omega(k/\log(s_{\mathsf{cell}}w/nk))$, that gives $t_{\mathsf{cell}} \geq \Omega(k)$ if the data structure uses linear space $s_{\mathsf{cell}} \cdot w = O(n \cdot k)$. However, the size of $\mathbb{F}$ has to be super-linear, i.e., $|\mathbb{F}| \geq n^{1+\Omega(1)}$, and it is not known if the bound holds for smaller fields, e.g., of linear size. Data structures in a bit probe model were studied by Gál and Miltersen [40]. The bit probe model is the same as the cell probe model but each cell contains only a single bit, i.e., $w = 1$. They studied succinct data structures that are data structures such that $s_{\mathsf{cell}} = (n + r) \cdot k$ for $r < o(n)$. Thus, the succinct data structures are related to systematic data structures but still, the succinct data structures are charged for any probe (as any other data structure in the cell probe model). Note that a succinct data structure stores only a few more bits than it is needed due to an information-theoretic requirement. Gál and Miltersen [40] showed that for any succinct data structure in the bit probe model it holds that $r \cdot t_{\mathsf{cell}} \geq \Omega(n \cdot k)$. We are not aware of any lower bound for systematic data structures for the polynomial evaluation.

Larsen et al. [68] also gave a log-squared lower bound for dynamic data structures in the cell probe model. Dynamic data structures also support updates of the polynomial p which usually impacts their query time.

There is a great interest in algorithmic questions about the polynomial interpolation such as how fast we can interpolate polynomials [41, 16, 45], how many probes we need to interpolate a polynomial if it is given by oracle [24, 48], how to compute the interpolation in a numerically stable way over infinite fields [97] and many others. However, we are not aware of any results about data structures for the interpolation, i.e., when the interpolation algorithm has an access to some precomputed advice.

4.3 Network Coding

We prove our conditional lower bounds based on the Network Coding Conjecture. In network coding, we are interested in how much information we can send

through a given network. A *network* consists of a graph $G = (V, E)$, positive capacities of edges $c : E \to \mathbb{R}^+$ and k pairs of vertices $(s_0, t_0), \ldots, (s_{k-1}, t_{k-1})$. We say a network $R = (G, c, (s_i, t_i)_{i\in[k]})$ is *undirected* or *directed (acyclic)* if the graph G is undirected or directed (acyclic). We say a network is *uniform* if the capacities of all edges in the network equal to some $q \in \mathbb{R}^+$ and we denote such network as $(G, q, (s_i, t_i)_{i\in[k]})$.

A goal of a coding scheme for directed acyclic network $R = (G, c, (s_i, t_i)_{i\in[k]})$ is that at each target t_i it will be possible to reconstruct an input message w_i which was generated at the source s_i. The coding scheme specifies messages sent from each vertex along the outgoing edges as a function of received messages. Moreover, the length of the messages sent along the edges has to respect the edge capacities.

More formally, each source s_i of a network receives an input message w_i as a sample of a uniform random variable $\mathbf{W}_i$ over a set W_i ($\mathbf{W}_i$ is independent on the other input random variables $\mathbf{W}_j$'s). Without loss of generality we can assume that each source s_i has an in-degree 0 (otherwise we can add a vertex s_i' and an edge (s_i', s_i) and replace s_i by s_i'). There is an alphabet Σ_e for each edge $e \in E(G)$. For each source s_i and each outgoing edge $e = (s_i, u)$ there is a function $f_{s_i,e} : W_i \to \Sigma_e$ which specifies the message sent along the edge e as a function of the received input message $w_i \in W_i$. For each non-source vertex $v \in V, v \neq s_i$ and each outgoing edge $e = (v, u)$ there is a similar function $f_{v,e} : \prod_{e'=(u',v)} \Sigma_{e'} \to \Sigma_e$ which specifies the message sent along the edge e as a function of the messages sent to v along the edges incoming to v. Finally, each target t_i has a decoding function $d_i : \prod_{e'=(u',t_i)} \Sigma_{e'} \to W_i$. The coding scheme is executed as follows:

1. Each source s_i receives an input message $w_i \in W_i$. Along each edge $e = (s_i, u)$ a message $f_{s_i,e}(w_i)$ is sent.

2. When a vertex v receives all messages $m_1, \ldots, m_a$ along all incoming edges (u', v) it sends along each outgoing edge $e = (v, u)$ a message

$$f_{v,e}(m_1, \ldots, m_a).$$

   As the graph G is acyclic, this procedure is well-defined and each vertex of non-zero out-degree will eventually send its messages along its outgoing edges.

3. At the end, each target t_i computes a string $\tilde{w}_i = d_i(m_1', \ldots, m_b')$ where m_j' denotes the received messages along the incoming edges (u', t_i). We say the encoding scheme is *correct* if $\tilde{w}_i = w_i$ for all $i \in [k]$ and any input messages $w_0, \ldots, w_{k-1} \in W_0 \times \cdots \times W_{k-1}$.

The coding scheme has to respect the edge capacities, i.e., if $\mathbf{M}_e$ is a random variable that represents a message sent along the edge e, then $H(\mathbf{M}_e) \leq c(e)$. Note that it means that only the expected length of the message sent along an edge e does not exceed the capacity of e. A *coding rate* of a network R is the maximum r such that there is a correct coding scheme for input random variables $\mathbf{W}_0, \ldots, \mathbf{W}_{k-1}$ where $H(\mathbf{W}_i) = \log |W_i| \geq r$ for all $i \in [k]$. A network coding can be defined also for directed cyclic networks or undirected networks but we will not use it here.

Network coding is related to multicommodity flows. A multicommodity flow for an undirected network $\bar{R} = (\bar{G}, c, (s_i, t_i)_{i\in[k]})$ specifies flows for each commodity i such that they transport as many units of commodity from s_i to t_i as possible. A flow of the commodity i is specified by a function $f^i : V \times V \to \mathbb{R}_0^+$ which describes for each pair of vertices (u, v) how many units of the commodity i are sent from u to v. Each function f^i has to satisfy:

1. If u, v are not connected by an edge, then $f^i(u, v) = f^i(v, u) = 0$.
2. For each edge $\{u, v\} \in E(\bar{G})$, it holds that $f^i(u, v) = 0$ or $f^i(v, u) = 0$.
3. For each vertex v that is not the source s_i or the target t_i, it holds that what comes to the vertex v, it goes out from the vertex v, i.e.,
$$\sum_{u\in V} f^i(u, v) = \sum_{u\in V} f^i(v, u).$$
4. What is sent from the source s_i comes to the target t_i, i.e.,
$$\sum_{u\in V} f^i(s_i, u) - f^i(u, s_i) = \sum_{u\in V} f^i(u, t_i) - f^i(t_i, u).$$

Moreover, all flows together have to respect the capacities, i.e., for each edge $e = \{u, v\} \in E(\bar{G})$ it must hold that $\sum_{i\in[k]} f^i(u, v) + f^i(v, u) \leq c(e)$. A *flow rate* of a network $\bar{R}$ is the maximum r such that there is a multicommodity flow $F = (f^0, \ldots, f^{k-1})$ that for each i transports at least r units of the commodity i from s_i to t_i, i.e., for all i, it holds that $\sum_{u\in V} f^i(u, t_i) - f^i(t_i, u) \geq r$. A multicommodity flow for directed graphs is defined similarly, however, the flows can transport the commodities only in the direction of edges.

Let R be a directed acyclic network of a flow rate r'. It is clear that for a coding rate r of R it holds that $r \geq r'$. As we can send the messages without coding and thus reduce the encoding problem to the flow problem. The opposite inequality does not hold: There is a directed network $R = (G, c, (s_i, t_i)_{i\in[k]})$ such that its coding rate is $\Omega(|V(G)|)$-times larger than its flow rate as shown by Adler et al. [4]. Thus, the network coding for directed networks provides an advantage over the simple solution given by the maximum flow. However, such a result is not known for undirected networks. Li and Li [70] conjectured that the network coding does not provide any advantage for undirected networks, thus for any undirected network $\bar{R}$, the coding rate of $\bar{R}$ equals to the flow rate of $\bar{R}$. This conjecture is known as *Network Coding Conjecture* (NCC) and we state a weaker version of it below.

For a directed graph $G = (V, E)$ we denote by $\mathsf{un}(G)$ the undirected graph $(V, \bar{E})$ obtained from G by making each directed edge in E undirected (i.e., replacing each $(u, v) \in E(G)$ by $\{u, v\}$). For a directed acyclic network $R = (G, c, (s_i, t_i)_{i\in[k]})$ we define the undirected network $\mathsf{un}(R) = (\mathsf{un}(G), \bar{c}, (s_i, t_i)_{i\in[k]})$ by keeping the source-target pairs and capacities the same, i.e,
$$c((u, v)) = \bar{c}(\{u, v\}).$$

Conjecture 4.1 (Weaker NCC)**.** *Let R be a directed acyclic network, r be a coding rate of R and $\bar{r}$ be a flow rate of $\mathsf{un}(R)$. Then, $r = \bar{r}$.*

As mentioned above, this conjecture was used to prove a conditional lower bound for certain algorithms and circuites [36, 5, 9].

4.4 NCC Implies Weak Data Structure Lower Bounds

In this section, we provide a connection that the Network Coding Conjecture (Conjecture 4.1) implies lower bounds for data structures for the permutation inversion and the polynomial evaluation and interpolation. Assuming NCC, we show that a query time t of a non-adaptive systematic data structure for any of the above problems satisfies $t \geq \Omega(\log n / \log\log n)$, even if it uses linear space, i.e., the advice string $\tilde{a}$ has size $\varepsilon n \log n$ for sufficiently small constant $\varepsilon > 0$. Formally, we define $t_{\mathsf{Inv}}(s)$ as a query time of the optimal non-adaptive systematic data structure for the permutation inversion using space at most s. Similarly, we define $t^{\mathbb{F}}_{\mathsf{Eval}}(s)$ and $t^{\mathbb{F}}_{\mathsf{Interp}}(s)$ for the polynomial evaluation and interpolation over $\mathbb{F}$.

Theorem 4.2. *Let $\varepsilon > 0$ be a sufficiently small constant. Assuming NCC, it holds that $t_{\mathsf{Inv}}(\varepsilon n \log n) \geq \Omega(\log n / \log\log n)$.*

Theorem 4.3. *Let $\mathbb{F}$ be a field and n be a divisor of $|\mathbb{F}| - 1$. Let $s = \varepsilon n \log |\mathbb{F}|$ for a sufficiently small constant $\varepsilon > 0$. Then assuming NCC, it holds that*

$$t^{\mathbb{F}}_{\mathsf{Eval}}(s), t^{\mathbb{F}}_{\mathsf{Interp}}(s) \geq \Omega(\log n / \log\log n).$$

Note that by Theorem 4.2, assuming NCC, it holds that

$$s \cdot t \geq \Omega\Big(n \log^2 n / \log\log n\Big)$$

for $s = \varepsilon n \log n$ and $t = t_{\mathsf{Inv}}(s)$. The same holds for $t^{\mathbb{F}}_{\mathsf{Eval}}$ and $t^{\mathbb{F}}_{\mathsf{Interp}}$ by Theorem 4.3. Thus, these conditional lower bounds cross the barrier $\Omega(n \log n)$ for $s \cdot t$ given by the best unconditional lower bounds known for the function inversion [106, 32, 3, 42, 99, 34, 27] and the lower bound for the succinct data structures for the polynomial evaluation by Gál and Miltersen [40]. Note that our lower bound does not contradict Hellman's attack [46] for the permutation inversion, as his data structure is heavily adaptive.

Our lower bound for the data structure for the polynomial evaluation and interpolation is applicable even for linear size field (i.e., linear number of queries). Larsen's lower bound for the data structure for the polynomial evaluation [67] is applicable only for superlinear fields, i.e., $|\mathbb{F}| \geq n^{1+\Omega(1)}$. We give the result for the polynomial evaluation here as it has an analogous proof as the lower bound for the polynomial interpolation and it might illustrate a more general phenomenon that our technique might be applicable to a broader class of functions that contains an involution as a subproblem.

To prove Theorems 4.2 and 4.3, we build on the technical machinery of Farhadi et al. [36]. The proof can be divided into two steps:

1. From a data structure for the problem we derive a network R with $O(tn)$ edges such that R admits an encoding scheme that is correct on a large fraction of the inputs. This step is distinct for each problem and the reductions are shown in Sections 4.4.1 and 4.4.2. This step uses new ideas and interestingly, it uses the data structure twice in a sequence.

2. If there is a network R with dn edges that admits an encoding scheme which is correct for a large fraction of inputs, then $d \geq \Omega(\log n / \log\log n)$. This step is common to all the problems. It was implicitly proved by Farhadi et al. [36] and Afshani et al. [5].

4.4.1 Function Inversion

In this section, we prove Theorem 4.2 that assuming NCC, any non-adaptive systematic data structure for the permutation inversion requires query time at least $\Omega(\log n / \log\log n)$ even if it uses linear space. Let $\mathcal{D}$ be a data structure for inverting permutations of a linear space $s = \varepsilon n \log n$, for sufficiently small constant $\varepsilon < 1$, with query time $t = t_{\mathsf{Inv}}(s)$. Recall that $t_{\mathsf{Inv}}(s)$ is a query time of the optimal non-adaptive systematic data structure for the permutation inversion using space s. From $\mathcal{D}$ we construct a directed acyclic network $R = (G, c, (s_i, t_i)_{i\in[n]})$ and an encoding scheme of a coding rate $\log n$. By Conjecture 4.1 we get that the flow rate of $\mathsf{un}(R) = (\bar{G}, c, (s_i, t_i)_{i\in[n]})$ is $\log n$ as well. We prove that there are many source-target pairs of distance at least $\Omega(\log_t n)$. Since the number of edges of $\bar{G}$ will be $O(tn)$ and flow rate of $\mathsf{un}(R)$ is $\log n$, we are able to derive a lower bound $t \geq \Omega(\log n / \log\log n)$.

We will design the network based on the *probe graph* of the data structure. By the probe graph of the data structure, we understand a graph with n input vertices corresponding to possible oracle probes and n output vertices corresponding to possible data structure queries. Each query vertex is connected to the vertices of oracle probes executed by the data structure when answering that query. Here, we use the non-adaptivity of the studied data structures as the probe graph does not depend on the stored data but only on the data structure itself. Our construction will utilize two copies of the probe graph connected in a sequence. The network will have input vertices $s_0, \ldots, s_{n-1}$ and output vertices $u_0, \ldots, u_{n-1}$ where each target t_i is set to $u_{i+b \bmod n}$ for a suitable constant b. The input vertices $s_0, \ldots, s_{n-1}$ correspond to the oracle vertices of the first copy of the probe graph. (See Fig. 4.1 for an illustration.)

We will feed distinct values $x_0, \ldots, x_{n-1} \in [n]$ to the input vertices which then send them to the query vertices of the first copy of the probe graph. Values $x_0, \ldots, x_{n-1}$ define a permutation $f(i) = x_i$. Each query vertex j of the first copy of the probe graph can determine $f^{-1}(j)$ if it is provided with the advice string $\tilde{a}_f$ of the data structure corresponding to f. (We will fix the most common advice string $\tilde{a}_f$ and restrict ourselves to inputs $x_0, \ldots, x_{n-1}$ consistent with it.) Each query vertex j can also determine the value of a newly defined function $h(j) = f^{-1}(j) + b$ which it sends along its outgoing edges. The second copy of the data structure serves to invert the function h similarly to inverting f. The oracle vertices of the second copy of the probe graph coincide with the query vertices of the first copy. The query vertices of the second copy of the probe graph are the output vertices $u_0, \ldots, u_{n-1}$. Hence, the query vertex $i + b$ of the second copy will be used to determine $h^{-1}(i + b) = x_i$. Thus, x_i is directed from s_i to $t_i = u_{i+b}$.

The above construction gives a network R with an encoding scheme E that is correct only on a substantial fraction of all possible inputs. Namely on inputs $x_0, \ldots, x_{n-1} \in [n]$ which are distinct and consistent with the fixed advices. This might bring correlations among messages received by the sources. However, the

Network Coding Conjecture requires independently sampled messages for each source. To overcome this issue we use the technique introduced by Farhadi et al. [36] to augment R so that it admits an encoding scheme for independent messages. We provide technical details next.

Let $R = (G, c, (s_i, t_i)_{i\in[k]})$ be a directed acyclic network. Let each source receive a binary string of length r as its input message, i.e., each $W_i = \{0,1\}^r$. If we concatenate all input messages w_i we get a string of length $r \cdot k$, thus the set of all possible inputs for an encoding scheme for R corresponds to the set $\mathcal{I} = \{0,1\}^{rk}$. We say an encoding scheme is correct on an input $\bar{w} = (w_0, \ldots, w_{k-1}) \in \mathcal{I}$ if it is possible to reconstruct all messages w_i at appropriate targets. An (ε, r)-*encoding scheme* is an encoding scheme which is correct on at least $2^{(1-\varepsilon)rk}$ inputs in $\mathcal{I}$.

We say a directed network $R = (G, c, (s_i, t_i)_{i\in[k]})$ is (δ, d)-*long* if for at least δk source-target pairs (s_i, t_i), it holds that distance between s_i and t_i in $\mathsf{un}(G)$ is at least d. Here, we measure the distance in the undirected graph $\mathsf{un}(G)$, even though the network R is directed. The following lemma is implicitly used by Farhadi et al. [36] and Afshani et al. [5].

Lemma 4.4 (Implicitly used in [36, 5])**.** *Let $R = (G, r, (s_i, t_i)_{i\in[k]})$ be a (δ, d)-long directed acyclic uniform network for $\delta > \frac{5}{6}$ and sufficiently large $r \in \mathbb{R}^+$. Assume there is an (ε, r)-encoding scheme for R for sufficiently small ε. Then assuming NCC, it holds that $\frac{|E(G)|}{k} \geq \delta' \cdot d$, where $\delta' = \frac{\delta - 5/6}{10}$.*

Now we are ready to prove a conditional lower bound for the permutation inversion. For the proof we use the following fact which follows from well-known Stirling's formula:

Fact 4.5. *The number of permutations $[n] \to [n]$ is at least $2^{n \log n - 2n}$.*

Theorem 4.2. *Let $\varepsilon > 0$ be a sufficiently small constant. Assuming NCC, it holds that $t_{\mathsf{Inv}}(\varepsilon n \log n) \geq \Omega(\log n / \log\log n)$.*

Proof. Let $\mathcal{D}$ be the optimal data structure for the inversion of permutation on $[n]$ using space $\varepsilon n \log n$. We set $t = t_{\mathsf{Inv}}(\varepsilon n \log n)$. We will construct a directed acyclic uniform network $R = (G, r, (s_i, t_i)_{i\in[n]})$ where $r = \log n$. Let $\varepsilon' = 2 \cdot \varepsilon + \frac{2}{q} + \frac{2}{\log n}$ for sufficiently large q so that we could apply Lemma 4.4. The network R will admit an (ε', r)-encoding scheme E. The number of edges of G will be at most $2tn$ and the network R will be $(\frac{9}{10}, d)$-long for $d = \frac{1}{2}\log_{qt} n$. Thus, by Lemma 4.4 we get that

$$2t = \frac{2tn}{n} \geq \Omega(\log_{qt} n),$$

from which we can conclude that $t \geq \Omega(\log n / \log\log n)$. Thus, it remains to construct the network R and the scheme E.

First, we construct a graph G' which will yield the graph G by deleting some edges. The graph G' has three layers of n vertices: a source layer A of n sources $s_0, \ldots, s_{n-1}$, a middle layer M of n vertices $v_0, \ldots, v_{n-1}$ and a target layer B of n vertices $u_0, \ldots, u_{n-1}$. The targets $t_0, \ldots, t_{n-1}$ of R will be assigned to the vertices $u_0, \ldots, u_{n-1}$ later.

We add edges according to the data structure $\mathcal{D}$: Let $Q_j \subseteq [n]$ be the set of oracle probes which $\mathcal{D}$ makes during the computation of $f^{-1}(j)$, i.e., for each $i \in Q_j$, the query j probes the oracle f for $f(i)$. As $\mathcal{D}$ is non-adaptive, the sets

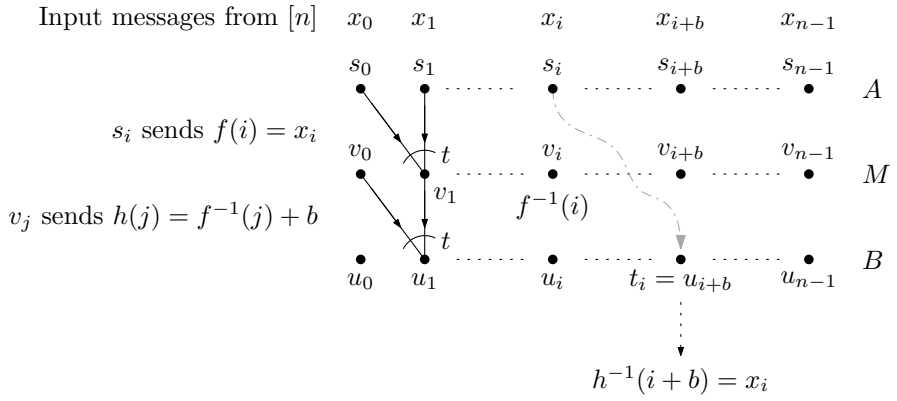

Figure 4.1: A sketch of the graph G' and encoding scheme E.

Q_j are well-defined. For each $j \in [n]$ and $i \in Q_j$ we add edges (s_i, v_j) and (v_i, u_j). We set a capacity of all edges to $r = \log n$. This finishes the construction of G', see Fig. 4.1 for illustration of the graph G'.

The graph G' has exactly $2tn$ edges. Moreover, the vertices of the middle and the target layer have in-degree at most t as the incoming edges correspond to the oracle probes made by $\mathcal{D}$. However, some vertices of the source and the middle layer might have large outdegree, which is a problem that might prevent the network R to be $(\frac{9}{10}, d)$-long. For example, the data structure $\mathcal{D}$ could always probe $f(0)$. Then, there would be edges (s_0, v_j) and (v_0, u_j) for all $j \in [n]$, hence all vertices would be at distance at most 4 in $\mathsf{un}(G')$. Thus, we need to remove edges adjacent to high-degree vertices. Let $W \subseteq V(G')$ be the set of vertices of out-degree larger than qt. We remove all edges incident to W from G' to obtain the graph G. (For simplicity, we keep the degree 0 vertices in G). Thus, the maximum degree of G is at most qt. Since the graph G' has $2tn$ edges, it holds that $|W| \leq \frac{2}{q} \cdot n$.

Now, we assign the targets of R in such a way that R is $(\frac{9}{10}, d)$-long. Let C_v be the set of vertices of G which have distance at most d from v in $\mathsf{un}(G)$. Since the maximum degree of G is at most qt and $d = \frac{1}{2}\log_{qt} n$, for each $v \in V(G)$, $|C_v| \leq 2\sqrt{n}$. In particular, for every source s_i it holds that $|C_{s_i} \cap B| \leq 2\sqrt{n}$, i.e., there are at most $2\sqrt{n}$ vertices in the target layer B at distance smaller than d from v. It follows from an averaging argument that there is an integer b such that there are at least $n - 2\sqrt{n}$ sources s_i with distance at least d from u_{i+b} in $\mathsf{un}(G)$. (Here the addition $i + b$ is modulo n.) We fix one such b and set $t_i = u_{i+b}$. For n large enough, it holds that $n - 2\sqrt{n} \geq \frac{9}{10} \cdot n$. Thus, the network R is $\left(\frac{9}{10}, d\right)$-long.

It remains to construct the (ε', r)-encoding scheme E for R (see Fig. 4.1 for a sketch of the encoding E). Each source s_i receives a number $x_i \in [n]$ as an input message. We interpret the string of the input messages $x_0, \ldots, x_{n-1}$ as a function. We define the function $f : [n] \to [n]$ as $f(i) = x_i$. We will consider only those inputs $x_0, \ldots, x_{n-1}$ which are pairwise distinct so that f is a permutation.

At a vertex v_j of the middle layer M we want to compute $f^{-1}(j)$ using the data structure $\mathcal{D}$. To compute $f^{-1}(j)$ we need the appropriate advice string $\tilde{a}_f$ and answers to the oracle probes Q_j. We fix an advice string $\tilde{a}_f$ to some particular value which will be determined later, and we focus only on inputs $x_0, \ldots, x_{n-1}$ which have the same advice string $\tilde{a}_f$. In G' the vertex v_j is connected exactly

to the sources s_i for $i \in Q_j$, but some of those connections might be missing in G. Thus for each i such that $s_i \in W$, x_i will be fixed to some particular value c_i which will also be determined later. Each source s_i sends the input x_i along all outgoing edges incident to s_i. Thus, at a vertex v_j we know the answers to all f-oracle probes in Q_j. Recall that $f(i) = x_i$ and each x_i for $i \in Q_j$ was either fixed to c_i or sent along the incoming edge $(s_i, v_j) \in E(G)$. We also know the advice string $\tilde{a}_f$ as it was fixed. Therefore, we can compute $f^{-1}(j)$ at every vertex v_j. Note that $f^{-1}(j)$ is the index of the source which received j as an input message, i.e., if $f^{-1}(j) = i$, then $x_i = j$.

Now, we define another permutation $h : [n] \to [n]$ as $h(j) = f^{-1}(j) + b$ where the addition is modulo n. Since b is fixed, we can compute $h(j)$ at each vertex v_j. The goal is to compute $h^{-1}(\ell)$ at each vertex u_ℓ of the target layer. First, we argue that $h^{-1}(i + b) = x_i$. The permutation f^{-1} maps an input message x_i to the index i. The permutation h maps an input message x_i to the index $i + b$. Thus, the inverse permutation h^{-1} maps the index $i + b$ to the input message x_i. If we are able to reconstruct $h^{-1}(i+b)$ at the target $t_i = u_{i+b}$, then in fact we are able to reconstruct x_i, the input message received by the source s_i.

To reconstruct $h^{-1}(\ell)$ at the vertex u_ℓ we use the same strategy as for reconstructing $f^{-1}(j)$ at vertices v_j. We use again $\mathcal{D}$, but this time for the function h. Again, we fix the advice string $\tilde{a}_h$ of $\mathcal{D}$, and we fix $h(j)$ to some d_j for each vertex $v_j \in W$. Each vertex v_j sends the value $h(j)$ along all edges outgoing from v_j. To compute $h^{-1}(\ell)$ we need values $h(j)$ for all $j \in Q_\ell$, which are known to the vertex u_ℓ. Again, they are either sent along the incoming edges or are fixed to d_j. Since the value of the advice string $\tilde{a}_h$ is fixed, we can compute the value $h^{-1}(\ell) = x_{\ell-b}$ at the vertex u_ℓ.

The network R is correct on all inputs $x_0, \ldots, x_{n-1}$ which encode a permutation and which are consistent with the fixed advice strings and the fixed values at the degree zero vertices. Now, we argue that we can fix all the values so that there will be many inputs consistent with them. By Fact 4.5, there are at least $2^{(n \log n)-2n}$ inputs $x_0, \ldots, x_{n-1}$ which encode a permutation. In order to make R work, we fixed the following values:

1. Advice strings $\tilde{a}_f$ and $\tilde{a}_h$, in total $2\varepsilon \cdot n \log n$ bits.

2. An input message c_i for each source s_i in W and a value d_j for each vertex v_j in W. Since $|W| \leq \frac{2}{q} \cdot n$ and $c_i, d_j \in [n]$, we fix $\frac{2}{q} \cdot n \log n$ bits in total.

Overall, we fix at most $(2\varepsilon + \frac{2}{q}) \cdot n \log n$ bits. Thus, the fixed values divide the input strings into at most $2^{(2\varepsilon+\frac{2}{q}) \cdot n \log n}$ buckets. In each bucket all the input strings are consistent with the fixed values. We conclude that there is a choice of values to fix so that its corresponding bucket contains at least $2^{(1-2\varepsilon-\frac{2}{q}-\frac{2}{\log n}) \cdot n \log n} = 2^{(1-\varepsilon') \cdot n \log n}$ input strings which encode a permutation. We pick that bucket and fix the corresponding values. Thus, the scheme E is (ε', r)-encoding scheme, which concludes the proof.

□

4.4.2 Polynomial Evaluation and Interpolation

In this section, we prove Theorem 4.3. The proof follows the blueprint of the proof of Theorem 4.2. The construction of a network R from a data structure is basically the same. Thus, we mainly describe only an (ε', r)-encoding scheme for R.

Theorem 4.3. *Let $\mathbb{F}$ be a field and n be a divisor of $|\mathbb{F}| - 1$. Let $s = \varepsilon n \log |\mathbb{F}|$ for a sufficiently small constant $\varepsilon > 0$. Then assuming NCC, it holds that*

$$t^{\mathbb{F}}_{\mathsf{Eval}}(s), t^{\mathbb{F}}_{\mathsf{Interp}}(s) \geq \Omega(\log n / \log \log n).$$

Proof. Let $\mathcal{D}$ be the optimal non-adaptive systematic data structure for the evaluation of polynomials of degree up to $n-1$ over $\mathbb{F}$ and using space $s = \varepsilon n \log |\mathbb{F}|$. We set $t = t^{\mathbb{F}}_{\mathsf{Eval}}(s), r = \log |\mathbb{F}|$ and $\varepsilon' = 2\varepsilon + \frac{2}{q}$ for sufficently large q. Again, we will construct a network $R = (G, r, (s_i, t_i)_{i\in[n]})$ from $\mathcal{D}$. To construct an (ε', r)-encoding scheme for R, we use entries of FFFT, i.e., we will evaluate polynomials of degree at most $n-1$ in powers of a primitive n-th root of the unity. Thus, we fix a primitive n-th root of unity $\sigma \in \mathbb{F}$, which we know exists, as discussed in Section 4.2.2.

We create a network R from $\mathcal{D}$ in the same way as we created in the proof of Theorem 4.2. By Lemma 4.4 we are able to conclude that $t \geq \Omega(\log n / \log \log n)$. First, we create a graph G' of three layers $A = \{s_0, \ldots, s_{n-1}\}, M = \{v_0, \ldots, v_{n-1}\}$ and $B = \{u_0, \ldots, u_{n-1}\}$ and we add $2tn$ edges to G' according to the probes of $\mathcal{D}$ – on the vertex v_j we will evaluate a polynomial in a point σ^j and on the vertex u_j we will evaluate a polynomial in a point σ^{-j}. Then, we create a graph G from G' by removing edges incident to vertices in a set W, which contains vertices of degree higher than qt. Finally, we set a shift $b \in [n]$ and set $t_i = u_{i+b}$ in such a way that the network R is $(\frac{9}{10}, d)$-long for $d = \frac{1}{2}\log_{qt} n$.

Now, we desribe an (ε', r)-encoding scheme E for R using $\mathcal{D}$. Each source s_i receives an input message $\alpha_i \in \mathbb{F}$ which we interpret as coefficients of a polynomial $p \in \mathbb{F}[x]$ (that is $p(x) = \sum_{i\in[n]} \alpha_i x^i$). Each source s_i sends its input message α_i along all outgoing edges from s_i. Each vertex v_j computes $p(\sigma^j)$ using $\mathcal{D}$. Again, we fix the advice string $\tilde{a}_p$ and the input messages α_i for the sources s_i in W. Each vertex v_j computes a value $h(j) = p(\sigma^j) \cdot \sigma^{jb}$ and sends it along all outgoing edges from v_j. We define a new polynomial $p'(x) = \sum_{j\in[n]} h(j) x^j$. We fix the advice string $\tilde{a}_{p'}$ and the values $h(j)$ for each vertex $v_j \in W$. Thus, each vertex u_ℓ can compute a value $p'(\sigma^{-\ell})$. We claim that $\frac{p'(\sigma^{-\ell})}{n} = \alpha_{\ell-b}$.

$$\begin{aligned}
\frac{p'(\sigma^{-\ell})}{n} &= \frac{1}{n}\sum_{j\in[n]} h(j)\sigma^{-\ell j} = \frac{1}{n}\sum_{j\in[n]} p(\sigma^j)\sigma^{jb}\sigma^{-\ell j} = \frac{1}{n}\sum_{j\in[n]} \left(\sum_{i\in[n]} \alpha_i \sigma^{ji}\right)\sigma^{jb}\sigma^{-\ell j} \\
&= \frac{1}{n}\sum_{i\in[n]} \alpha_i \left(\sum_{j\in[n]} \sigma^{ji}\sigma^{jb}\sigma^{-\ell j}\right) = \frac{1}{n}\sum_{i\in[n]} \alpha_i \left(\sum_{j\in[n]} \sigma^{j(i+b-\ell)}\right) \\
&= \alpha_{(\ell-b \mod n)}
\end{aligned}$$

The last equality is by noting that $\sum_{j\in[n]} \sigma^{j(i+b-\ell)} = n$ for $i = \ell - b$ and 0 otherwise. Therefore, at each target $t_i = u_{i+b}$ we can reconstruct the input message α_i. See Fig. 4.2 for a sketch of the scheme E.

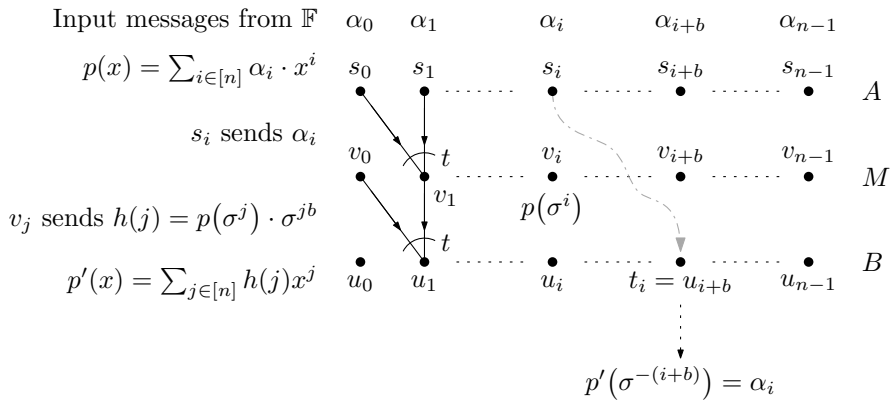

Figure 4.2: Sketch of the encoding scheme E.

Again, we can fix values of advice strings $\tilde{a}_p$ and $\tilde{a}_{p'}$ (at most $2\varepsilon \cdot n \log |\mathbb{F}|$ fixed bits), input messages α_i for each $s_i \in W$ and value of $h(j)$ for each $v_j \in W$ (at most $\frac{2}{q} \cdot n \log |\mathbb{F}|$ fixed bits) in such a way there is a set $\mathcal{F}$ of inputs $(\alpha_0, \ldots, \alpha_{n-1})$ consistent with such fixing and $|\mathcal{F}| \geq 2^{(1-2\varepsilon-\frac{2}{q})n \log |\mathbb{F}|}$. Therefore, the scheme E is (ε', r)-encoding scheme. This finishes the proof that $t^{\mathbb{F}}_{\mathsf{Eval}}(s) \geq \Omega(\log n / \log \log n)$.

Essentially, the same proof can be used to prove the lower bound for $t^{\mathbb{F}}_{\mathsf{Interp}}(s)$. Note that, the data structure $\mathcal{D}$ is used only for evaluating some polynomials in powers of the primitive root σ, i.e., computing entries of $\mathrm{FFFT}_{n,\sigma}(\alpha_0, \ldots, \alpha_{n-1})$. However as discussed in Section 4.2.2, it holds that $\mathrm{FFFT}_{n,\sigma} = n \cdot \mathrm{FFFT}^{-1}_{n,\sigma^{-1}}$. Moreover, entries of $\mathrm{FFFT}^{-1}_{n,\sigma^{-1}}(\beta_0, \ldots, \beta_{n-1})$ can be computed by a data structure for the polynomial interpolation. Thus, we may replace both uses of the data structure for the polynomial evaluation with a data structure for the polynomial interpolation. Therefore, we can use a data structure for the interpolation as $\mathcal{D}$ and with slight changes of R and E, we would get again an (ε', r)-encoding scheme.

□

5. Barrington Plays Cards

5.1 Introduction

A study of card-based protocols as a means for secure two-party computation was initiated by den Boer [33]. In this scenario, we have two players – Alice and Bob – who hold inputs x and y, respectively. Their goal is to securely compute a given function f on those inputs. By secure computation, we mean that the players learn nothing from observing the computation except for what is implied by the output $f(x, y)$. Den Boer introduced a model where the inputs x and y are encoded by a sequence of playing cards and the players operate on the cards to compute the function. They can use additional cards for computation. In particular, den Boer showed how to securely compute AND of two bits using five cards in total.

Crépeau and Kilian [30] improved this results. They represent each input bit by two face-down cards: 1 is represented as ♡♣, and 0 as ♣♡. They provided a secure protocol for AND which takes two bits b_1 and b_2 represented by two face-down cards and outputs $b_1 \wedge b_2$ represented again by two face-down cards. Since NOT can be obtained by swapping the two cards representing a given bit this allows us to use their technique to compute any function. This allows us to evaluate any Boolean circuit on the inputs by a protocol of length proportional to the size of the circuit and using a number of auxiliary cards that corresponds to the *width* of the circuit.

Nishida et al. [81] reduced the number of auxiliary cards to 6 for any Boolean function. For most functions, the protocol will be of exponential length as it essentially evaluates the DNF of f. Several other works studied the number of cards necessary for computing various elementary functions such as AND and XOR [39, 62, 59, 56, 81, 98, 78, 75, 77, 62, 2, 59].

Motivated by the question of what can be efficiently computed by such protocols and how many cards one needs to compute various functions, in this work, we investigate secure *efficient* protocols that are protocols of polynomial length. The result provided in this chapter is two-fold: We classify a large class of protocols with respect to the computational complexity of functions they compute, and we propose other encodings of inputs that require fewer cards than the 2-card representation. We summarize our results next:

1. We show that *oblivious* protocols of polynomial length that do not modify their input (they are *read-only*) and use only a constant number of auxiliary cards compute precisely the functions in NC^1, the class of functions computed by Boolean circuits of logarithmic depth. (Alternatively, NC^1 is the class of functions computed by Boolean formulas of polynomial size.) By oblivious protocol we mean a protocol whose actions depend only on the current visible state.

2. Oblivious read-only protocols of polynomial length with a logarithmic number of auxiliary cards correspond to the class of functions computable by polynomial-size branching programs. (This class is also known as $\mathsf{L}/poly$, the non-uniform version of deterministic log-space.)

3. We also investigate protocols that use a constant number of auxiliary cards but are allowed to use the cards representing the input for their computation provided that they guarantee that by the end of the computation the input will be restored to its original value. We show that such protocols can compute functions that are believed to be outside of NC^1. For example, they can compute languages that are complete for NL, the non-deterministic log-space. Hence, read-only protocols are presumably weaker than protocols that may modify their input.

4. We study alternative encodings of inputs that are more efficient than the 2-card encoding. We look at 1-card encoding where 1 is represented by ♡ and 0 by ♣. In this encoding, Alice and Bob need only one card per bit to commit the bit. We show similar complexity results for this encoding as for the 2-card encoding: read-only protocols with a constant number of auxiliary cards are NC^1, with a logarithmic number of cards it is the non-uniform log-space, and if we allow using the input cards for computation we get potentially more powerful protocols.

   A disadvantage of the 1-card protocol is that it still needs a supply of n cards ♡ and n cards ♣ to represent any n-bit input. Although, if one restricted his attention to inputs that contain the same number of 1's and 0's, it would suffice to have $n/2$ cards ♡ and $n/2$ cards ♣. Such inputs form a substantial fraction of all n-bit inputs, they are $\Theta(2^n/\sqrt{n})$ many.

5. We propose a new 1/2-card encoding which requires only $n/2$ cards ♡ and $n/2$ cards ♣ to represent any n-bit input. The 1/2-card encoding is obtained from the 2-card encoding by removing from each pair of cards one card, in total one half of the ♡-cards and one half of the ♣-cards. There is an empty space left instead of each removed card. There is a way for each player to encode his input so that the other player learns no information about the opponent's input. We show that using this encoding we can simulate any read-only protocol that uses 2-card encoding. Hence, any NC^1 function on n bits can be securely computed using only $n + O(1)$ cards, counting also the input cards. We do not know how to securely perform protocols for 1/2-card encoding that would modify their input.

5.1.1 Previous Work

As mentioned above, a study of card-based protocols was initiated by den Boer [33] who introduced a secure 5-card protocol for computing AND. However, this protocol does not produce output in a face-down 2-card format, thus it can not be used for designing protocols for arbitrary function. Since then a lot of work was done in improving AND protocols and other primitive functions. Crépeau and Kilian [30] provided a 1-party card-based protocol where the player can pick a random permutation π with no fixed point and the player has no information about π. They introduced an AND protocol, which takes two bits b_1, b_2 represented in the 2-card format as input and outputs two cards which represent $b_1 \wedge b_2$ in face-down 2-card format. They also introduced a protocol for copying a bit in the 2-card format, which is used during the simulation of circuits. Their protocols with a protocol for NOT (which is trivial) can be used for computing

any Boolean function f and the number of used cards is at most linear in the size of a circuit (using AND and NOT gates) computing f. However, during the computation, they use helping cards of other suits than ♣ and ♡. Niemi and Renvall [79] improved it and they designed a protocol computing arbitrary function f which uses only cards of suits ♣ and ♡. They also introduced a protocol to copy a single card with almost perfect security – the card suit is revealed only with a small probability. Such protocol cannot exist with perfect security as was proved by Mizuki and Shizuya [76]. The copying and AND protocols were further improved and simplified in [98, 78, 75, 77, 62, 2, 59].

Nishida et al. [81] proved that any Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ can be computed with $4n$ cards encoding the input and 6 additional helping cards. Mizuki [74] proved that at most $2n + 2$ is needed to compute AND of n bits. Francis et al. [39] provided protocols and tight lower bounds on the number of cards needed for computing any two-bit input two-bit output function. Other lower bounds for AND of 2 bits and of n bits in various regimes were provided by Koch et al. [62, 59] and by Kastner et al. [56].

The security of card-based protocols is provided by shuffling the cards – one player shuffles the cards (applies some random permutation to them) in a way so that the other player has no information about the new order of shuffled cards. Koch et al. [62] provided a 4-card AND protocol. However, they used a non-uniform distribution for picking a random permutation, which is difficult to perform by humans. Nishimura et al. [82] suggested an "easy-for-human" procedure how to apply a shuffling permutation picked from a non-uniform distribution using envelopes.

Koch and Walzer [60] studied private function evaluation. In this setting, Alice has an encoding of some program P which computes a function f. They studied various models, thus the program P can be a Turing machine, RAM, a circuit, or a branching program. Bob has an encoding of a string x and their goal is to compute securely $f(x)$. (Again, after the computation they know only $f(x)$ and they have no other information about each other input, i.e., Alice has no other information about x and Bob has no other information about f.) They proved that such universal computation exists for those models. They did not study the complexity of such simulations. Whereas, we study the complexity of computing a function f when the input to f is divided between Alice and Bob.

One can distinguish two types of attack.

1. Passive: honest-but-curious player – she follows the protocol but she wants to retrieve as much information as possible about the other player input.

2. Active: malicious player – she can deviate from the protocol.

Koch and Walzer [61] proved that if a passive-secure protocol Π uses only uniform closed shuffles (each shuffling permutation is picked uniformly from some permutation group) then the protocol Π can be transformed into an active-secure protocol.

Organization. In the next section, we provide formal definitions needed to state and prove our results. In Section 5.3, we prove the main result of this chapter, i.e., characterization of the circuit class NC^1 by card-based protocols. In

Section 5.4, we study connections between Turing machines and card-based protocols and differences between read-only and read-write protocols. In Section 5.5, we study more efficient encodings than the 2-card encoding.

5.2 Preliminaries

5.2.1 Card-based Protocols

In this section, we define *card-based protocols* which securely compute some Boolean function on a joint input of Alice and Bob. Alice gets an input $x \in \{0,1\}^n$ and Bob gets an input $y \in \{0,1\}^n$, and their goal is to compute $f(x,y)$ for some function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, while not revealing anything about their input to the other player. The protocol proceeds first by Alice and Bob committing their input into a sequence of cards, and then operating on the cards together with some auxiliary cards. At the end of the protocol, the players learn the output $f(x,y)$.

In this section, we consider the usual *2-card encoding* of the input, where each input is represented as a sequence of cards, two cards per bit: value 1 is represented by ♡♣ and value 0 is represented by ♣♡ where the cards are put face-down on the table. Hence each player needs $2n$ cards to commit his input. All the cards have the same back. In the beginning, face-down cards representing the player inputs are in front of the players. Between them, there is a *deck* of s prescribed auxiliary cards of ♡ and ♣. There is available some empty space on the table to operate with the cards. We assume that the cards are placed on the table in some specific positions (locations), numbered $1, \ldots, m$, where:

- $1, \ldots, 2n$ are positions of Alice's input cards,
- $2n+1, \ldots, 4n$ are positions of Bob's cards,
- $4n+1, \ldots, 4n+s$ are the initial positions of the helping cards in the deck,
- $4n+s+1, \ldots, m$ are initially empty positions.

We call the positions $1, \ldots, 4n$ as the input positions and the remaining positions as the *work space*. We say a position is *occupied* if there is a card on it, otherwise, it is *empty*. Let $q = m - 4n$ denote the amount of the work space. We assume $q = O(s)$. Thus, there are $4n+s$ cards on the table and $4n+q = m$ positions.

The players can move their input cards and cards from the deck to the work space and back. Formally, the basic *actions* which can be executed by the players are:

Move(p,i,j)**:** The player p moves a card from the position i to position j.

Shuffle(p,T,Γ)**:** The player p applies a random permutation from Γ to the cards on the table on positions $T \subseteq \{4n+1, \ldots, m\}$.

Turn(p,i)**:** The player p turns the i-th card on the table face-up if it is face-down, and vice versa.

The protocol specifies which action to take next based on the sequences of visible states seen on the table so far. The current visible state of the table is what an external viewer could observe, that is which positions are currently occupied and what is the top of each card laying on the table. If there are c distinct cards then there are at most $(c+2)^m$ distinct visible states. Hence, based on the sequence of visible states from the beginning of the game the protocol specifies which action to take next or whether to end. In the end, the protocol specifies which cards represent the output of the run of the protocol. (They might be face-down.) We say a protocol Π *computes* a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ if for all inputs $x, y \in \{0,1\}^n$, on the inputs x and y the protocol outputs $f(x,y)$. The length of the protocol is the maximum number of actions executed by the protocol over all inputs $(x,y) \in \{0,1\}^n \times \{0,1\}^n$ and all possible outcomes of shuffling. We say that a protocol is *oblivious* if the action executed next depends only on the current visible state and the number of actions taken so far.

The shuffling operation provides randomness for the execution of the protocol. Hence, the sequence of visible states the protocol passes through is a random variable. We will say that a protocol is *secure* if for any pair of inputs (x,y) and (x',y') to Alice and Bob, where $f(x,y) = f(x',y')$, the distribution of the sequence of visible states of the protocol on inputs (x,y) and (x',y') is the same. Notice, that this implies that neither of the players learns anything about the input of the other player except for what is implied by $f(x,y)$.

Often we will be interested in protocols that provide their output encoded in face-down cards. In such a scenario we will require for the security of the protocol that the distributions of visible states during the protocol will be identical for all input pairs (x,y).

We say the protocol is *robust* if a cheating player, that is a player who deviates from the protocol, is either caught by reaching an invalid visible state (where cards have unexpected values or positions) or the distribution of visible states does not leak any information about the other player input except for what would be leaked by honest players. In particular, if say Bob is cheating and Alice is honest, for a robust protocol we require that for any two inputs x, x' of Alice and any input y of Bob, where $f(x,y) = f(x',y)$, the distribution of the sequence of visible states during the game on inputs (x,y) and (x',y) is the same. We will be designing only robust oblivious protocols.

We say the protocol is *read-only* if the value of cards placed on the input positions $1, \ldots, 4n$ is always the same whenever a position is occupied.

Let s-SP be the class of function families $\{f_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\}_{n \geq 0}$ for which we have a sequence of secure read-only oblivious protocols, one for each n, which are of length polynomial in n, with deck size s and work space size $2s$. (At the beginning the first s work space positions are occupied by the deck of cards, and the remaining s positions are empty). We might allow s to be a function of n. We define a class of secure polynomial-length protocols $\mathsf{SP} = \bigcup_{s \geq 1} s$-$\mathsf{SP}$. That is, a function belongs to SP if it has polynomial length protocols which use a constant number of auxiliary cards and constant size work space.

5.2.2 Branching Programs

A *branching program* B is a directed acyclic graph G such that each vertex has an out-degree either 2 or 0. The set of edges E of the graph G is split into two sets, zero-edges E_0 and one-edges E_1, in such a way that every vertex v of out-degree 2 is incident to exactly one outgoing zero-edge and exactly one outgoing one-edge. Each vertex of out-degree 2 is labeled by an index $\ell \in [n]$, by $[n]$ we mean a set $\{1, \ldots, n\}$.

A branching program B is *layered* if the vertices are partitioned into layers $L_1, \ldots, L_d$. The edges go only from a layer L_i to a layer L_{i+1} (for all $i < d$). Vertices of out-degree 0 are exactly vertices in the layer L_d. The number of layers d is the *length* of B and the *width* w of B is the maximum size of its layers, i.e., $w = \max_i |L_i|$.

A layered branching program is *oblivious* if vertices in the same layer have the same label. A branching program is a *permutation branching program* if each layer has exactly w vertices and for every two consecutive layers L_j and L_{j+1} zero-edges and one-edges form matching M_j^0 and M_j^1, respectively. We can view the matchings M_j^0 and M_j^1 as two permutations $\pi_j^0, \pi_j^1 : [w] \to [w]$. Note that we can rearrange all layers such that all permutations π_j^0 are identities.

One vertex of in-degree 0 is an *initial* vertex $\bar{v}$. Some vertices of out-degree 0 are denoted as *accepting* vertices. The computation of a branching program B on an input string $x \in \{0,1\}^n$ proceeds as follows. It starts in the initial vertex $\bar{v}$ which is the first *active* vertex. Suppose v is an active vertex and $\ell \in [n]$ is the label of v. If the out-degree of v is 2, then the next active vertex is determined by the zero- or one-edge according to the value of x_ℓ. More formally, let $e = \{v, v'\}$ be the edge in E_{x_ℓ}. Then, the vertex v' is the new active vertex. We repeat this procedure until a vertex u of out-degree 0 is reached. An input $x \in \{0,1\}^n$ is accepted if and only if u is an accepting vertex. The branching program B computes a function $f : \{0,1\}^n \to \{0,1\}$ if it accepts exactly those $x \in \{0,1\}^n$ such that $f(x) = 1$.

The class of functions PB contains all the functions computable by layered branching programs of constant width and polynomial length. A permutation branching program is *restricted* if it has exactly one accepting vertex v_{acc} and exactly one rejecting vertex v_{rej} in the last layer L_d. The computation of a restricted permutation branching program ends always in the vertices v_{acc} or v_{rej} and it accepts an input if it ends in the accepting vertex v_{acc}. A class w-PBP contains Boolean functions which are computable by restricted permutation branching programs of width w and polynomial length. We use the famous Barrington's theorem [13], which says that constant-width (permutation) branching programs are as powerful as NC^1-circuits.

Theorem 5.1 (Barrington [13])**.** $\mathsf{PB} \subseteq \mathsf{NC}^1 \subseteq 5\text{-}\mathsf{PBP}$.

5.3 Simulating Branching Programs

In this section, we prove one of the main theorems of this chapter that read-only oblivious protocols of polynomial length that use constant work space compute the same functions as polynomial-size constant-width branching programs.

Theorem 5.2. $\mathsf{SP} = \mathsf{NC}^1$.

To simulate a branching program by SP-protocol we need an oblivious implementation of copying a bit in the committed 2-card format. We use a procedure by Crépeau and Kilian [30]. It is straightforward to implement the procedure to be oblivious. We state the proof of the following theorem for the sake of completeness.

Theorem 5.3. *There is a secure oblivious protocol that takes a bit b in 2-card representation placed in the work space and produces two 2-card copies of the bit in the work space. The protocol needs an auxiliary deck with three cards ♡ and three ♣ with the same back as the input bit.*

Proof.

1. Alice arranges the cards from the auxiliary deck face-up to create the following configuration.
$$\underbrace{? \; ?}_{b} \heartsuit\clubsuit\heartsuit\clubsuit\heartsuit\clubsuit$$
2. She turns the last six cards. Both, Alice and Bob, apply a random cyclic shift (denoted by $\langle, \rangle$) to them.
$$\underbrace{? \; ?}_{b}\langle\underbrace{? \; ?}_{b'}\;\underbrace{? \; ?}_{b'}\;\underbrace{? \; ?}_{b'}\rangle$$
3. They apply a random cyclic shift to the first four cards.
$$\langle ? \; ? \; ? \; ? \rangle \underbrace{? \; ?}_{b'}\;\underbrace{? \; ?}_{b'}$$
4. She turns the first four cards face-up.
   (a) If the sequence is alternating (i.e., ♡♣♡♣ or its shift) then $b = b'$. Thus, the last 4 cards represent two copies of b.
$$\heartsuit\clubsuit\heartsuit\clubsuit\underbrace{? \; ?}_{b}\underbrace{? \; ?}_{b}$$
   (b) Otherwise (i.e., ♡♣♣♡ or its shift) then $b = 1 - b'$. Thus, the last 4 cards represent two copies of negation of b. In that case, she switches the fifth with the sixth card and the seventh with the eighth card to represent two copies of b as well.
$$\heartsuit\clubsuit\clubsuit\heartsuit\underbrace{? \; ?}_{1-b}\underbrace{? \; ?}_{1-b}$$

   Alice and Bob might want to turn over and shuffle the first four left-over auxiliary cards after step 4. The last four cards represent two copies of b face-down in the 2-card format. To make the protocol oblivious we must implement both 4.a) and 4.b) by the same number of actions. To do so we include additional actions in 4.a) which have no effect such as shuffling a single card.

The only step where Alice or Bob can gain some information about b is Step 4, when Alice turns some cards. However, the cyclic shifts in Steps 2 and 3 were done by both players. Thus, Alice reveals the alternating sequence (♡♣♡♣) in Step 4 with the probability exactly $\frac{1}{2}$ (independently on the value of b) even if one of the players would be cheating. Thus, the protocol is secure.

□

To prove $\mathsf{NC}^1 \subseteq \mathsf{SP}$ we use as the first step Barrington's theorem [13]. By Barrington's theorem, each function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ from NC^1 can be computed by a polynomial length width-5 restricted permutation branching program. We will build a protocol that simulates the actions of the branching program layer by layer. We will keep track of the image of the initial vertex of the branching program. For that, we will use five cards ♡♣♣♣♣, where the position of ♡ corresponds to the image of the initial vertex (the active vertex), and we will apply the permutations prescribed by the branching program on those five cards. If the input variable assigned to a particular level of the branching program is set to 1 we are expected to perform the permutation otherwise we are supposed to do nothing, i.e., apply the identity permutation. Any permutation can be decomposed into a sequence of simple transpositions (swaps) so we will use swaps conditioned by the input variable to either permute the five cards or leave them the way they are. We will implement the following primitive: Alice and Bob want to conditionally swap two cards α, β according to the value of bit b represented in the face-down 2-card form in the work space without revealing the value of b. They also want to make sure that if $b = 1$ the swap occurs and if $b = 0$ the swap does not occur.

Theorem 5.4. *Let $\tilde{\alpha}, \tilde{\beta}$ be two sequences of face-down cards of the same length in the work space, and let $\gamma\,\delta$ be a face-down 2-card representation of b in the work space. There is a secure oblivious protocol such that during the protocol players swap the sequences α and β if and only if $b = 1$. The protocol uses two auxiliary cards ♣.*

Proof. The swapping protocol works as follows.

1. Alice rearranges the input cards together with two auxiliary face-up cards ♣ as follows:

   $$\clubsuit\gamma\ \tilde{\alpha}\ \clubsuit\delta\ \tilde{\beta}$$

   Thus, if $b = 0$ we have ♣♣$\tilde{\alpha}$♣♡$\tilde{\beta}$ and if $b = 1$ we have ♣♡$\tilde{\alpha}$♣♣$\tilde{\beta}$. The players do not know which situation are they in.

2. Both, Alice and Bob, apply a random cyclic shift to the cards, e.g.:

   $$\tilde{\alpha}\ \clubsuit\delta\ \tilde{\beta}\ \clubsuit\gamma$$

3. Alice turns the cards γ and δ representing b face-up. She knows what cards to turn, as the cards γ and δ are preceded by ♣ face-up. At the end she reorders the sequence (keeping the cyclic order) so that ♣♣ are the first cards, e.g.:

   $$\tilde{\alpha}\clubsuit\heartsuit\tilde{\beta}\clubsuit\clubsuit \to \clubsuit\clubsuit\tilde{\alpha}\clubsuit\heartsuit\tilde{\beta}$$

If $b = 0$ then $\gamma\delta = \clubsuit\heartsuit$ and the sequences $\tilde{\alpha}\tilde{\beta}$ are not swapped. On the other hand if $b = 1$ then $\gamma\delta = \heartsuit\clubsuit$ and the sequences are swapped.

Note that the cards in $\tilde{\alpha}$ and $\tilde{\beta}$ are face-down during the whole protocol. It is also clear that this is a robust protocol, and it can be implemented obliviously. In Step 3 the cards δ and γ representing the bit b are revealed. However, because of random cyclic shifts in Step 2 (again done by both players), these cards are in the order $\heartsuit\clubsuit$ with probability $\frac{1}{2}$, independently of the value of b. Thus the swapping protocol is secure.

□

Remark. We point out that there is a more efficient swap protocol that does not use any auxiliary card. In step 2 Alice and Bob could perform a random bisection cut introduced by Mizuki and Sone [78]. During the random bisection cut, they split the deck into two halves of the same number of cards and then they swap them or not (at random). Thus, in step 3 of the swap protocol, they would know which cards they should turn, even without the face-up $\clubsuit$ cards. Mizuki and Sone [78] also introduced a more efficient copy protocol (it uses 6 cards instead of 8) but it also uses the random bisection cut. We prefer the standard random cut as it is easier to cheat during the random bisection cut.

Now, we prove the first inclusion of Theorem 5.2.

Theorem 5.5. 5-PBP $\subseteq$ SP.

Proof. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function in 5-PBP. Then, the function f can be computed by a branching program P with the following properties:

1. Each layer has exactly 5 vertices. The input vertex is the first vertex in the first layer. The computation ends either in the accepting vertex v_{acc} or in the rejecting vertex v_{rej}.

2. The permutation from each layer i to the layer $i + 1$ corresponding to 0 is the identity.

Alice and Bob represent the first layer as $\heartsuit\clubsuit\clubsuit\clubsuit\clubsuit$, each card represents one vertex in the layer. The card $\heartsuit$ represents the active vertex in the layer (the initial vertex in the first layer). We call these 5 cards the *program cards*. Alice and Bob put the program cards in the work space and turn them face-down. Alice and Bob simulate the program P layer by layer. They apply permutations determined by P to the program cards according to the player's input bits. Suppose we have a representation of the active vertex in the i-th layer and we want to calculate the active vertex at the $(i + 1)$-th layer. Without loss of generality the label of the i-th layer is Alice's bit x_ℓ (otherwise the roles of Alice and Bob are reversed). Thus, we want to apply some permutation $\rho_i \in S_5$ to the program cards if $x_\ell = 1$ and keep the order of the program cards if $x_\ell = 0$.

We decompose the permutation ρ_i into transposition $\tau_1 \circ \cdots \circ \tau_r$. For $j = 1, \ldots, r$, Alice will apply the transposition τ_j to the program cards. She runs the protocol Γ_1 of Theorem 5.3 to get cards γ, δ representing her bit x_ℓ in the work

space. More formally, after the execution of Γ_1, there are two pairs of cards such that each pair represent the bit x_ℓ in the 2-card format. She puts one pair back to the input positions, i.e., the protocol Π is indeed read-only. We denote the cards of the second pair as γ and δ. Alice will use them for a conditional swap. She runs the protocol Γ_2 given by Theorem 5.4 (applied to the cards γ, δ and to the two program cards which should be affected by the transposition τ_j). That is, Alice swaps the two cards that τ_j is acting on if and only if $x_\ell = 1$. After applying this procedure for all transpositions $\tau_1, \ldots, \tau_r$, the permutation ρ_i got applied to the program cards if and only if $x_\ell = 1$ (otherwise the order of the program cards does not change).

Alice and Bob repeat this procedure for each layer of the branching program. Let α be the card representing the accepting vertex v_{acc} and β be the card representing the rejecting vertex v_{rej} at the end of the simulation. The cards α, β represent the output of the program. If the input is accepted, then the accepting vertex v_{acc} is active at the end of the simulation and thus the card α has suit $\heartsuit$ and the card β has suit $\clubsuit$. Thus, the cards α, β represent 1. On the other hand, if the input is rejected, then the rejecting vertex v_{rej} is active. Thus, the cards α, β have suits $\clubsuit$ and $\heartsuit$, respectively, and they represent 0.

The protocol Π is clearly SP-protocol as the players only sequentially apply the copying protocol Γ_1 and the swapping protocol Γ_2 to the program cards. We claim the simulation protocol Π is secure. Both protocols Γ_1 and Γ_2 are secure. The other helping cards which are not used only during a single run of Γ_1 or Γ_2 are exactly the program cards. They are placed face-up from the deck and then turned face-down for the rest of the protocol. Thus, the program cards could only reveal the output $f(x, y)$ at the end of the protocol (if they are turned face-up) and no other information. Therefore, we conclude that protocol Π is secure.

□

Now, we prove the opposite inclusion of Theorem 5.2.

Theorem 5.6. $\mathsf{SP} \subseteq \mathsf{PB}$.

Proof. Consider a family of functions $\left\{f_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\right\}_{n \geq 0}$ for which we have a sequence of secure read-only oblivious protocols, one for each n, which are of length polynomial in n, with deck size s and work space size $2s$. Let c be the number of different cards used by the protocol. At any moment, the work space can be in at most $(2c+1)^{2s}$ different states which we call the internal states of the protocol. For any $n \geq 1$ we will build a width-$(2c+1)^{2s}$ branching program of the same length T_n as the protocol for f_n. Each layer of the branching program consists of vertices where each vertex corresponds to one internal state of the protocol. We need to define edges between the layers of the branching program.

Let v be a vertex at layer $t \in \{0, \ldots, T_n - 1\}$. It corresponds to some internal state which in turn determines a visible state that together with t determines the action taken by the protocol at such a state. We define the edges based on the type of that action. If the action is a move of a card from some input position into the work space then the node v queries the value of the corresponding input variable and the outgoing edges lead to nodes corresponding to internal states that reflect a move of the card into the work space. If the action is a shuffle

operation then the node v queries variable x_1 and irrespective of its value both outgoing edges go to the node in the next layer corresponding to the internal state obtained by applying one of the allowed permutations. (The particular choice of the permutation does not matter.) Similarly, for a turn of a card or a move of a card within the work space or out from the work space, the edges will go into a node that reflects the internal state after the move. Note that during an execution of a secure read-only protocol, the players never turn any card at an input position. Thus, the suit of a turned card is always determined by the current internal state. Therefore, we can add edges for a turn action without querying any specific input bit. In the last layer, we designate vertices that correspond to accepting states of the protocol as accepting all other nodes will be rejecting.

It should be clear from the construction that the resulting branching program computes f_n and has the required properties.

□

Theorem 5.2 is a corollary of Theorems 5.5 and 5.6 and Barrington's theorem (Theorem 5.1).

5.4 Simulating Turing Machines

In this section, we will look at computation that obliviously and securely computes on committed inputs in 2-card representation. The exact split of the input between Alice and Bob is irrelevant in this section so we assume that the total length of the input is n bits. The protocols are expected to preserve the committed inputs: Although they may be allowed to modify the committed input during the computation, by the end of the computation the committed input must be restored to its original form. The protocols do not leak any information about the committed inputs except for what can be derived from the output cards if they are inspected. The protocols can be carried out by either player. To guarantee robustness and security shuffle operations should be always done by both players. (We use only uniformly random shuffle and random cyclic shift so performing them twice does not change their output distribution.)

Theorem 5.7. *Let $s(n) \geq \log n$ be a non-decreasing function. Let f be a function computable by a polynomial-time Turing machine in space $s(n)$. Then f is in $O(s(n))$-SP.*

Let $SEL : \{0,1\}^3 \to \{0,1\}$ be the selection function that $SEL(c,b,a) = a$ if $c = 0$ and $SEL(c,b,a) = b$ otherwise.

Proof. We describe the algorithm for the protocol in high-level form and leave details of the construction to the interested reader. Let f be computable by a Turing machine M. Without loss of generality, we assume M uses a binary alphabet, on inputs of length n it uses work space exactly $s(n)$ bits and computes for $t(n)$ steps. The output of M is determined by the first bit of its work tape. We will simulate the computation of M step by step.

The protocol will use $8s(n) + 4\log n + O(1)$ auxiliary cards. Two blocks w and w' will represent $2s(n)$ bits, each, and two blocks p and j will represent $\log n$

bits each. In addition to that there is a block q of $O(1)$ bits, and some additional auxiliary bits. We need a constant number of positions to be empty. All the bits are encoded in the 2-card representation. The block w represents the content of the work tape of M, 2 bits per tape cell, where the second bit indicates the presence of the work tape head on that particular tape cell. The block p encodes in binary the current position of the input head of M. The block q encodes the internal state of M.

The protocol simulates one step of M as follows: first, it determines the value b of the bit scanned by the input head, then it calculates into w' the content of the work tape of M after this step. Then, it updates the internal state, the input head position p, and switches w' and w.

To determine b the protocol looks at each input bit x_i one by one and records the one that has an index corresponding to p. Set b to 0. For $i = 1, \ldots, n$, the protocol copies x_i into some work space b', it sets j to represent i, obliviously compares p and j while recording the result into c. (Comparing bit strings can be done by an NC^1 circuit so there is an oblivious protocol for that of poly-logarithmic length.) From c, b and b' we can calculate the new value of b by evaluating $SEL(c, b, b')$. This can be done obliviously. After processing all the input bits, b has the value of the currently scanned input bit.

Now, we can determine w', the content of the work tape of M after this step of the computation. We compute w' cell by cell. The value of each cell is a function of the input bit b, M's state q, and the previous content of the cell in w together with the content of adjacent cells. Hence, the value of each bit of w' is a function of constantly many bits and can be computed obliviously.

After computing w', we can also calculate d, the direction in which the input head of M should move, and the new state q' of M. This can be done by scanning w for the work tape position, and recording the relevant information for q' and d when we pass over the current work cell similarly to determining the value of the input bit b.

From p and d, we obliviously calculate the next position of the input head into j. (Each bit of j can be computed by an NC^1 circuit from p and d.) Finally, we switch the contents of w and w', p and j, and q and q'.

We repeat this procedure $t(n)$ times. In the end, the first bit of w indicates the output of M. As each step of the computation can be implemented securely, obliviously, and robustly, we obtain a secure, oblivious, and robust protocol for f that uses $O(s(n))$ work space and $O(s(n))$ auxiliary cards.

□

Remark. Note that in the proof of the previous theorem, the length of the protocol corresponds to the time of the simulating Turing machine and we do not need the assumption of the polynomial time for the simulation. Thus, for a function f computable by a Turing machine in space $s(n)$ and time $t(n)$, we would get a protocol computing f of length $poly(t(n))$ and using $O(s(n))$ auxiliary cards.

As the card-based protocols allow for non-uniformity by protocols using work space of size $O(\log n)$ we can simulate not only log-space Turing machines but also polynomial-size branching programs (the *non-uniform log-space*). The above proof can be extended to Turing machines taking advice: the protocol can provide the advice bit by bit during the phase when the input is scanned bit by bit to

determine b. (We assume that the advice is provided to the Turing machine on bit positions with index $> n$. For those positions instead of copying the non-existent input bits, the protocol hardwires the appropriate bit into b'. As the advice is the same for each input, this can be done publicly.)

By essentially the same proof as Theorem 5.6 we can obtain a simulation of oblivious, read-only secure protocols that use a logarithmic amount of work space by branching programs of polynomial size. Let $O(\log n)$-$\mathsf{SP} = \bigcup_k (k + k \log n)$-$\mathsf{SP}$. We get:

Theorem 5.8. *The class of functions computable by polynomial-size branching programs equals to $O(\log n)$-SP.*

5.4.1 Read-write Protocols

So far we have looked only at read-only protocols. If we remove the condition to be read-only we get a potentially larger class of functions computable by such protocols. When the protocol is not read-only, we still require the protocol to restore its input into the original state by the end of the computation. We also require the protocol to be secure so not to leak any information about the input except for what is implied by the protocol output cards.

We give examples of functions that can be computed by protocols modifying their input which we conjecture are outside of the read-only protocol class with similar bound on the work space. Proving this conjecture would amount to separating NC^1 from log-space, a major open problem in complexity theory.

Let $s(n)$ be a non-decreasing function such that $\log n \leq s(n) \leq n/2 \log^* n$. Let $g : \{0,1\}^n \to \{0,1\}$ be in NC^1, and $h : \{0,1\}^{n - s(n)\log^* n} \to \{0,1\}$ be a function computable by a Turing machine in space $O(s(n))$ and polynomial time. Define $f : \{0,1\}^n \to \{0,1\}$ as follows:

$$f(x) = \begin{cases} g(x) & \text{if } x_{n+1-s(n)\log^* n} \cdots x_n \neq 0 \cdots 0, \\ h(x_1 \cdots x_{n-s(n)\log^* n}) & \text{otherwise.} \end{cases}$$

Theorem 5.9. *The function f defined above is computable by secure robust oblivious protocols of polynomial length that use a constant amount of work space.*

Proof. The protocol for f proceeds as follows. It first computes the OR of the input bits $x_{n+1-s(n)\log^* n} \cdots x_n$ using a protocol for NC^1 functions where the output c of the protocol is encoded in the 2-card representation in its work space. Then it computes the value g of $g(x)$ encoded in 2-card representation in the work space. Finally, it uses the cards representing input bits $x_{n+1-s(n)\log^* n} \cdots x_n$ to simulate the computation of a Turing machine for h as in the proof of Theorem 5.7. The simulation is done so that if $c = 1$ then nothing is done to the input (the simulation is vacuous) and if $c = 0$ the simulation is really happening. The simulation uses the input bits $x_{n+1-s(n)\log^* n} \cdots x_n$ to store w, w', p and j (from the simulation), everything else is done in the actual work space of constant size. Whenever the simulation wants to write some value a into an input position used for the simulation, it copies the value into the work space, it copies there the current value d of the destination position, computes $SEL(c, a, d)$, and replaces cards in the destination by the output of $SEL(c, a, d)$. (Hence, if $c = 0$ nothing

has happened.) Reading a value can be done by copying the particular bit into the work space and then working with the copied cards. This way the input is undisturbed if $c = 1$ and it is overwritten if $c = 0$. At the end of simulation, the protocol copies the output bit h, which is the first bit of w into the actual work space, and writes value 0 to all input bits $x_{n+1-s(n)\log^* n} \cdots x_n$ conditionally on $c = 0$. (Bit values read from the storage, taking part in the vacuous computation, will be the same throughout the computation. So the simulations of the NC^1 circuits implementing various steps of the computation will be secure.)

Finally, the protocol computes $SEL(c, h, g)$ which is the output of the protocol. All parts of the protocol can be done securely and obliviously. (This is true also when $c = 0$ and the simulation of the Turing machine is bogus.) The protocol restores its committed input by the end of the computation.

□

One can also use the technique of catalytic computation to construct protocols for functions not know to be in NC^1. Buhrman et al. [20, 25] show how to use memory that contains some information for computation while restoring the memory to its original content by the end of the computation. For example, they can solve the connectivity on directed graphs this way, the problem $CONN(G)$: Given an $n \times n$ adjacency matrix of a directed graph G decide whether there is a path from vertex 1 to vertex n. They present a polynomial-size program for $CONN(G)$, which uses $3n^2 + 1$ work registers and n^2 input registers, each holding one bit of information. The program consists of instruction of the form

$$r_i \leftarrow r_i \oplus u \cdot v,$$

where u and v are arbitrary registers different from r_i or constants 0 and 1. The program is oblivious, so it is a straight line program consisting of such instructions. The program has the property that all registers are guaranteed to have the initial value by the end of the computation except for one specified work register which contains the output value. It is straightforward to implement each such an instruction by secure and robust protocol since the instructions are computable in NC^1.

This allows to design an oblivious, secure protocol of polynomial length with constant work space for a function $f' : \{0,1\}^{4n^2} \to \{0,1\}$ that is defined as:

$$f'(G_1, G_2, G_3, G_4) = 1$$

if and only if from the vertex 1 we can reach the vertex n in each of the graphs represented by adjacency matrices G_1, G_2, G_3 and G_4. Such a function is unlikely to be contained in NC^1, as $CONN(G)$ is known to be complete for *nondeterministic* log-space computation. Hence, it is unlikely that protocols that are allowed to modify their input could be simulated by read-only protocols using similar resources.

5.5 More Efficient Input Encodings

5.5.1 1-Card Encoding

In this section, we consider other ways how Alice and Bob can commit their input which use fewer cards. The first natural encoding is to represent each bit 1 by face-down card ♡ and bit 0 by ♣. These cards would be stored in front of the players in input positions $1, \ldots, 2n$. Whenever the players want to operate with the committed bit they need to extend it to the 2-card representation.

There are two ways we know how to do it. Niemi and Renvall [79] gave a protocol that is able to extend the bit without knowing its value. However, there is a small probability of leaking the value of the bit being extended. The probability is inversely proportional to the number of cards used for the protocol. Hence, one would need a large number of helping cards in order to make sure that the probability of leaking information is negligible. That would erase any savings from the 1-card representation.

The other way which we use here is to allow the player who owns the particular input bit to extend it using a designated deck of two face-down cards ♣ and ♡. Once the bit is extended it can be copied by the protocol from Theorem 5.3, the first card of the first copy can be put back in the input position, the second card can be put back into the auxiliary deck, and the second copy can be used for further computation. The auxiliary deck containing the same cards as earlier should be shuffled by both players at the end of this procedure. The protocol is robust since a player cheating by extending the input bit by a wrong card will be caught in Step 4 of the copying protocol.

For this procedure, we need to augment our set of actions by the action of extending a bit by a complementary card from a designated deck. This action can be performed by shuffling the auxiliary deck, then peeking at the value of the card we are extending, and selecting the complementary card by peeking at each card in the deck.

With this operation in mind, we need to extend the definition of protocol security. We say a protocol is *secure from Alice* if for any pair of inputs (x, y) and (x, y') to Alice and Bob, the distribution of the sequence of visible states of the protocol together with the sequence of cards seen by Alice while peeking at them during the extension action on inputs (x, y) and (x, y') is the same. Similarly, the protocol is *secure from Bob* if for any pair of inputs (x, y) and (x', y) to Alice and Bob, the distribution of visible states and cards peeked at by Bob will be the same on both inputs (x, y) and (x', y). The protocol is secure if it is secure from both Alice and Bob.

Using the extension action we can perform all read-only protocols that used the 2-card bit commitment of inputs even for inputs committed in 1-card representation. They will be secure as long as the player performing each extension is the owner of the input bit as seeing his/her input bits does not affect the security definition. Hence, the power of the model stays essentially the same with this modification.

Security becomes more of an issue for protocols that are allowed to modify their inputs. Yet, we can prove a result similar to Theorem 5.9 for slightly modified function f'. Let $g : \{0,1\}^n \to \{0,1\}$ be in NC^1, and $h : \{0,1\}^{n-s(n)\log^* n} \to$

$\{0,1\}$ be a function computable by a Turing machine in space $O\big(s(n)\big)$ and polynomial time, where $\log n \leq s(n) \leq n/2\log^* n$. Define $f' : \{0,1\}^n \to \{0,1\}$ as follows:

$$f'(x) = \begin{cases} g(x) & \text{if } x_{n+1-s(n)\log^* n} \cdots x_n \neq 0101\cdots01, \\ h(x_1 \cdots x_{n-s(n)\log^* n}) & \text{otherwise.} \end{cases}$$

We assume $s(n)\log^* n$ is even, and the first half of $x_{n+1-s(n)\log^* n} \cdots x_n$ is held by Alice and the other half by Bob. The other bits can be split between the players arbitrarily.

Theorem 5.10. *The function f' defined above is computable by secure robust oblivious protocols of polynomial length that use a constant amount of work space and 1-card encoding of input bits.*

Proof. The protocol for f' proceeds similarly to the one in Theorem 5.9. It first verifies whether the input bits $x_{n+1-s(n)\log^* n} \cdots x_n$ differ from $0101\cdots01$ (assuming their number is even) using protocol for NC^1 functions. The output c of the verification is encoded in 2-card representation in the work space. Then, the protocol computes the value g of $g(x)$ encoded in the 2-card representation in the work space. Up until this point, we use the protocol described above to extend input bits into 2-card representations by the player who owns the input bit.

Now we want to use the cards representing input bits $x_{n+1-s(n)\log^* n} \cdots x_n$ to simulate computation of a Turing machine M for computing

$$h = h(x_1, \ldots, x_{n-s(n)\log^* n})$$

as in the previous proofs. We will use these input cards for storage when $c = 0$ and when $c = 1$ we will keep them intact. In the former case, we will eventually reset the input bits/cards to the initial state. Let I be positions of cards which represent bits $x_1 \cdots x_{n-s(n)\log^* n}$ at the beginning of the protocol. Thus, the cards on the positions I represent (in the 1-card encoding) the input for M. These cards will be in a read-only regime during the whole computation. Let J be the positions of cards representing $x_{n+1-s(n)\log^* n} \cdots x_n$. The cards on J will represent in 2-card encoding the content of tapes of M during the computation. Thus, $|J| = s(n)\log^* n$ but the cards on J will represent $\frac{1}{2}s(n)\log^* n$ bits. As $x_{n+1-s(n)\log^* n} \cdots x_n = 0101\cdots01$ (if $c = 0$), the cards on J represent $\frac{1}{2}s(n)\log^* n$ zeros at the beginning of the protocol.

The simulation proceeds in a similar way as the simulation in proof of Theorem 5.9. For reading bits from I that encode the input bits to M we use the same 1-card extension protocol as above to copy them into work-space. Now, we need procedures that will read and write bits in 2-card representations from positions J. However, if $c = 1$ some two consecutive positions would not represent a bit correctly (the two cards on them would have the same suit). The read/write procedures need to work and be secure even in this case. The players cannot simply inspect the cards on positions J because they may represent some intermediate results of the computation.

First, we describe how to read a bit b encoded in J. We want to create a 2-card representation of bit b in work space if $c = 0$ or a valid 2-card representation

of some bit if $c = 1$. The bit b is represented by 2 cards α, β on positions in J. Note that α and β can be of the same suit if $c = 1$. Suppose Alice owns the positions in J representing the bit b, the case of Bob's positions is symmetric. First, Alice will add complementary cards to α and β as follows. Alice prepares the sequence: $\clubsuit\clubsuit\alpha\clubsuit\heartsuit\beta$, then she turns the second and fifth card face-down to get a sequence

$$\clubsuit?\alpha\clubsuit?\beta.$$

Bob shuffles the six cards cyclically at random. Now, Alice extends the card preceding each $\clubsuit$ that is face-up (cards α and β) by a complementary card to the right taken from an auxiliary deck. Alice sees the suit of the cards α and β during this action but she does not know their actual order. Bob shuffles the cards cyclically again and turns face-up the cards following the two $\clubsuit$ that are face-up. Now, by a cyclic shift, they rearrange the cards so that they look like

$$\clubsuit\clubsuit\alpha\overline{\alpha}\clubsuit\heartsuit\beta\overline{\beta},$$

where $\overline{\alpha}$ and $\overline{\beta}$ are cards complementary to α and β, respectively. They can copy each of the card pairs $\alpha, \overline{\alpha}$ and $\beta, \overline{\beta}$ to verify that Alice used complementary cards and the pairs $\alpha, \overline{\alpha}$ and $\beta, \overline{\beta}$ indeed represent two bits in 2-card encodings.

By following this protocol, Alice learns whether the two cards α, β have the same suit or are distinct. If the suits are the same she also learns the suit. However, in that situation, $c = 1$ and she already knew all this information. In the later case, she knows that the bits at those input positions are distinct, but she knew that already. She does not learn their relative order because of the shuffle by Bob. Thus, she does not know whether they were altered since the beginning of the protocol or not.

To finish the read procedure, Alice copies the pair $\alpha, \overline{\alpha}$ (by the protocol of Theorem 5.3) to get two copies represented by cards $\alpha', \overline{\alpha'}, \alpha'', \overline{\alpha''}$. She returns the cards α'' and β back to the positions in J from which she moved the cards α and β at the beginning. The cards α' and $\overline{\alpha'}$ are used further in the computation. Other cards $(\overline{\alpha''}, \overline{\beta})$ are moved back to the auxiliary deck. If the cards α and β have different suits then the cards $\alpha', \overline{\alpha'}$ represent the bit b, as the card α and α' have the same suit. If the cards α and β have the same suit then the cards $\alpha', \overline{\alpha'}$ are a valid representation of some bit b'. However, in that case $c = 1$ and the value of b' is irrelevant for the computation. Bit values read from the storage, taking part in the bogus computation, will be consistent throughout the computation. Thus, the simulations of the NC^1 circuits implementing various steps of the computation are secure.

Now, we describe how to store a bit in 2-card encodings on to some positions in J. Again, suppose we want to store a bit b on to positions owned by Alice and occupied by cards γ_1 and γ_2. Let α and β be cards representing b. We want a procedure that will do the following. If $c = 0$ then the cards γ_1 and γ_2 are replaced by cards of the same suits as α and β, respectively. Otherwise, if $c = 1$ then the new cards need to have the same suits as γ_1 and γ_2. First, Alice will add complementary cards to γ_1 and γ_2 to get a sequence $\gamma_1, \overline{\gamma_1}, \gamma_2, \overline{\gamma_2}$ (she proceeds in the same was as in the read procedure above). Let d_1 and d_2 be bits represented by $\gamma_1, \overline{\gamma_1}$ and $\gamma_2, \overline{\gamma_2}$, respectively. She creates two copies of b, negates the second one, and computes $a_1 = \mathit{SEL}(c, b, d_1)$ and $a_2 = \mathit{SEL}(c, 1 - b, d_2)$. Let $\delta_1, \overline{\delta_1}$ and $\delta_2, \overline{\delta_2}$ be cards representing a_1 and a_2 respectively. If $c = 0$ then the cards δ_1, δ_2

represent the bit b. If $c = 1$ then the cards δ_1 and δ_2 have the same suits as the cards γ_1 and γ_2, respectively. Thus, Alice moves the cards δ_1 and δ_2 into the positions of the cards γ_1 and γ_2 and moves the rest of the cards to the deck.

To avoid leakage of information from the way Alice picks the cards from the auxiliary deck when picking a card of a particular suit, she proceeds as follows. She knows how many cards of that suit are in the deck. Thus, she shuffles the cards at random and then proceeds left to right to pick one of the cards of that suit uniformly at random. To achieve that she picks each card of the desired suit with probability $1/(k+1)$, where k is the number of unseen cards of the desired suit still in the deck. This process guarantees that Alice will pick a card from a completely random position.

In this way the protocol can use $x_{n+1-s(n)\log^* n} \cdots x_n$ to compute $h = h(x)$. After obtaining value h it outputs $SEL(c, h, g)$.

□

Hence, also in the case of the 1-card representation of the input one can take advantage of the input cards to compute functions that seem unattainable with read-only protocols.

5.5.2 1/2-Card Encoding

In the 1/2-card encoding we represent value 1 by either ♡× or ×♣, and value 0 by either ♣× or ×♡, where × represents an empty position. To commit her input Alice picks $n/2$ of her input bits, and for those input bits, she leaves the empty spot × in the position of ♡, for the remaining bits she leaves the empty spot in place of ♣ (in the 2-card encoding of the bit). This way, she uses exactly $n/2$ cards ♣ and ♡ to commit her input. It is easy to verify that for each bit there is exactly a 1/2 probability that the missing card will be on the left. Hence, the positions of the missing cards do not leak any information about her input. Bob proceeds in the same way to commit his input.

After committing their inputs they can run any read-only protocol similar to the case of 1-card encoding. Whenever an input bit is needed it is copied into a 2-card representation by essentially the same protocol as in the case of 1-card encoding. This means that we need only $n + O(1)$ cards to compute any NC^1 function on n-bit inputs.

We do not know how to implement protocols that could modify their inputs. Modifying an input bit would require either picking the empty spot in the representation at random (which could lead to using substantially more cards of each type) or reusing the cards that are there. In the latter case, we do not know how to do it without leaking information.

Bibliography

[1] A. Abboud and V. V. Williams, *Popular conjectures imply strong lower bounds for dynamic problems*, in Proceedings of the 55th FOCS, 2014.

[2] Y. Abe, Y.-i. Hayashi, T. Mizuki, and H. Sone, *Five-card and protocol in committed format using only practical shuffles*, in Proceedings of the 5th ACM on ASIA Public-Key Cryptography Workshop, 2018.

[3] H. Abusalah, J. Alwen, B. Cohen, D. Khilko, K. Pietrzak, and L. Reyzin, *Beyond hellman's time-memory trade-offs with applications to proofs of space*, in Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II, 2017.

[4] M. Adler, N. J. A. Harvey, K. Jain, R. Kleinberg, and A. R. Lehman, *On the capacity of information networks*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, 2006.

[5] P. Afshani, C. B. Freksen, L. Kamma, and K. G. Larsen, *Lower Bounds for Multiplication via Network Coding*, in 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), 2019.

[6] M. Ajtai, *A lower bound for finding predecessors in yao's cell probe model*, Combinatorica, 8 (1988).

[7] A. Ambainis, H. Buhrman, W. Gasarch, B. Kalyanasundaram, and L. Torenvliet, *The communication complexity of enumeration, elimination, and selection*, Journal of Computer and System Sciences, 63 (2001).

[8] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.

[9] G. Asharov, W.-K. Lin, and E. Shi, *Sorting short keys in circuits of size o (n log n)*, in Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), 2021.

[10] L. Babai, P. Frankl, and J. Simon, *Complexity classes in communication complexity theory*, in Proceedings of the 27th FOCS, 1986.

[11] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar, *An information statistics approach to data stream and communication complexity*, in Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS), 2002.

[12] B. Barak, M. Braverman, X. Chen, and A. Rao, *How to compress interactive communication*, in Proceedings of the 42nd Symposium on Theory of Computing (STOC), 2010.

[13] D. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in nc1*, Journal of Computer and System Sciences, 38 (1989).

[14] P. Beame, T. Pitassi, N. Segerlind, and A. Wigderson, *A strong direct product theorem for corruption and the multiparty communication complexity of disjointness*, Computational Complexity, 15 (2006).

[15] A. Beimel, S. B. Daniel, E. Kushilevitz, and E. Weinreb, *Choosing, agreeing, and eliminating in communication complexity*, Computational Complexity, 23 (2014).

[16] M. Ben-Or and P. Tiwari, *A deterministic algorithm for sparse multivariate polynomial interpolation*, in Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, 1988.

[17] M. Braverman and A. Rao, *Information equals amortized communication*, in Proceedings of the 52nd Symposium on Foundations of Computer Science (FOCS), 2011.

[18] M. Braverman and O. Weinstein, *A discrepancy lower bound for information complexity*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 2012.

[19] J. Brody and K. G. Larsen, *Adapt or die: Polynomial lower bounds for non-adaptive dynamic data structures*, Theory Comput., 11 (2015).

[20] H. Buhrman, R. Cleve, M. Koucký, B. Loff, and F. Speelman, *Computing with a full memory: Catalytic space*, in Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, 2014.

[21] A. Chakrabarti, R. Kondapally, and Z. Wang, *Information complexity versus corruption and applications to orthogonality and gap-hamming*, in Proceedings of the 16th RANDOM, 2012.

[22] A. Chattopadhyay, P. Dvořák, M. Koucký, B. Loff, and S. Mukhopadhyay, *Lower bounds for elimination via weak regularity*, in 34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany, 2017.

[23] A. Chattopadhyay, J. Edmonds, F. Ellen, and T. Pitassi, *A Little Advice Can Be Very Helpful*, in Proceedings of the 23rd SODA, 2012.

[24] M. Clausen, A. Dress, J. Grabmeier, and M. Karpinski, *On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields*, Theoretical Computer Science, 84 (1991).

[25] R. E. Cleve, *Methodologies for Designing Block Ciphers and Cryptographic Protocols*, PhD thesis, University of Toronto, 1989.

[26] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex fourier series*, Mathematics of computation, 19 (1965).

[27] S. Coretti, Y. Dodis, S. Guo, and J. Steinberger, *Random oracles and non-uniformity*, in Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2018 Proceedings, 2018.

[28] H. Corrigan-Gibbs and D. Kogan, *The function-inversion problem: Barriers and opportunities*, in Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I, 2019.

[29] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*, Wiley-Interscience, 2006.

[30] C. Crépeau and J. Kilian, *Discreet solitary games*, in Advances in Cryptology — CRYPTO' 93, 1994.

[31] I. Csiszar and J. Körner, *Information theory: coding theorems for discrete memoryless systems*, Cambridge University Press, 2011.

[32] A. De, L. Trevisan, and M. Tulsiani, *Time space tradeoffs for attacks against one-way functions and prgs*, in Advances in Cryptology – CRYPTO 2010, 2010.

[33] B. den Boer, *More efficient match-making and satisfiability the five card trick*, in Advances in Cryptology — EUROCRYPT '89, 1990.

[34] Y. Dodis, S. Guo, and J. Katz, *Fixing cracks in the concrete: Random oracles with auxiliary input, revisited*, in Advances in Cryptology – EUROCRYPT 2017, 2017.

[35] Z. Dvir, A. Golovnev, and O. Weinstein, *Static data structure lower bounds imply rigidity*, in Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019.

[36] A. Farhadi, M. Hajiaghayi, K. G. Larsen, and E. Shi, *Lower bounds for external memory integer sorting via network coding*, in Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019.

[37] U. Feige, J. Killian, and M. Naor, *A minimal model for secure computation*, in Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 1994.

[38] A. Fiat and M. Naor, *Rigorous time/space trade-offs for inverting functions*, SIAM Journal on Computing, 29 (1999).

[39] D. Francis, S. R. Aljunid, T. Nishida, Y.-i. Hayashi, T. Mizuki, and H. Sone, *Necessary and sufficient numbers of cards for securely computing two-bit output functions*, in Paradigms in Cryptology – Mycrypt 2016. Malicious and Exploratory Cryptology, 2017.

[40] A. Gál and P. B. Miltersen, *The cell probe complexity of succinct data structures*, in Automata, Languages and Programming, 2003.

[41] J. v. z. Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, 2013.

[42] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan, *Bounds on the efficiency of generic cryptographic constructions*, SIAM Journal on Computing, 35 (2005).

[43] O. Goldreich and R. Ostrovsky, *Software protection and simulation on oblivious rams*, Journal of the ACM, 43 (1996).

[44] M. Göös, S. Lovett, R. Meka, T. Watson, and D. Zuckerman, *Rectangles are nonnegative juntas*, in Proceedings of the 47th Symposium on Theory of Computing (STOC), 2015.

[45] D. Grigoryev, M. Karpinski, and M. Singer, *Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields*, SIAM Journal on Computing, 19 (1990).

[46] M. Hellman, *A cryptanalytic time-memory trade-off*, IEEE Transactions on Information Theory, 26 (1980).

[47] R. Impagliazzo and R. Paturi, *On the complexity of k-sat*, J. Comput. Syst. Sci., 62 (2001).

[48] G. Ivanyos, M. Karpinski, M. Santha, N. Saxena, and I. E. Shparlinski, *Polynomial interpolation and identity testing from high powers over finite fields*, Algorithmica, 80 (2018).

[49] R. Jain and H. Klauck, *The partition bound for classical communication complexity and query complexity*, in Proceedings of the 25th CCC, 2010.

[50] R. Jain, H. Klauck, and M. Santha, *Optimal direct sum results for deterministic and randomized decision tree complexity*, Information Processing Letters, 110 (2010).

[51] T. S. Jayram, R. Kumar, and D. Sivakumar, *Two applications of information complexity*, in Proceedings of the 35th Symposium on Theory of Computing (STOC), 2003.

[52] B. Kalyanasundaram and G. Schintger, *The Probabilistic Communication Complexity of Set Intersection*, SIAM Journal of Discrete Mathematics, 5 (1992).

[53] M. Karchmer, E. Kushilevitz, and N. Nisan, *Fractional covers and communication complexity*, SIAM Journal on Discrete Mathematics, 8 (1995).

[54] M. Karchmer, R. Raz, and A. Wigderson, *Super-logarithmic depth lower bounds via the direct sum in communication complexity*, Computational Complexity, 5 (1995).

[55] M. Karchmer and A. Wigderson, *Monotone circuits for connectivity require super-logarithmic depth*, SIAM Journal on Discrete Mathematics, 3 (1990).

[56] J. Kastner, A. Koch, S. Walzer, D. Miyahara, Y.-i. Hayashi, T. Mizuki, and H. Sone, *The minimum number of cards in practical card-based protocols*, in Advances in Cryptology – ASIACRYPT 2017, 2017.

[57] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2007.

[58] K. S. Kedlaya and C. Umans, *Fast modular composition in any characteristic*, in 2008 49th Annual IEEE Symposium on Foundations of Computer Science, 2008.

[59] A. Koch, *The landscape of optimal card-based protocols*. Cryptology ePrint Archive, Report 2018/951, 2018. urlhttps://eprint.iacr.org/2018/951.

[60] A. Koch and S. Walzer, *Private function evaluation with cards*. Cryptology ePrint Archive, Report 2018/1113, 2018. `https://eprint.iacr.org/2018/1113`.

[61] ——, *Foundations for actively secure card-based cryptography*, in 10th International Conference on Fun with Algorithms, FUN 2021, May 30 to June 1, 2021, Favignana Island, Sicily, Italy, 2021.

[62] A. Koch, S. Walzer, and K. Härtel, *Card-based cryptographic protocols using a minimal number of cards*, in Advances in Cryptology – ASIACRYPT 2015, 2015.

[63] G. Kol, S. Moran, A. Shpilka, and A. Yehudayoff, *Direct sum fails for zero-error average communication*, in Proceedings of the 5th Innovations in Theoretical Computer Science (ITCS), 2014.

[64] Y. Kun-Ko and O. Weinstein, *An adaptive step toward the multiphase conjecture*, in 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020, 2020.

[65] E. Kushilevitz and N. Nisan, *Communication Complexity*, Cambridge University Press, 1996.

[66] K. G. Larsen, *The cell probe complexity of dynamic range counting*, in Proceedings of the 44th STOC, 2012.

[67] ——, *Higher cell probe lower bounds for evaluating polynomials*, in Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science, 2012.

[68] K. G. Larsen, O. Weinstein, and H. Yu, *Crossing the logarithmic barrier for dynamic boolean data structure lower bounds*, in Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, 2018.

[69] T. Lee, A. Shraibman, and R. Špalek, *A direct product theorem for discrepancy*, in Proceedings of the 23rd Conference on Computational Complexity (CCC), 2008.

[70] Z. Li and B. Li, *Network coding: The case of multiple unicast sessions*, in Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing, 2004.

[71] P. B. Miltersen, *The bit probe complexity measure revisited*, in STACS 93, 1993.

[72] ——, *On the cell probe complexity of polynomial evaluation*, Theoretical Computer Science, 143 (1995).

[73] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson, *On data structures and asymmetric communication complexity*, in Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 1995.

[74] T. Mizuki, *Card-based protocols for securely computing the conjunction of multiple variables*, Theoretical Computer Science, 622 (2016).

[75] T. Mizuki, M. Kumamoto, and H. Sone, *The five-card trick can be done with four cards*, in Advances in Cryptology – ASIACRYPT 2012, 2012.

[76] T. Mizuki and H. Shizuya, *A formalization of card-based cryptographic protocols via abstract machine*, International Journal of Information Security volume, 13 (2014).

[77] ——, *Practical card-based cryptography*, in Fun with Algorithms, 2014.

[78] T. Mizuki and H. Sone, *Six-card secure and and four-card secure xor*, in Frontiers in Algorithmics, 2009.

[79] V. Niemi and A. Renvall, *Secure multiparty computations without computers*, Theoretical Computer Science, 191 (1998).

[80] N. Nisan, S. Rudich, and M. Saks, *Products and help bits in decision trees*, SIAM Journal on Computing, 28 (1999).

[81] T. Nishida, Y.-i. Hayashi, T. Mizuki, and H. Sone, *Card-based protocols for any boolean function*, in Theory and Applications of Models of Computation, 2015.

[82] A. Nishimura, Y.-i. Hayashi, T. Mizuki, and H. Sone, *An implementation of non-uniform shuffle for secure multi-party computation*, in Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography, 2016.

[83] R. Panigrahy, K. Talwar, and U. Wieder, *Lower bounds on near neighbor search via metric expansion*, in 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, 2010.

[84] M. Pătraşcu and E. D. Demaine, *Tight bounds for the partial-sums problem*, in Proceedings of the 15th SODA, 2004.

[85] ——, *Logarithmic lower bounds in the cell-probe model*, SIAM Journal on Computing, 35 (2006).

[86] N. Pippenger and M. J. Fischer, *Relations among complexity measures*, Journal of the ACM, 26 (1979).

[87] J. M. Pollard, *The fast fourier transform in a finite field*, Mathematics of computation, 25 (1971).

[88] M. Pătraşcu, *Towards Polynomial Lower Bounds for Dynamic Problems*, in Proceedings of the 42nd STOC, 2010.

[89] ——, *Unifying the landscape of cell-probe lower bounds*, SIAM J. Comput., 40 (2011).

[90] R. Raz and A. Wigderson, *Probabilistic communication complexity of boolean relations*, in Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS), 1989.

[91] A. A. Razborov, *On the distributional complexity of disjointness*, Theoretical Computer Science, 106 (1992).

[92] A. L. Selman, *P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP*, in Proceedings of the 6th International Colloquium on Automata, Languages and Programming (ICALP), 1979.

[93] J.-P. Serre, *A course in arithmetic*, Springer Science & Business Media, 2012.

[94] R. Shaltiel, *Towards proving strong direct product theorems*, in Proceedings of the 16th Conference on Computational Complexity (CCC), 2001.

[95] A. Shamir, *How to share a secret*, Commununications of the ACM, 22 (1979).

[96] A. A. Sherstov, *The communication complexity of gap hamming distance*, Theory of Computing, 8 (2012).

[97] A. Smoktunowicz, I. Wróbel, and P. Kosowski, *A new efficient algorithm for polynomial interpolation*, Computing, 79 (2007).

[98] A. Stiglic, *Computations with a deck of cards*, Theoretical Computer Science, 259 (2001).

[99] D. Unruh, *Random oracles and auxiliary input*, in Advances in Cryptology - CRYPTO 2007, 2007.

[100] L. G. Valiant, *Graph-theoretic arguments in low-level complexity*, in Mathematical Foundations of Computer Science 1977, 1977.

[101] L. G. Valiant, *Exponential lower bounds for restricted monotone circuits*, in Proceedings of the fifteenth annual ACM symposium on Theory of computing, 1983.

[102] E. Viola, *On the power of small-depth computation*, Foundations and Trends in Theoretical Computer Science, 5 (2009).

[103] ——, *The communication complexity of addition*, Combinatorica, 35 (2015).

[104] ——, *Lower bounds for data structures with space close to maximum imply circuit lower bounds*, Theory of Computing, 15 (2019).

[105] A. C. Yao, *Some complexity questions related to distributive computing (preliminary report)*, in Proceedings of the 11h STOC, 1979.

[106] A. C.-C. Yao, *Coherent functions and program checkers*, in Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, 1990.

List of Publications

[1] Václav Blažej, Pavel Dvořák, and Michal Opler. Bears with hats and independence polynomials, 2021.

[2] Pavel Dvořák, Michal Koucký, Karel Král, and Veronika Slívová. Data structures lower bounds and popular conjectures, 2021.

[3] Pavel Dvořák and Michal Koucký. Barrington plays cards: The complexity of card-based protocols. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, 2021.

[4] Pavel Dvořák and Tomáš Valla. Automorphisms of the cube n^d. *Discrete Mathematics*, 344, 2021.

[5] Pavel Dvořák and Bruno Loff. Lower bounds for semi-adaptive data structures via corruption. In *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, December 14-18, 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference)*, 2020.

[6] Pavel Dvořák, Andreas Emil Feldmann, Ashutosh Rai, and Paweł Rzążewski. Parameterized inapproximability of independent set in h-free graphs. In *Graph-Theoretic Concepts in Computer Science - 46th International Workshop, WG 2020, Leeds, UK, June 24-26, 2020, Revised Selected Papers*, 2020.

[7] Václav Blažej, Pavel Dvořák, and Tomáš Valla. On induced online ramsey number of paths, cycles, and trees. In *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, 2019.

[8] Pavel Dvořák and Dušan Knop. Parameterized complexity of length-bounded cuts and multicuts. *Algorithmica*, 80, 2018.

[9] Pavel Dvořák, Dušan Knop, and Tomáš Toufar. Target set selection in dense graph classes. In *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, 2018.

[10] Pavel Dvořák, Andreas Emil Feldmann, Dušan Knop, Tomáš Masarík, Tomaš Toufar, and Pavel Veselý. Parameterized approximation schemes for steiner trees with small number of steiner vertices. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, 2018.

[11] Pavel Dvořák, Eduard Eiben, Robert Ganian, Dušan Knop, and Sebastian Ordyniak. Solving integer linear programs with a small number of global variables and constraints. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 2017.

[12] Arkadev Chattopadhyay, Pavel Dvořák, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Lower bounds for elimination via weak regularity. In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, 2017.

[13] Pavel Dvořák and Tomáš Valla. On the computational complexity and strategies of online ramsey theory. *Electron. Notes Discret. Math.*, 49, 2015.