Space-Bounded Communication Complexity

by

Hao Song

Submitted to the Department of Computer Science and Technology
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at

TSINGHUA UNIVERSITY

July 2014

Author .....
Department of Computer Science and Technology
April 19, 2014

Certified by .....
Periklis A. Papakonstantinou
Assistant Professor
Thesis Supervisor

Accepted by .....
Shiqiang Yang
Chairman, Department Committee on Graduate Theses

Space-Bounded Communication Complexity

by

Hao Song

Submitted to the Department of Computer Science and Technology
on April 19, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In this work, we introduce clean, purely information theoretic space-bounded communication complexity models. In our new general space-bounded communication model, unlike in classical communication complexity models, the players only have limited space to compress previous communication history. The players remain all-powerful when performing local computation, the same as in classical models. We also introduce restricted variants of this general model, in particular the limited-memory communication model and the memoryless communication model. In these models the communication is one-way and the players are constrained in the way they utilize their limited memory. We show several basic properties about these new models, in particular that the limited-memory communication model can simulate the general space-bounded two-way communication model with moderate overhead.

We introduce a new combinatorial concept called rectangle overlay, which naturally generalizes the rectangle partition and rectangle cover concepts in classical communication complexity. This concept fully characterizes our memoryless communication model.

Our new space-bounded communication models provide several new characterizations and new lower bound techniques for the study of the communication complexity polynomial hierarchy [7]. This communication analog of the Turing machine polynomial hierarchy has recently attracted a lot of attention because of its newly found technical connection with its Turing machine counterpart [2, 20]. In particular, our rectangle overlay concept fully characterizes $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$, the communication analog of the oracle Turing machine complexity class P^{NP}. It is the first combinatorial characterization of $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ and it properly conceptualizes and strengthens previously known separation results concerning $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$. We also provide the first characterization of the $\mathsf{PSPACE}^{\mathsf{cc}}$ complexity class, the communication analog of PSPACE, in terms of a natural space notion.

We show equivalences and separations to other models (e.g. the garden-hose model [17], bounded-width branching programs, etc.), unify and extend techniques that appearing in [30] and [10] regarding previously proposed space-bounded communication models, all using the same models we introduce.

Thesis Supervisor: Periklis A. Papakonstantinou
Title: Assistant Professor

Acknowledgments

I would like to thank my adviser Professor Periklis Papakonstantinou, for all the guidance and help he has given me all these years, and for always having a very high standard regarding every aspect of my research. This work would have never been done without his supervision. I would also like to thank Professor Andrew C. Yao, Dean of Institute for Interdisciplinary Information Sciences, Tsinghua University, for creating a research environment I very much enjoyed.

I would like to express my appreciation to all the colleagues and classmates I have worked with: Joshua Brody, Shiteng Chen, Dominik Scheder, and Xiaoming Sun. It is a great experience working with these talented researchers. In particular, I would like to thank Dominik Scheder for our involved collaboration.

I would like to thank Harry Buhrman, Amit Chakrabarti, Peter Bro Miltersen, Janos Simon, John P. Steinberger, Iddo Tzameret, and Ronald de Wolf for their involved feedback, comments, and thorough suggestions on this thesis. I would also like to thank Paul Beame, Andrej Bogdanov, Mark Braverman, Kristoffer A. Hansen, Iordanis Kerenidis, Frederic Magniez, Miklos Santha, Florian Speelman, and Chengu Wang for enlightening discussions and very helpful comments regarding my work.

This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61361136003, 61350110536.

Contents

List of Figures

List of Tables

Notations and Symbols

Notation for Standard Concepts

$\log n$: the base 2 logarithm of n.
$P(S)$: the power set of set S.
$\mathbb{Z}^+$: the set of all positive integers.
$\mathbb{R}$: the set of all real numbers.
$GF(p^k)$: the Galois field of size p^k, for prime number p and positive integer k.
$H(p)$: the binary entropy function, for $0 < p < 1$, $H(p) \stackrel{\text{def}}{=} p \cdot \log\left(\frac{1}{p}\right) + (1-p) \cdot \log\left(\frac{1}{1-p}\right)$, and $H(0) = H(1) = 0$ by continuity.

Complexity Measures

Complexity Classes

[1] To be more precise, this is bounded-error public-coin randomized communication complexity in literature.

[2] To be more precise, this is bounded-error public-coin memoryless complexity.

P^{cc}: $\mathsf{polylog}(n)$ deterministic communication complexity ... Definition 1.3.1 page 23
$\mathsf{NP}^{\mathsf{cc}}$: $\mathsf{polylog}(n)$ nondeterministic communication complexity Definition 2.1 page 39
$\mathsf{coNP}^{\mathsf{cc}}$: $\mathsf{polylog}(n)$ co-nondeterministic communication complexity ... Definition 2.1 page 39
Π_k^{cc}, Σ_k^{cc}: k-th level of the communication polynomial hierarchy . Definition 2.1 page 39
$\mathsf{PH}^{\mathsf{cc}}$: constant level of the communication polynomial hierarchy Definition 2.1 page 39
$\mathsf{PSPACE}^{\mathsf{cc}}$: $\mathsf{polylog}(n)$ level of the communication polynomial hierarchy .. Definition 2.1 page 39
$\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$: protocol with $\mathsf{polylog}(n)$ cost and oracle access to $\mathsf{NP}^{\mathsf{cc}}$ Definition 2.3 page 40
AC^k, NC^k: bounded-depth circuit classes .....Definition 2.7 page 42
$\mathsf{SPACE}_{\mathsf{LESS}}[s]$: memoryless protocol with message length s ... Definition 4.1 page 61
$\mathsf{SPACE}_{\mathsf{LTD}}[s, w]$: limited-memory protocol with message length s and w memory states Definition 5.1 page 78

Computation/Communication Problems

GT: Greater-Than .....Problem 1.1 page 21
IP_{p^k}: Inner-Product on $GF(p^k)$ .....Problem 1.2 page 25
IP: abbreviation for IP_2 .....Problem 1.2 page 25
$\mathrm{LNE}_{k,l}$: List-Non-Equality .....Problem 1.3 page 25
GHD: Gap-Hamming-Distance .....Problem 1.5 page 26
AppMaj: Approximate-Majority .....Problem 4.6 page 71
EQ: Equality .....Problem 1.6 page 27
NEQ: Non-Equality .....Problem 2.4 page 41
ALL-EQ: All-Subset-Equality .....Problem 1.8 page 31
EQ-WITH-DESIGN$_k$: Equality-with-Combinatorial-Design .....Problem 1.9 page 32

Miscellaneous

$\{f_n^{cc}\}_{n=1}^{\infty}$, $\{f_n\}_{n=1}^{\infty}$: function family, one function per input length Section 1.3.1 page 25
$HD(x, y)$: Hamming distance ..... Definition 1.4 page 25
$HS(x, y)$: complement of Hamming distance ..... Section 3.5.2 page 55
$|x|_1$: number of 1s in a bit string ..... Section 3.5.2 page 55
$NB(A, d)$: neighbor set in Hamming cube .....Section 4.5.3 page 72

Chapter 1

Introduction

One of the central objectives of theoretical computer science is to unveil the intrinsic hardness of computational problems. That is, to study lower bounds on the amount of certain kind of resources (e.g. computation time or memory space) required to successfully solve the problem.

In 1979 Andrew Yao [47] introduced a model called *communication complexity*. This model turned out to be a very powerful tool in proving lower bounds. It has been successfully applied to various areas of computer science, including circuit complexity (e.g. [23, 37]), streaming computation (e.g. [5, 22]), and property testing (e.g. [11]), to name a few. Typically, a lower bound is obtained through the use of communication complexity in the following way: we divide a particular computation process into several phases, or we divide a system or device into several parts (e.g. several computers connected by a network in a distributed system) and then we can abstract away the details of the computation performed inside each phase or inside each part. We think of individual phases or parts to be all powerful players who can do any information processing with no cost at all, instead, we try to lower bound the amount of information exchange that must happen among the players for solving the problem. Interestingly, strong lower bounds can be proved by solely focusing on this kind of information exchange among the identified parts of the process or system.

In this work we seek to strengthen this lower bound tool by taking into account another resource constraint that is common in real-world computation: *memory space*. We consider the information-theoretic approach of classical communication complexity to be its most important aspect which makes this tool so widely applicable. Therefore, we introduce a set of space-bounded communication complexity models which are still purely information theoretic, in the sense that the details of the computation required for information processing are still abstracted away (there are also previous similar attempts as e.g. by Buhrman et al. [17]). In this thesis, using these models, we unify and improve existing techniques, and provide new paradigms, new characterizations and new combinatorial tools.

Besides the conceptual contribution in introducing these models, the most interesting application of our communication models is that they provide new tools for the study of some important communication complexity classes, such as the communication complexity polynomial hierarchy by Babai, Frankl, and Simon [7]. Recent results

by Aaronson and Wigderson [2] and by Impagliazzo, Kabanets, and Kolokolova [20] connected the study of these communication complexity classes with some prominent open problems in complexity theory concerning Turing machine complexity classes NL, P and NP.

Our new communication models also provide a unified framework for understanding some previous space-bounded communication models that were proposed more than 20 years ago by Lam, Tiwari and Tompa [30], and by Beame, Tompa and Yan [10], and some contemporary works by Buhrman, Fehr, Schaffner, and Speelman [17].

We also perform a comprehensive study about these new communication models themselves and obtain what is commonly understood as elementary results for new models of computation.

Figure 1-1 depicts the relationship between this work and previous works, especially the rich set of connections among the models and concepts we introduce and existing models and concepts.

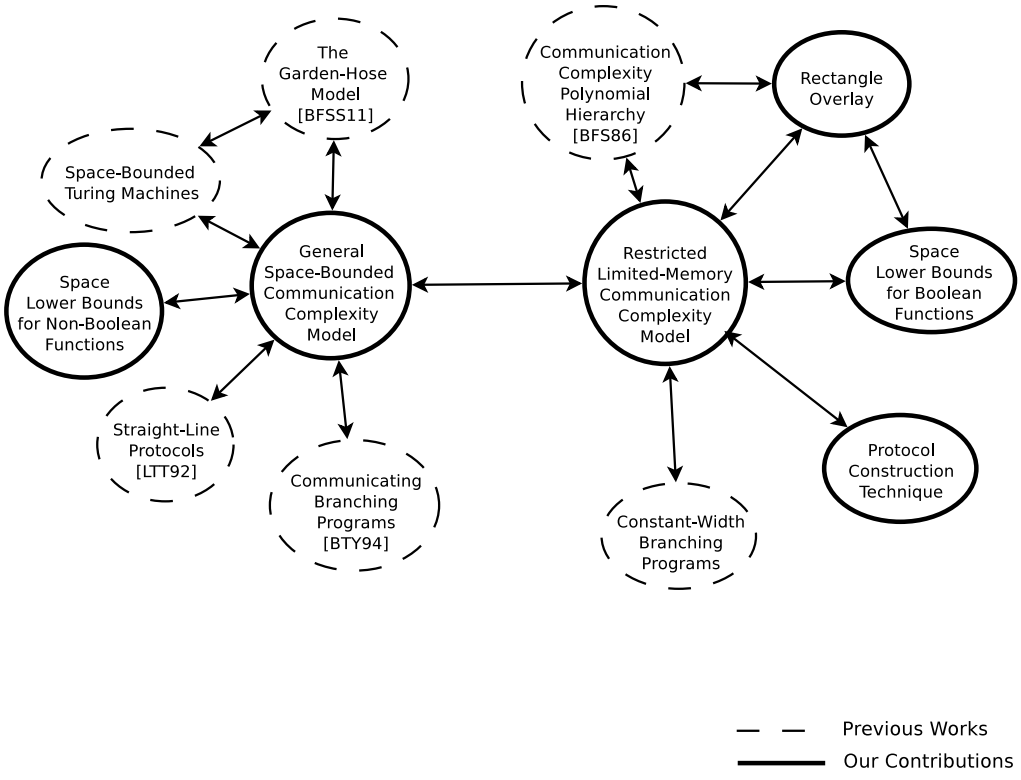


Figure 1-1: The Unified View of Our Treatment

Technical parts of this thesis have appeared in a published form in [16] and [34].

1.1 Philosophy: the "Information-Theoretic" Approach

To understand the rationales behind our space-bounded communication models we first elaborate some more about what we understand as the information theoretic

approach.

The term "information-theoretic" is used in this work in a rather liberal way. When we say a lower bound proof technique is information-theoretic, we mean that the main argument in it is about some type of "information cost" that we must pay to solve the problem. For example, a typical proof of the $\Omega(n \cdot \log n)$ lower bound for comparison-based sorting algorithms on n input elements argues about the fact that we need to gather at least $O(n \cdot \log n)$ bits of information from the input in order to distinguish between the $n!$ different orderings of n input elements. This type of argument, which pervades the nature of lower bounds in computational models, is in contrast to arguments using, for example diagonalization (e.g. space/time hierarchy theorems). We do not want to discuss further, neither it seems appropriate, to get into the epistemological differences between types of lower bounds.

As a tool for proving lower bounds, the standard communication model by definition focuses on information. In the basic communication complexity setting, the two players Alice and Bob each receive an n-bit input. In each step one player sends a bit string to the other player. The player who receives that bit string can compute an arbitrary function (at no cost) based on her input and all the communication bits received so far, and decide to either send a bit string back or give a final answer. In such a communication protocol the complexity measure, communication cost, is the maximum total number of communication bits in the worst case (over all possible input values). The *communication complexity* of a function is intuitively defined to be the communication cost of the optimal protocol that correctly computes the function on every possible input value. Lower bounds proved using classical communication complexity are based on the amount of information exchange that must happen before the players can be sure about the correct answer of the problem. The players are assumed to be all-powerful.

The ability of the players to process an arbitrary amount of local information at no cost is an important reason why the classical communication complexity model is so widely applicable as a lower bound tool. Because the details of local information processing are abstracted away, the players can be anything, e.g. the whole or a part of a boolean circuit, the whole or a part of a Turing machine, etc. Therefore many problems in other models of computation can be mapped into a communication problem.

Before this work there were several attempts to add space bounds to the classical communication complexity model, the notable ones include Lam, Tiwari and Tompa's work on straight-line protocols [30], Beame, Tompa and Yan's work on communicating branching programs [10], and Klauck's work on communicating circuits [25]. In all these works the authors realized the players in the communication models using some standard "machines". These machines specify the details of information processing. The "space" in these communication models includes the space required to store intermediate computation results in local information processing. Although it is conceivable to obtain stronger lower bounds based on stronger local information processing constraints, it becomes harder to map problems in other models of computation to these communication models, like it is commonly done for classical communication complexity.

In this work, we introduce space-bounded communication models that are purely information theoretic like the classical communication complexity model. The details of local information processing are still abstracted away. The space concept in our models is not about space requirement for performing local information processing, rather it is for storing previous communication bits in a compressed form.

An interesting fact is that the majority of the proof techniques the authors used in two of the aforementioned papers [30, 10] are information theoretic. These arguments do not rely on the specific details about local information processing, and can actually be transferred to the models we introduce in this work, which is a more general information theoretic setting (see Section 3.4 on page 51). We observe that this is actually a very common phenomenon. The success of the classical communication complexity shows that the best known lower bounds can oftentimes be obtained by information theoretic means, while ignoring the details of information processing in standard "machines" like Turing machines.

Furthermore, our space-bounded communication models have a lot of interesting connections to existing computational models and complexity classes, as we will discuss later on in Section 1.3.1 (page 21).

1.2 Conceptual Contributions: Space-Bounded Communication Models

1.2.1 The General Model

In our *general* space-bounded communication complexity model, the players Alice and Bob have limited memory space to remember their prior communication history. Consider a protocol in this space-bounded model that computes a *boolean-output* function. Again each of them gets part of the input to the function. In each step they exchange two bits, one from Alice to Bob, and one from Bob to Alice. Upon receiving the bit from the other player each of them computes an arbitrary *transition function* (with no associated cost) which takes three pieces of information as input: her local input, the content of her local memory, and the communication bit received. Based on the output of the *transition function*, each of them may decide to either give an answer and halt the protocol, or continue by updating the content of her local memory and sending another bit to the other player.

When defining a protocol to compute a *non-boolean* function (which may have more than one bit of output), we allow the players to answer a subset of the output bits in each step (not necessarily all at the end). The function is correctly computed if the union of the answers given by both players gives the correct values for all the output bits of the function.

Concerning how many output bits can be answered in each step we consider two possible variants.

- the *fluent* variant, in which each player can choose to answer as many output bits as she wants (or nothing at all) in each step;

- the *stuttering* variant, in which each player can only answer at most one output bit in each step.

In the classical communication complexity model, since the players have no space constraints at all, one player, e.g. Alice, can send her entire local input to the other player and rely on the other player to come up with an answer. That is to say, the length parameter $n + 1$ is always a trivial upper bound on the communication complexity of any function with two n-bit inputs.

In previous works on straight-line protocols [30] and communicating branching programs [10], the authors all showed that the trivial upper bound in the classical communication complexity no longer holds when a space bound is added to the model. They gave communication-space tradeoff lower bounds to some non-boolean functions (such as matrix-vector multiplication). By transferring their techniques to the stuttering variant of our model, we show similar results in our model as well (see Section 1.3.1 page 30 for more).

As for the fluent variant of our model, we also obtain non-trivial space lower bound results for some explicitly defined functions (see Theorem 1.18 on page 32 and Theorem 1.19 on page 32). That is, such functions are not computable at all within certain space-bound (regardless of the amount of communication). In this work we mainly focus on "computability" problems, which studies whether a function is computable at all within certain space constraint. Note that classical communication complexity deals with communication resources.

Proving nontrivial space lower bounds for explicitly defined *non-boolean* functions in our general space-bounded communication model is doable. For boolean functions we can also show that almost every boolean function will require close to linear memory space to compute (see Theorem 1.20 page 32). However, proving interesting (e.g. super-logarithmic) space lower bounds for *explicitly defined boolean* functions in our general space-bounded model goes beyond current techniques (see Section 3.5.1 page 54).

1.2.2 The Restricted Variants: Main Contribution

The more restricted variants of the space-bounded communication model was initially introduced to address the difficulties of proving nontrivial space lower bounds in the general model for explicitly defined boolean functions. These restricted variants turned out to be very powerful, being able to simulate the general model with moderate overhead (see Theorem 1.14 on page 31). At the same time they provide new tools and new characterizations for the study of the communication complexity analog of the Turing machine polynomial-time hierarchy (see Section 1.3.1 page 21). These results are conceptually more significant than the results we get so far in the general model.

In all the restricted variants, the communication is one-way, Bob can receive information from Alice, but he cannot send any information back to Alice. In addition, for some *special part of his memory*, Bob can only access it in restricted ways. For example, some part of Bob's memory (we call it "oblivious memory") can only be

used to compress the information received from Alice. When updating this part of his memory, Bob cannot mix-in any information about his own local input; some other part of Bob's memory (we call it "write-once memory") can only be written into once, each bit in this part is initialized to the special "blank" state, and after writing either 0 or 1 into it, Bob cannot change the value of that bit again. A combination of these restricted types of memory can give rise to a variety of communication models.

The two most important such models are the *memoryless* communication model and the *limited-memory* model. In the *memoryless* model, a protocol proceeds in rounds, and like the name suggests, Bob has no permanent memory, he receives a bounded-length message from Alice in each round. Upon receiving the message, Bob either decides to give a final answer (based on available information, that is, his local input and the received message) and halt, or completely forget what has happened so far and continue. In the next round, Bob will receive another brand new message from Alice. The *memoryless complexity* of a function is the minimum message length requirement in the memoryless model to correctly compute the function. Note here we mainly care about the "space" (message length) requirement, without considering the total amount of communication. We can also define the memoryless model as a restricted variant of the general model with one-way communication in which all Bob's memory is *oblivious* (see Section 4.1.1 page 62 for more). We prefer the above definition because it better reflects the concept formalized by the model.

In the more general *limited-memory* model, Bob does have some constant size permanent memory. The communication is still one-way, from Alice to Bob. In each round, Bob receives a bounded-length message from Alice, then makes a decision based on available information whether to give a final answer or continue after forgetting about Alice's message. Unlike the memoryless model, in the limited-memory model, if Bob decides to continue, he can update the content of his constant size permanent memory and carry that memory to the next round. A *limited-memory* protocol has two complexity measures: the message length, and the size of Bob's permanent memory. We are usually interested in the minimum message length requirement to correctly compute a function when the size of Bob's memory is fixed, regardless the total amount of communication. This model can also be defined as a restricted variant of the general model, in which communication is one-way, and the vast majority of Bob's memory is *oblivious* (see Section 5.1.1 on page 78).

As we will see later (cf. Theorem 1.6 on page 28 and Theorem 1.7 on page 28), with each additional memory bit, the limited-memory model gains much more power. With 3 memory bits (to be more precise, just 5 memory *states*), the limited-memory model can simulate the two-way general model with moderate overhead (see Theorem 1.14 on page 31). Therefore in order to have a finer analysis of the power of Bob's permanent memory, in this work we measure Bob's permanent memory in the limited-memory model in terms of the number of memory states instead of the number of memory bits (s memory bits is obviously equivalent to 2^s memory states). The memoryless model is a special case of the limited-memory model with 1 memory state.

Note that in the *limited-memory* and *memoryless* models, the protocols only have one-way communication by design. The reason behind this is discussed in Section 4.1.1 on page 62.

1.3 Technical Contributions

Now, we list the technical contributions of this work in order of significance. We present the important applications of our new models first. Relevant properties of the models are presented along the way. Answers to certain elementary questions about new models of computation (e.g. does more memory help) are presented after that.

First comes a new combinatorial concept we introduced, the *rectangle overlay*. This surprisingly simple combinatorial concept provides full characterization for both the memoryless complexity and the important class $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$, the communication complexity analog of the oracle Turing machine class P^{NP}. We then put the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ class into the bigger picture of the communication complexity polynomial hierarchy (first defined in [7], see Section 2.2 on page 39 for definition) and present more connections between our limited-memory communication model and the higher levels of this hierarchy. Some new lower bounds and separation results in the communication complexity polynomial hierarchy achieved through these newly found connections and characterizations are presented along the way.

Next, we explore more connections between our new space-bounded communication models and other related models like space-bounded Turing machines, the garden-hose model [17], straight-line protocols [29], communicating branching programs [9]. We show that we provide a unified information theoretic framework in which we are able to put each of these other models at its place.

After discussing these applications of our space-bounded communication models we present some results pertaining to the models themselves. We start presenting some properties of the models, and compare the power of the general two-way model with the more restricted variants. Then we present some complexity lower bound results and some nontrivial protocols.

1.3.1 Applications: New Tools and New Paradigms

Rectangle Overlay and $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$

In this work, we introduce a new combinatorial tool called *rectangle overlay* [1]. It is immensely useful in the study of both our memoryless model and the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ complexity class. In order to put this new tool into context, we need to first look at the combinatorial tools in classical communication complexity that precede it: communication matrix, rectangle partition, and rectangle cover.

Communication matrix is the central tool in communication complexity research. Table 1.1 shows the communication matrix of the Greater-Than function (denoted as GT) for input length $n = 2$.

Problem 1.1 (Greater-Than, GT)**.** *For two n-bit strings x and y,* $\mathrm{GT}(x, y)$ *outputs* 1 *if and only if $x > y$ when both are interpreted as binary encoded integers.*

[1] see Definition 4.3 on page 63 for formal definition

Each row in the matrix corresponds to one possible value for input x, and each column corresponds to one possible value for input y, the entry at the intersection of a particular row and a particular column is filled with the output of the function on the corresponding input pair (x, y).

$\mathrm{GT}(x, y)$	$y = 00$	$y = 01$	$y = 10$	$y = 11$
$x = 00$	0	0	0	0
$x = 01$	1	0	0	0
$x = 10$	1	1	0	0
$x = 11$	1	1	1	0

Table 1.1: Communication Matrix of GT

A *combinatorial rectangle* (or just *rectangle* for short) in a communication matrix is simply a sub-matrix of that communication matrix. In other words, a rectangle is the Cartesian product of a subset of rows and a subset of columns in the communication matrix. The process of executing a communication protocol is a process in which Alice and Bob narrow down the set of possible input pairs (x, y) that are consistent with all the communication bits so far. It is not hard to prove that in every step of the protocol, this set of possible input pairs always forms a rectangle in the communication matrix. Originally this rectangle contains of course all the entries in the communication matrix. And with each bit of communication between Alice and Bob, the rectangle shrinks. Whenever the rectangle shrinks to a *monochromatic* one (that is, it contains either only 0 entries or only 1 entries in the communication matrix), Alice and Bob may safely give a final answer.

Prior to this work, there were two combinatorial concepts related to rectangles that are central to the study of communication complexity: *rectangle partition* and *rectangle cover*. A *rectangle partition* of a communication matrix consists of a set of monochromatic rectangles. These rectangles should not intersect with each other, and collectively they should contain every entry in the matrix. That is, every entry in the matrix is contained in exactly one rectangle from this set. A *rectangle cover* also consists of a set of monochromatic rectangles. This set of rectangles should collectively contain either all 1 entries (a 1-cover) or all 0 entries (a 0-cover) in the communication matrix, and the rectangles are allowed to intersect with each other. In a sense, the rectangle cover concept is a generalization of the rectangle partition concept. An optimal rectangle partition of GT's communication matrix is shown in Figure 1-2. An optimal rectangle 0-cover and an optimal rectangle 1-cover of the same communication matrix are shown in Figure 1-3.

The *rectangle overlay* concept is a further generalization of the rectangle cover concept. A rectangle overlay is a sequence of rectangles, each with an associated "color" of either 0 or 1. Unlike a partition or a cover, the rectangles in an overlay come in order. These rectangles may intersect with each other, and collectively they should contain all matrix entries. The output value of an entry must be equal to the color of the first rectangle in the sequence that contains it. In other words, we "overlay" the rectangles one-by-one (see Definition 4.3 on page 63 for formal definition). An

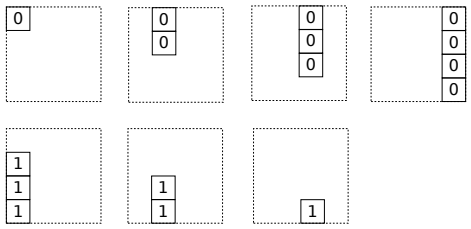
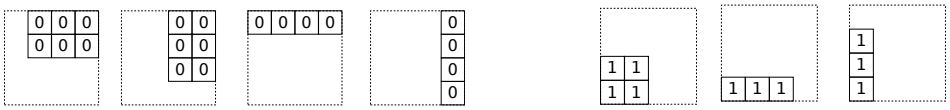


Figure 1-2: Rectangle Partition of GT's communication matrix



(a) 0-Cover (b) 1-Cover

Figure 1-3: Rectangle 0-Cover and 1-Cover of GT's communication matrix

optimal rectangle overlay of GT's communication matrix is shown in Figure 1-4 (note the rectangles are listed from left to right).

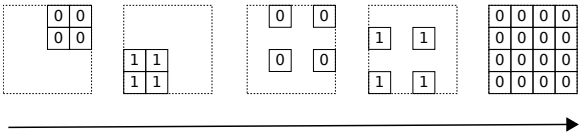


Figure 1-4: Rectangle Overlay of GT's communication matrix

We care about these combinatorial concepts because they are closely related to the communication complexity in various communication models. Suppose the minimum partition, the minimum 1-cover, the minimum 0-cover and the minimum overlay of the communication matrix of a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ each have $RP(f)$, $RC^1(f)$, $RC^0(f)$ and $RO(f)$ rectangles respectively, then it is well-known results that the deterministic communication complexity of f is between $\lceil \log RP(f) \rceil$ and $O((\log RP(f))^2)$, the nondeterministic communication complexity of f is $\lceil \log RC^1(f) \rceil$, and the co-nondeterministic communication complexity of f is $\lceil \log RC^0(f) \rceil$ (see Section 2.1 on page 35 for definitions of the communication complexity measures). In this work, we show that the memoryless complexity of f is fully characterized by $RO(f)$ (up to a factor of 2):

Theorem 1.1 (restated and proved as Theorem 4.5 on page 65)**.** *The memoryless complexity of a function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *is between* $\frac{\lceil \log RO(f) \rceil - 1}{2}$ *and* $\lceil \log RO(f) \rceil$.

Recall that *memoryless complexity* is the minimum message length requirement to solve a problem in our memoryless communication model, as defined above in Section 1.2.2. This gives a beautiful connection between four communication complexity measures and four combinatorial concepts.

In a 1986 paper, Babai, Frankl and Simon [7] defined a number of communication complexity classes as analogs to the well-known Turing machine classes. These include

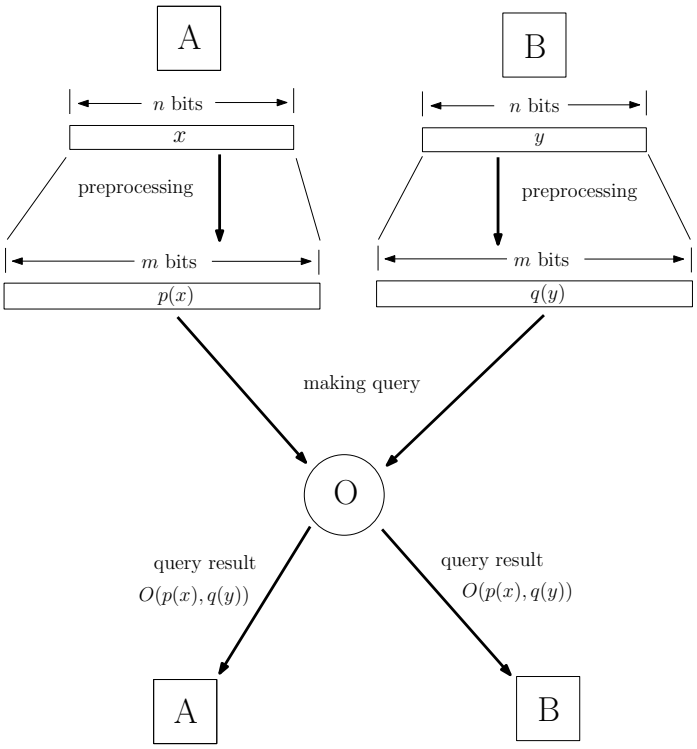


Figure 1-5: Oracle Query in Communication Protocols

the communication complexity analog of the Turing machine polynomial hierarchy (see Section 2.2 on page 39 for definitions).

At the bottom level of this communication polynomial hierarchy, we have the complexity class P^{cc}, which contains all communication problems with $\mathsf{polylog}(n)$ *deterministic* communication complexity; the complexity class $\mathsf{NP}^{\mathsf{cc}}$, which contains all communication problems with $\mathsf{polylog}(n)$ *nondeterministic* communication complexity; and the complexity class $\mathsf{coNP}^{\mathsf{cc}}$, which contains all communication problems with $\mathsf{polylog}(n)$ *co-nondeterministic* communication complexity. (See Section 2.1 on page 35 for definitions of the various communication complexity measures.)

In the hierarchy, the complexity class that is right above $\mathsf{NP}^{\mathsf{cc}}$ and $\mathsf{coNP}^{\mathsf{cc}}$, is $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ (Definition 2.3 on page 40), the communication analog of the oracle Turing machine complexity class P^{NP} (polynomial-time computable, with access to an NP oracle). An oracle query in the communication setting is depicted in Figure 1-5. Here Alice and Bob first each does some arbitrary *local preprocessing* to their respective inputs x and y, and get $p(x)$ and $q(y)$. Then they submit their preprocessed inputs to an oracle O, and they both immediately get the answer of the oracle query $O(p(x), q(y))$. The cost of such an oracle query is $\log m$ (m being the output length of $p(x)$ and $q(y)$), as compared to the cost of one bit of communication, which is always 1. A protocol in the class $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ class has access to a particular oracle in the class $\mathsf{NP}^{\mathsf{cc}}$, and has total cost at most $\mathsf{polylog}(n)$.

In this work we show:

Theorem 1.2 (restated and proved as Theorem 4.6 on page 66)**.** $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ *contains*

exactly the set of functions $\{f_n^{cc}\}_{n=1}^{\infty}$ [2] *with* $\mathsf{polylog}(n)$ memoryless *complexity.*

This gives the first combinatorial characterization of the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ class in terms of combinatorial rectangles that we know of:

Corollary 1.3. $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ *contains exactly the set of functions* $\{f_n^{cc}\}_{n=1}^{\infty}$ *with rectangle overlays of size* $2^{\mathsf{polylog}(n)}$.

This complexity class $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$, which has been around since the early days of communication complexity and studied in several prior works [7, 18, 21], has such a simple combinatorial characterization that was overlooked for quite some time. Previously, in the communication complexity polynomial hierarchy, both the complexity classes immediately below $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ (that are, $\mathsf{NP}^{\mathsf{cc}}$ and $\mathsf{coNP}^{\mathsf{cc}}$) and the complexity classes above immediately $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ (that are, Σ_2^{cc} and Π_2^{cc}) have combinatorial characterizations in terms of iterated intersections and unions of monochromatic rectangles [7].

As a combinatorial tool, the rectangle overlay concept provides an intuitive lower bound technique in terms of maximum monochromatic rectangle size.

Theorem 1.4 (a simplified version, see Theorem 4.8 on page 68 for general version and proof)**.** *Suppose in the communication matrix of* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *the maximum monochromatic rectangle contains* m *entries (note the whole matrix contains* 2^{2n} *entries) then the length of any rectangle overlay of* f *must be at least* $2^{n/2-(\log m)/4-1/2}$.

This lower bound technique significantly simplifies and appropriately conceptualizes the technique presented by Impagliazzo and Williams in their 2010 paper [21].

Using this technique, we can prove lower bounds for the following boolean functions.

Problem 1.2 (Inner-Product, IP)**.** *For a prime number* p *and a positive integer* k*, the* IP_{p^k} *function computes the inner product of two* n*-dimensional vectors in* $GF(p^k)$*. For* $x = x_1x_2\ldots x_n$ *and* $y = y_1y_2\ldots y_n$, $\mathrm{IP}_{p^k}(x,y) = \sum_{i=1}^{n} x_iy_i$*, in which all multiplication and addition operations are taken in* $GF(p^k)$*. For the special case where* $p = 2$ *and* $k = 1$*, we abbreviate* IP_2 *to* IP.

Problem 1.3 (List-Non-Equality, LNE)**.** *The List-Non-Equality* $\mathrm{LNE}_{k,l}$ *function is defined on two* (lk)*-bit strings* $x = x_1^{(1)}x_2^{(1)}\ldots x_l^{(1)}x_1^{(2)}\ldots x_1^{(k)}\ldots x_l^{(k)}$ *and* $y = y_1^{(1)}y_2^{(1)}\ldots y_l^{(1)}y_1^{(2)}\ldots y_1^{(k)}\ldots y_l^{(k)}$. $\mathrm{LNE}_{k,l}(x,y) = 1$ *if and only if* $\forall i \in \{1,2,\ldots,k\}$ $x^{(i)} \neq y^{(i)}$

Definition 1.4 (Hamming distance)**.** *For two* n*-bit strings* $x = x_1x_2\ldots x_n$ *and* $y = y_1y_2\ldots y_n$*, the* Hamming distance *between* x *and* y*, denoted as* $HD(x,y)$*, is defined to be the cardinality of the set* $\{i \mid x_i \neq y_i\}$

[2] In this work, when we talk about asymptotics, we use the notation $\{f_n^{cc}\}_{n=1}^{\infty}$ ($\{f_n\}_{n=1}^{\infty}$) to denote a family of functions, one for each input length n, that is $f_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ ($f_n : \{0,1\}^n \to \{0,1\}$). For convenience, sometimes we also refer to such a family as just "one function" that takes input of arbitrary length.

Problem 1.5 (Gap-Hamming-Distance, GHD)**.** *The partial function Gap-Hamming-Distance* GHD *is defined in terms of the* Hamming distance *between two bit strings. For two n-bit strings $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_n$, if the Hamming distance between x and y is at least $2n/3$,* $\mathrm{GHD}(x, y)$ *outputs* 1*; if the Hamming distance is at most $n/3$, the function outputs* 0*; otherwise the output of the function is unspecified.*

Theorem 1.5 (restated and proved as Theorem 4.9 on page 69, Theorem 4.10 on page 69, and Theorem 4.11 on page 70)**.** *The List-Non-Equality function* $\mathrm{LNE}_{\sqrt{n},\sqrt{n}}$*, the Inner-Product over $GF(2)$ function* IP*, and (any full-function extension of) the Gap-Hamming-Distance function* GHD *have memoryless complexity $\Omega(\sqrt{n})$, $\Theta(n)$, and $\Theta(n)$ respectively. None of them is in the communication complexity class* $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$*.*

In their 2010 paper [21], Impagliazzo and Williams proved that $\mathrm{IP} \notin \mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$, and separated $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ from $\Sigma_2^{cc} \cap \Pi_2^{cc}$ using the $\mathrm{LNE}_{\sqrt{n},\sqrt{n}}$ function (Lam and Ruzzo proved that $\mathrm{LNE}_{\sqrt{n},\sqrt{n}} \in \Sigma_2^{cc} \cap \Pi_2^{cc}$ in 1992 [28]). Here we give a more intuitive proof for this result using the new combinatorial tool we introduce, rectangle overlay. In addition, we show a stronger separation between $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ and Σ_2^{cc}, and between $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ and Π_2^{cc}, by proving that there are two full function extensions of the Gap-Hamming-Distance function GHD, one in Σ_2^{cc}, another in Π_2^{cc} (see Theorem 4.13 on page 71).

Going Beyond $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ in the Communication Polynomial Hierarchy

As mentioned before, the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ class is one of the (infinite) levels of the communication polynomial hierarchy. Now we show further connections between our model and the higher levels of this hierarchy.

The communication polynomial hierarchy [7] (see Section 2.2 on page 39 for formal definitions), like its Turing machine counterpart, consists of a tower of levels of complexity classes. A complexity class at a certain level contains all the complexity classes at lower levels. At "level k" ($k = 1, 2, \dots$), we have two complexity classes: Σ_k^{cc} and Π_k^{cc}. The aforementioned $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ class is between the "first level" ($\Sigma_1^{cc} = \mathsf{NP}^{\mathsf{cc}}$ and $\Pi_1^{cc} = \mathsf{coNP}^{\mathsf{cc}}$) and the "second level" (Σ_2^{cc} and Π_2^{cc}) of this hierarchy. This hierarchy, like its Turing machine counterpart, is defined in terms of alternations of quantifiers $\forall$ and $\exists$. For example, the definition of Π_2^{cc} can be intuitively represented as a depth-3 "boolean formula" as depicted in Figure 1-6. The top gate of the formula is a $\wedge$-gate, corresponding to a $\forall$ quantifier, with fan-in $2^{\mathsf{polylog}(n)}$. Every middle layer gate is a $\vee$-gate, corresponding to an $\exists$ quantifier, with fan-in $2^{\mathsf{polylog}(n)}$. And every bottom layer gate is a $\wedge$-gate with fan-in 2, one of its two children is an "input" gate taking a boolean preprocessing function $p_\cdot(x)$ on x, the other is another "input" gate taking a boolean preprocessing function $q_\cdot(y)$ on y. We will more formally discuss this relationship between the communication polynomial hierarchy and the so-called circuit with local preprocessing model in Fact 2.6 on page 43.

The "level number" k can be defined to be a function of the input length n, and the "largest" meaningful value for k is $\mathsf{polylog}(n)$, which gives rise to the PSPACE^{cc}

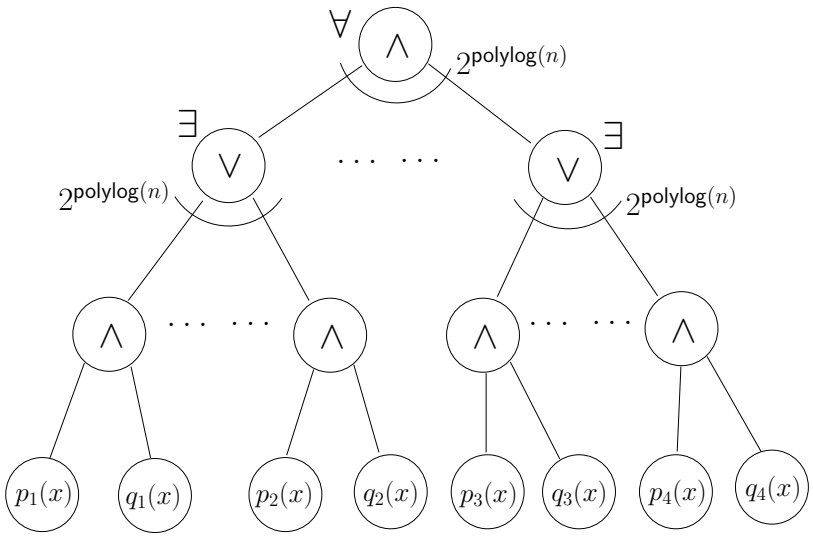


Figure 1-6: Circuit Representation of Π_2^{cc}

class, more precisely defined as $\mathsf{PSPACE}^{\mathsf{cc}} \stackrel{\text{def}}{=} \cup_{c=1}^{\infty} \Sigma^{cc}_{(\log n)^c}$. Note that despite its name, when this $\mathsf{PSPACE}^{\mathsf{cc}}$ class was first introduced in 1986 [7], the authors wrote

> "...we do not have a notion corresponding to space..."
>
> (Babai, Frankl, Simon, 1986 [7])

Similar to its Turing machine counterpart, a central problem concerning the communication polynomial hierarchy is whether the complexity classes at different levels of this hierarchy can be separated. Recent works by Aaronson, Wigderson [2] and Impagliazzo, Kabanets, Kolokolova [20] on the algebrization barrier put much more importance into these separations. For example, Aaronson and Wigderson showed that for two Turing machine complexity classes $\mathcal{K}_1$ and $\mathcal{K}_2$, if their communication complexity counterparts $\mathcal{K}_1^{cc}$ and $\mathcal{K}_2^{cc}$ satisfies $\mathcal{K}_1^{cc} \not\subseteq \mathcal{K}_2^{cc}$, then the inclusion $\mathcal{K}_1 \subseteq \mathcal{K}_2$ does not algebrize, that is to say, we need some *non-algebrizing* technique to prove the statement $\mathcal{K}_1 \subseteq \mathcal{K}_2$. A systematic investigation in [2] showed that all known nonrelativizing results based on arithmetization do algebrize. Therefore separation of such communication complexity classes constitute very strong evidence that resolving the relationship between the corresponding Turing machine classes is really hard.

In addition, Aaronson and Wigderson [2] showed that proving certain lower bounds in the communication version of interactive proof model implies the separation of NL and NP, an important open problem in complexity theory (see Theorem 7.2 in [2]). These results brought a lot of new attention to the study of communication polynomial hierarchy.

Separating the different levels in the communication polynomial hierarchy turns out to be an easier task than separating their counterparts in the Turing machine world. In particular, the classes P^{cc}, $\mathsf{NP}^{\mathsf{cc}}$ and $\mathsf{coNP}^{\mathsf{cc}}$ are easily separated using the equality function EQ and its complement.

Problem 1.6 (Equality, EQ)**.** *For two n-bit strings x and y,* $\mathrm{EQ}(x, y)$ *outputs* 1 *if and only if x and y are exactly the same.*

The separation $\mathsf{NP}^{\mathsf{cc}} \cup \mathsf{coNP}^{\mathsf{cc}} \subsetneq \mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ is showcased by the Greater-Than function GT presented on page 21. Based on the rectangle overlay characterization of $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$, and the classical rectangle cover characterization of $\mathsf{NP}^{\mathsf{cc}}$ and $\mathsf{coNP}^{\mathsf{cc}}$, this result can be easily obtained by simply observing the fact that GT only has exponentially large covers (in terms of input length n, see Figure 1-3 on page 23) [3], whilst it has a linear length overlay (again, in terms of input length n, see Figure 1-4 on page 23) based on the naive "comparing two numbers bit-by-bit" protocol.

As mentioned on page 26, Impagliazzo and Williams in 2010 [21] separated $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ from $\Sigma_2^{cc} \cap \Pi_2^{cc}$ using the List-Non-Equality function $\mathrm{LNE}_{\sqrt{n},\sqrt{n}}$. We provide a more intuitive proof for that result, together with a stronger separation with the Gap-Hamming-Distance function GHD.

Although the first level and the second level of the communication polynomial hierarchy can now be separated, climbing even higher in the hierarchy remains an open problem. In particular, it remains open whether $\mathsf{\Sigma}_2^{\mathsf{cc}}$ is equal to $\mathsf{\Pi}_2^{\mathsf{cc}}$. Our limited-memory communication model has additional connections to the higher levels of the polynomial hierarchy, besides the one between our memoryless model and $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ we have just discussed. These connections may help resolving open problems in the communication polynomial hierarchy. Specifically, we have:

Theorem 1.6 (restated and proved as Theorem 5.8 on page 85)**.** *For any constant k, if a function family $\{f_n^{cc}\}_{n=1}^{\infty}$ is in Σ_k^{cc}, then $\{f_n^{cc}\}_{n=1}^{\infty}$ can also be computed in our limited-memory model with message length $\mathsf{polylog}(n)$ and 3 memory states for Bob's permanent memory.*

Theorem 1.7 (restated and proved as Corollary 3.2 on page 50, and Theorem 5.14 on page 89)**.** *For any integer constant $w \geq 5$, the complexity class defined by message length $\mathsf{polylog}(n)$ and w memory states in our limited-memory model is exactly equivalent to $\mathsf{PSPACE}^{\mathsf{cc}}$. So is the complexity class defined by $\mathsf{polylog}(n)$ space in our general space-bounded communication model.*

This is the first space characterization of $\mathsf{PSPACE}^{\mathsf{cc}}$ that we know of.

Because of the connections between the communication polynomial hierarchy and the bounded-depth circuit classes (see Fact 2.6 on page 43), the proofs of Theorem 1.6 and Theorem 1.7 also imply the following:

Theorem 1.8 (restated and proved as Corollary 5.13 on page 89)**.** *If a boolean function family $f = \{f_n\}_{n=1}^{\infty}$ ($f_n : \{0,1\}^{2n} \to \{0,1\}$) is computable by an AC^0 circuit family, then under every input partition (to split the $(2n)$-bit input of every f_n into two n-bit inputs), the resulting communication problem is computable by a limited-memory protocol family with message length $\mathsf{polylog}(n)$ and 3 memory states.*

Theorem 1.9 (restated and proved as Theorem 5.15 on page 90)**.** *If a boolean function family $f = \{f_n\}_{n=1}^{\infty}$ ($f_n : \{0,1\}^{2n} \to \{0,1\}$) is computable by an NC^1 circuit family, then under every input partition (to split the $(2n)$-bit input of every f_n into two n-bit inputs), the resulting communication problem is computable by a limited-memory protocol family with message length $O(\log n)$ and 5 memory states.*

[3] Easily shown with classical fooling set argument, see Section 1.3 of the standard Kushilevitz-Nisan textbook[27] for an explanation of the fooling set argument.

More Connections to Other Related Models

Besides providing new tools for the study of the communication polynomial hierarchy, our model also provides a unified framework for understanding the information theoretic nature of a lot of other models and techniques. We now examine the connections between our model and other related models one by one.

First, the space-bounded Turing machines:

Here we add local preprocessing capability to a Turing machine and make it more relevant to the communication setting, we exempt the Turing machine from the cost of local computation on x and local computation on y, make it more focused on the "communication" or information flow between these two.

Definition 1.7 (Turing machine with local preprocessing)**.** *Turing machines with local preprocessing are of the form $\mathcal{M}(p(x), q(y))$. Here x and y are two n-bit input strings, p and q are arbitrary* preprocessing *functions on input x and y respectively, and $\mathcal{M}$ is a standard space $s(n)$ Turing machine which receives $p(x)$ and $q(y)$ on its input tape. Note the space-bound of $\mathcal{M}$, $s(n)$, is computed on the input length n of original inputs x and y, not on the output length of $p(x)$ or $q(y)$.*

The inspiration for the following theorem comes from a discussion with Buhrman and Speelman [1].

Theorem 1.10 (restated and proved as Theorem 3.1 on page 49)**.** *For any function $s(n) \geq \log n$, n being the input length, the set of functions computable in our general space-bounded communication model with space $O(s(n))$ is exactly the set of functions computable by a space $O(s(n))$ Turing machine augmented with local preprocessing.*

Even with just one-way communication, the general space-bounded communication model can still simulate space-bounded Turing machines (note this time without local preprocessing capabilities)

Theorem 1.11 (restated and proved as Theorem 3.3 on page 50)**.** *For any function $f : \{0,1\}^{2n} \to \{0,1\}$ and any $s \geq \log n$, if f can be computed by a Turing machine with space s and time t, then under any input partition for f, the resulting communication problem $f' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is always computable in our general space-bounded communication model with one-way communication, space $s + \log n + O(1)$ and communication $t \cdot n$.*

A consequence of the above simulation is that, proving any nontrivial (superlogarithmic) space lower bound for any explicitly defined boolean function in our general one-way communication model is quite hard, as already discussed in Section 1.2.1 on page 18.

In [17], Buhrman, Fehr, Schaffner and Speelman introduced the garden-hose model. Roughly speaking, in this model computation of a function $f(x, y)$ is realized by two players, Alice and Bob who are thought to be standing on two opposite ends of a wall with water-pipe-ends connecting the two sides. Given their respective inputs the players connect pipes on their respective ends and then Alice lets water

flow starting through a pipe on her side. On this input the value of the function f is 0 or 1 depending on whether the water ends up running on Alice's or on Bob's side. The authors of [17] proved that the garden-hose model is also equivalent to the space-bounded Turing machine with local preprocessing model. Therefore, their garden-hose model is equivalent to our model:

Theorem 1.12 (restated and proved as Corollary 3.4 on page 51)**.** *For any function $s(n) \geq \log n$, n being the input length, the set of functions computable in our general space-bounded communication model with space $O(s(n))$ is exactly the set of functions computable in the garden-hose model with $2^{O(s(n))}$ pipes.*

The above two models are equivalent to our general space-bounded communication complexity model. Next, we look at models that are more restricted. We have mentioned before that the most notable previous space-communication tradeoff results include Lam, Tiwari and Tompa's work on straight-line protocols [29] and Beame, Tompa and Yan's work on communicating branching programs [9]. Both models can be seen as restricted variants of our general space-bounded communication complexity model, in which the transition function takes a restricted form. But we notice that the proof techniques in these works are actually of information theoretic nature, and are therefore transferable to our model.

More precisely, the proof technique in Lam, Tiwari and Tompa's work on straight-line protocols [29] can be applied to any protocol in our general space-bounded communication model that satisfies the following conditions: first, the protocol is stuttering; second, the set of output bits produced in each step does not depend on the input value. The proof technique in Beame, Tompa and Yan's work on communicating branching programs [9] is even more widely applicable to our model. The technique can be applied to any stuttering protocol in our general space-bounded communication model. That means the same kind of space-communication tradeoff also holds in the stuttering variant of our general space-bounded communication model. As an example, we have:

Theorem 1.13 (restated and proved as Theorem 3.6 on page 52)**.** *Any stuttering protocol in our general space-bounded communication model computing the product of an $n \times n$ matrix and an n-vector over $GF(2)$ requires communication C and space S such that $C \cdot S = \Omega(n^2)$, as long as $S = o(n/\log n)$.*

1.3.2 One-Way Communication versus Two-Way Communication

In this section we present some interesting results concerning the comparison of the computation power of two-way communication models and one-way communication models.

One first result that compares one-way communication with two-way communication is about the limited-memory model, it can be quite surprising at a first glance.

Theorem 1.14 (restated and proved as Theorem 5.7 on page 84)**.** *For any $s(n) > \log n$, if a function family $\{f_n^{cc}\}_{n=1}^{\infty}$ can be computed with space $s(n)$ in the general*

space-bounded communication model, then it is computable in the limited-memory model with message length $O(s(n)^2)$ and 5 memory states.

This shows that the general space-bounded communication model with two-way communication model can always be simulated by the limited-memory model with one-way communication and moderate space overhead.

Our second result compares the general two-way communication model with the one-way memoryless model.

Theorem 1.15 (informally stated, see Theorem 4.2 on page 64 for formal version and proof)**.** *For any $0 < s(n) < \frac{n}{5}$, almost every boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that can be computed in the memoryless model with message length $s(n)+\lceil \log n \rceil$ is not computable in the general space-bounded model with space $s(n)$.*

This shows that with slightly more space ($\lceil \log n \rceil$), the seemingly severely restricted memoryless model will be able to compute many functions that are not computable by the general model with slightly less space. Here are some corollaries:

Corollary 1.16 (restated and proved as Corollary 4.3 on page 65)**.** *For any $n > 40$ and $0 < s < \frac{n}{5}$, there exist functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that are computable with space $s + \lceil \log n \rceil + \lceil \log s \rceil$ in the general space-bounded communication model, but not computable with space s in the general space-bounded communication model.*

Corollary 1.17 (restated and proved as Corollary 4.4 on page 65)**.** *For any $n > 40$ and $0 < s < \frac{n}{5}$, there exist functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that are computable with space $s+\lceil \log n \rceil+\lceil \log s \rceil$ in the memoryless model, but not computable with space s in the memoryless model.*

1.3.3 More Lower Bounds and Protocols

In this section, we present some more lower bound results and nontrivial protocols.

Lower Bounds

First, for non-boolean functions, even in the fluent variant of our general space-bounded communication model, which is the strongest one among all the (deterministic) models we introduce in this work, we are able to prove non-trivial space requirement lower bound for explicitly defined, albeit not very natural functions.

Problem 1.8 (All-Subset-Equality, ALL-EQ)**.** *Define the non-boolean function* ALL-EQ $: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2^n}$ *such that it computes equality on all subsets of corresponding bits in two n-bit strings x and y. Let $\{I_1, I_2, \ldots, I_{2^n}\}$ enumerate all subsets of $\{1, 2, \ldots, n\}$, and define the i-th output bit of* ALL-EQ *to be*

$$\text{ALL-EQ}_i(x,y) \stackrel{def}{=} \begin{cases} 1 & \text{if } \forall j \in I_i, x_j = y_j \\ 0 & \text{otherwise} \end{cases}$$

Theorem 1.18 (restated and proved as Theorem 3.9 on page 54)**.** ALL-EQ *requires* $\Theta(n)$ *space to be computed in the fluent variant of our general space-bounded communication model.*

Problem 1.9 (Equality-with-Combinatorial-Design, EQ-WITH-DESIGN$_k$)**.** *Define the non-boolean function* EQ-WITH-DESIGN$_k$ *(*k *is any positive integer) like* ALL-EQ*, except that instead of using all the* 2^n *subsets of* $\{1, 2, \ldots, n\}$*, we use a combinatorial design containing* p^k *such subsets, where* p *is a prime satisfying* $p^2 = n$*.*[4] *This family of subsets has the following two properties: first, each subset in this family is of size* p*; second, the intersection of every two subsets in this family is of size at most* k*.*[5] *These subsets defines* EQ-WITH-DESIGN$_k : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{p^k}$*.*

Theorem 1.19 (restated and proved as Theorem 3.10 on page 54)**.** *For every positive integer* k*,* EQ-WITH-DESIGN$_k$ *requires space* $\Theta(k \log n)$ *to be computed in the fluent variant of our general space-bounded communication model.*

In the boolean case, a nontrivial lower bound in the general space-bounded communication model for some explicitly defined function still eludes us. But by following Shannon [39], one can easily show:

Theorem 1.20 (informally stated, see Theorem 3.7 on page 53 for formal version and proof)**.** *Almost all boolean functions* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *require* $\Omega(n)$ *space to be computed in our general space-bounded communication model.*

Protocol Construction

In this section, we present some nontrivial results concerning the construction of efficient protocols in our space-bounded communication model.

First is a technique for composing multiple memoryless protocols:

Theorem 1.21 (restated and proved as Theorem 4.16 on page 74)**.** *For positive integers* n*,* c *and functions* $f_1, f_2, \ldots, f_c : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$*,* $h : \{0,1\}^c \to \{0,1\}$*, suppose for each* $i \in \{1, 2, \ldots, c\}$*,* f_i *can be computed by a message length* s_i *memoryless protocol* $\mathcal{P}_i$*, then the function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *defined as* $f(x, y) \stackrel{def}{=} h(f_1(x, y), f_2(x, y), \ldots, f_c(x, y))$ *is computable by a message length* $\sum_{i=1}^{c} s_i$ *memoryless protocol*

[4] We assume for simplicity that n is always the square of some prime number. Strictly speaking, it suffices to find a prime between $\lceil\sqrt{n}\rceil$ and $2\lceil\sqrt{n}\rceil$, the existence of such a prime number is guaranteed by Bertrand's postulate (see e.g. Ramanujan's work [35]).

[5] For a specific construction, we use the one-to-one mapping between $GF(p) \times GF(p)$ and $\{1, 2, \ldots, n\}$. Consider all polynomials of degree at most $k-1$ on $GF(p)$, they are all of the form $q(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \ldots + a_1 x + a_0$, in which all the coefficients $a_{k-1}, a_{k-2}, \ldots, a_1, a_0 \in GF(p)$. There are p^k such polynomials, and each such polynomial defines a size-p subset of $GF(p) \times GF(p)$: $\{(0, q(0)), (1, q(1)), \ldots, (p-1, q(p-1))\}$, which can be mapped into a subset of $\{1, 2, \ldots, n\}$. These subsets have the specified properties.

Note that in the memoryless model Bob has no way of remembering intermediate computational results, therefore the straightforward solution of simulating the protocol for each component function one by one does not work. The trick here is to try to run all these protocols together "in parallel" (see Section 4.6 on page 74).

Next, we have some techniques for constructing non-trivial limited-memory protocols with 2 memory states.

Theorem 1.22 (restated and proved as Theorem 5.1 on page 80)**.** *For communication problem $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, suppose $f(x,y)$ can be decomposed into $g(h_1(x,y), h_2(x,y), \ldots, h_k(x,y))$, where each $h_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ depends on at most l bits in x ($i \in \{1,2,\ldots,k\}$), then f can be computed in the limited memory communication model with message length $l + k + \lceil \log k \rceil$ and 2 memory states for Bob.*

This gives a nontrivial protocol for inner product functions IP_{p^k} on $GF(p^k)$ with $p^k \geq 3$,[6] because for this function, we can cut each of the two inputs into smaller blocks, compute the partial inner product for each pair of corresponding blocks, then sum them up.

Corollary 1.23. *For every prime power $p^k \geq 3$, the inner product function of two n-dimensional vector over $GF(p^k)$, IP_{p^k}, can be computed in our limited-memory communication model with space $O(\sqrt{n})$ and 2 memory states for Bob. Note p and k are considered to be constant here.*

A consequence of this is that, although a width-w branching program of length l (definition in Section 2.4) can always be simulated by a limited-memory protocol with message length $\lceil \log l \rceil + 1$ and w memory states for Bob (Fact 5.5 on page 83), the simulation in the reverse direction is not always possible.

Theorem 1.24 (restated and proved as Theorem 5.6 on page 84)**.** *The shortest width-2 branching program that correctly computes IP_3 has length $\Omega\left(\left(\frac{3}{2\sqrt{2}}\right)^n\right)$, whilst there is a limited-memory protocol with message length $O(\sqrt{n})$ and 2 memory states that correctly computes IP_3.*

The branching program length lower bound is proved by Shearer [40].

Another result for constructing nontrivial protocols concerning bounded-size boolean formulas (recall that formulas are circuits with maximum fan-out 1):

Theorem 1.25 (restated and proved as Theorem 5.2 on page 81)**.** *If a function $f : \{0,1\}^{2n} \to \{0,1\}$ can be computed with a size S boolean formula $\mathcal{C}$, then under any input partition, the resulting communication problem $f' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ can be solved in our limited-memory communication model with message length $4\left\lceil \sqrt{S} \right\rceil + \left\lceil \frac{\log S}{2} \right\rceil + 4$ and 2 memory states for Bob.*

[6] Problem 1.2 on page 25

Chapter 2

Overview of Standard Models of Computation

In this chapter, we fix our notation, introduce definitions of models that appeared previously in the literature, and briefly discuss their basic properties which are used later on. For a more complete treatment, we refer the reader to standard textbooks: [6] for general complexity theory and models of computation, and [27] for communication complexity, and [44] for circuit complexity.

2.1 Classical Communication Complexity

2.1.1 The Deterministic Model

This work considers only two-party communication complexity. In this basic setting, there are two computationally unlimited players, Alice and Bob. Alice has an n-bit input string $x = x_1 x_2 \dots x_n$ and Bob has another n-bit input string $y = y_1 y_2 \dots y_n$. They want to cooperatively compute a function $f(x, y)$ by communicating as little as possible between the two of them. Usually the function f in question is *boolean*, meaning its output is also a bit from the set $\{0, 1\}$. In some part of this work, we also study the case in which f is *non-boolean*, where the output of f can be a bit string of variable length. For now, we focus on the boolean function case.

A *deterministic* protocol $\mathcal{P}$ in the communication complexity model is defined by its associated *protocol tree* $\mathcal{T}$. Figure 2-1 depicts such a protocol tree $\mathcal{T}$ corresponding to a protocol for computing the Greater-Than function GT (Problem 1.1 on page 21), here n=2.

Each internal node v of $\mathcal{T}$ is either designated as an Alice-node (denoted in Figure 2-1 as "A"), with an associated boolean function $A_v(\cdot)$ on Alice's input x, or a Bob-node (denoted in Figure 2-1 as "B"), with an associated boolean function $B_v(\cdot)$ on Bob's input y. Each leaf node has a designated output value of either 0 or 1.

During the execution of the protocol, Alice and Bob traverse the tree starting from the root node v_0. Suppose Alice initiates the communication (that is, v_0 is an Alice-node, as in the case of the protocol tree in Figure 2-1). She starts by computing

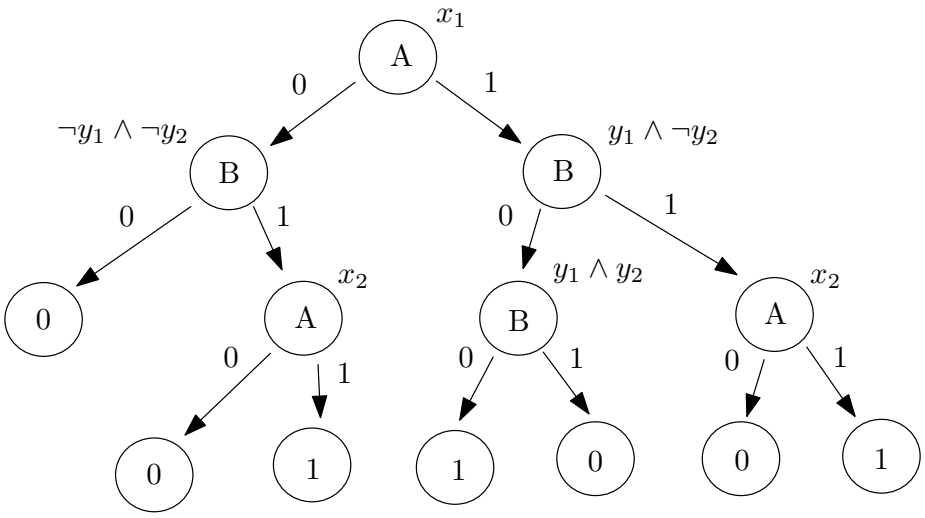


Figure 2-1: Protocol Tree for GT

the boolean function A_{v_0} associated with the root node (in the case of the protocol tree shown in Figure 2-1, this function is simply x_1). Then, she sends the output of the function $A_{v_0}(x)$ to Bob. Based on this one bit of communication both Alice and Bob descend to the corresponding child node of the currently traversed node and continue from there. When a leaf node is reached, Alice and Bob take the designated output value of this node as the protocol's final output.

Such a protocol is called *deterministic* for the obvious reason that every move of Alice and Bob is deterministic and totally predictable once their inputs x and y are fixed. The cost of such a protocol is defined to be $h(\mathcal{T})$, the height of the associated protocol tree, that is, the number of communication bits required in the worst case. The *deterministic communication complexity* of a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, denoted by $D(f)$, is defined to be the cost of the optimal deterministic protocol that correctly computes f.

We note that the communication complexity model is a *non-uniform* model. That is to say, if we need to compute a function family $\{f_n^{cc}\}_{n=1}^{\infty}$ with variable input length, then we must define a family of protocols, one for each input length n.

2.1.2 The Nondeterministic Model

Next we consider the *nondeterministic* variant of the communication complexity model.

In the nondeterministic communication complexity model, like in its better-known Turing machine counterpart, there is an all-powerful prover, who tries to convince Alice and Bob that $f(x, y) = 1$. Alice and Bob first receive a *proof* w (or as in commonly adopted terminology, a *witness*) from the prover. Then, they run a proof-specific *deterministic* protocol $\mathcal{P}_w$ to verify the proof. We say such a protocol correctly computes the target function $f(x, y)$ if Alice and Bob can be convinced only when

the output of $f(x, y)$ is actually 1, in other words:

$$f(x, y) = 1 \Leftrightarrow \exists w \text{ such that } \mathcal{P}_w(x, y) \text{ outputs } 1$$

The cost of the protocol is $\max_w (|w| + h(\mathcal{T}_w))$, where $|w|$ is the length of w when encoded as a bit string, and $h(\mathcal{T}_w)$ is the height of $\mathcal{P}_w$'s protocol tree $\mathcal{T}_w$. The *nondeterministic communication complexity* of a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, denoted as $N^1(f)$, is defined to be the cost of the optimal nondeterministic communication protocol as always. The *co-nondeterministic communication complexity* $N^0(f)$ of function f is defined as a complement, with the roles of output value 0 and output value 1 in the previous definition interchanged.

An observation about the nondeterministic/co-nondeterministic communication complexity model is that we can assume a normal form for every nondeterministic/co-nondeterministic protocol we encounter: it always uses exactly one bit of communication. Because the all-powerful prover always knows Alice and Bob's whole communication transcript given her proof, she can simply concatenate that to the end of the proof/witness. Then Alice can verify if her part of the communication is represented correctly in the transcript given by the prover and send her result to Bob. Bob, upon receiving Alice's result, verifies his part and makes a decision whether to accept.

2.1.3 The Randomized Model

In the randomized communication complexity model the players first draw some random bits r from a random source, these random bits are shared between the two of them by default and are called *public-coins*. Then, they execute a public-coin specific *deterministic* protocol $\mathcal{P}_r$. We say a randomized protocol satisfactorily approximates a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ if for every input pair $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, $\Pr_r[f(x, y) = \mathcal{P}_r(x, y)] \geq \frac{2}{3}$. That is, for every possible input pair the probability that the protocol chosen according to the random outcomes of public-coin tosses makes a mistake is smaller than $\frac{1}{3}$.

The cost of the protocol is the worst-case communication cost, that is, $\max_r h(\mathcal{T}_r)$, here $h(\mathcal{T}_r)$ is the height of $\mathcal{P}_r$'s protocol tree $\mathcal{T}_r$. The *(public-coin) randomized communication complexity* of function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, denoted as $R(f)$, is the cost of optimal (public-coin) randomized protocol. [1]

We can identify a public-coin randomized communication protocol as a probability distribution over a set of deterministic communication protocols.

[1] Another commonly studied variant is the *private-coin* model (this definition is consistent only with part of the literature). In this model, during the execution of public-coin specific protocol $\mathcal{P}_r$, the players also draw random bits that they do not share with each other by default (unless explicitly communicated in the protocol) a.k.a. *private-coins*. See for example section 3.1 of [27] for a formal definition. We do not deal with this model here.

2.1.4 Communication Matrices

We have introduced the concept of communication matrices and combinatorial rectangles for a communication problem $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ in Section 1.3.1 (page 21). After we have presented the formal definitions of the various communication complexity measures, we now examine on a more technical level how these combinatorial tools (combinatorial rectangles, rectangle partitions, rectangle covers etc.) connect with the complexity measures.

First, we note the central property that makes combinatorial rectangles so relevant to the study of communication complexity: for any deterministic protocol tree $\mathcal{T}$ corresponding to a protocol $\mathcal{P}$, and any node v in the tree $\mathcal{T}$, the following set is a combinatorial rectangle.

$$\mathcal{R}_v \stackrel{\text{def}}{=} \{(x,y) \mid \text{execution of } \mathcal{P} \text{ on input } (x,y) \text{ will reach } v\}$$

Furthermore, we note:

- In a deterministic protocol $\mathcal{P}$, suppose that the leaf nodes in its protocol tree $\mathcal{T}$ are $\{v_1, v_2, \ldots, v_l\}$. Then, the rectangles $\{\mathcal{R}_{v_1}, \mathcal{R}_{v_2}, \ldots, \mathcal{R}_{v_l}\}$ are all *monochromatic*. And these rectangles form a *partition* of the communication matrix.

  This gives part of the following characterization of $D(f)$, the *deterministic* communication complexity, in terms of $RP(f)$, the size of the minimum *rectangle partition*. The other part is trickier, see section 2.2 and section 2.3 of [27] for a proof.

  $$\lceil \log RP(f) \rceil \leq D(f) \leq O(\log RP(f))^2$$

- Since we can assume that in every nondeterministic/co-nondeterministic protocol, Alice and Bob verify the proof/witness they receive by exchanging one bit, then every witness in the nondeterministic (co-nondeterministic) protocol corresponds to one 1 (0) monochromatic rectangle. And the set of 1 (0) monochromatic rectangles that corresponds to the set of all witnesses in a nondeterministic (co-nondeterministic) protocol must form a *cover* for all the 1 (0) entries in the communication matrix.

  We also observe that whenever there is a monochromatic rectangle cover of all 1 (0) entries, we can construct a corresponding nondeterministic (co-nondeterministic) communication protocol. Alice and Bob simply guess (equivalent to having a witness) which rectangle in the cover covers the matrix entry of their input (x,y).

  Recall that we denote the smallest number of monochromatic rectangles to cover all the 1 (0) entries in the communication matrix of a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as $RC^1(f)$ ($RC^0(f)$). Then we have

  $$N^1(f) = \left\lceil \log RC^1(f) \right\rceil, N^0(f) = \left\lceil \log RC^0(f) \right\rceil$$

  That is actually how nondeterministic (co-nondeterministic) communication complexity is defined in some literature (for example, Definition 2.3 in [27]).

In this way *rectangle partition* and *rectangle cover* concepts basically characterizes the *deterministic*, *nondeterministic* and *co-nondeterministic* communication complexity. The *rectangle overlay* concept we introduce in this work generalizes these classical concepts, as demonstrated in Section 1.3.1.

2.2 The Communication Complexity Polynomial Hierarchy

Now we define the complexity classes that are related to the communication complexity polynomial hierarchy: Σ_k^{cc}, Π_k^{cc}, $\mathsf{PH}^{\mathsf{cc}}$, $\mathsf{PSPACE}^{\mathsf{cc}}$ and oracle query communication complexity classes. This hierarchy has recently regained attention because of its connection to the algebrization barrier in complexity theory (see Section 1.3.1 page 26).

The complexity classes in the communication polynomial hierarchy can be defined in terms of a (deterministic) game. Here Alice still only gets input x, and Bob still only gets input y, but they are merely referees in this setting, and there are two other players: the *prover* who wants to convince Alice and Bob that the output of the function $f(x, y)$ should be 1; and the *disprover* who wants to convince Alice and Bob of the opposite, that is, $f(x, y) = 0$. The prover and the disprover know both x and y, and they take a total of k turns to prove/disprove that $f(x, y) = 1$. After the prover and the disprover are done, the referees Alice and Bob will discuss about the conversation between the prover and the disprover and decide who wins. The protocol outputs 1 if and only if the prover has a winning strategy. In the game, if the prover speaks first then we call the protocol a Σ_k^{cc}-protocol, whereas if the disprover speaks first then we call the protocol a Π_k^{cc}-protocol. The complexity classes Σ_k^{cc} (Π_k^{cc}) are defined to be the set of functions $\{k_n^{cc}\}_{n=1}^{\infty}$ computable with "efficient" Σ_k^{cc} (Π_k^{cc}) protocols. Here is the formal definition:

Definition 2.1. *Given function $k(n) \leq \mathsf{polylog}(n)$, We say a family of functions $\{f_n^{cc}\}_{n=1}^{\infty}$ is in the complexity class $\Sigma_{(k(n))}^{cc}$ if there exist functions $\phi, \psi : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}$, such that for every pair of n-bit strings x and y*

$$f_n(x, y) \Leftrightarrow \exists u_1 \forall u_2 \exists u_3 \ldots Q_{k(n)} u_{k(n)} (\phi(x, u) \diamond \psi(y, u))$$

Here $u \stackrel{def}{=} u_1 u_2 u_3 \ldots u_{k(n)}$ is the concatenation of the $k(n)$ component bit strings, it should have length at most $\mathsf{polylog}(n)$. When $k(n)$ is odd, then $Q_{k(n)}$ stands for $\exists$ and $\diamond$ stands for $\wedge$; when $k(n)$ is even, then $Q_{k(n)}$ stands for $\forall$ and $\diamond$ stands for $\vee$.

We define the complexity class Π_k^{cc} analogously, by switching the roles of the $\forall$ and $\exists$ quantifiers and the roles of the $\wedge$ and $\vee$ boolean operators in the above definition.

We define $\mathsf{NP}^{\mathsf{cc}} = \Sigma_1^{cc}$ and $\mathsf{coNP}^{\mathsf{cc}} = \Pi_1^{cc}$.

We define $\mathsf{PH}^{\mathsf{cc}} \stackrel{def}{=} \cup_{k=1}^{\infty} \Sigma_k^{cc}$ (here k ranges over all constants *that are independent of input length n) and $\mathsf{PSPACE}^{\mathsf{cc}} \stackrel{def}{=} \cup_{c=1}^{\infty} \Sigma_{(\log n)^c}^{cc}$ (here again c ranges over all* constants*).*

That is, in $\mathsf{PH}^{\mathsf{cc}}$ we allow any constant number of alternations between the prover and the disprover, whereas in $\mathsf{PSPACE}^{\mathsf{cc}}$ we allow $\mathsf{polylog}(n)$ many alternations.

It is easy to see that the first level of the hierarchy, $\Sigma_1^{cc} = \mathsf{NP}^{\mathsf{cc}}$ and $\Pi_1^{cc} = \mathsf{coNP}^{\mathsf{cc}}$, is characterized by $\mathsf{polylog}(n)$ nondeterministic/co-nondeterministic communication complexity.

Fact 2.1. $\mathsf{NP}^{\mathsf{cc}} = \Sigma_1^{cc}$ *is exactly the set of functions* $f = \{f_n^{cc}\}_{n=1}^{\infty}$ *with* $\mathsf{polylog}(n)$ *nondeterministic communication complexity.* $\mathsf{coNP}^{\mathsf{cc}} = \Pi_1^{cc}$ *is exactly the set of function* $f = \{f_n^{cc}\}_{n=1}^{\infty}$ *with* $\mathsf{polylog}(n)$ *co-nondeterministic communication complexity.*

Analogously, we define:

Definition 2.2. P^{cc} *is the set of functions* $f = \{f_n^{cc}\}_{n=1}^{\infty}$ *with* $\mathsf{polylog}(n)$ *deterministic communication complexity.*

Example: Π_2^{cc} Protocol for LNE. To demonstrate the above definition, let us take a look at a concrete example. Recall the List-Non-Equality function defined in Problem 1.3 on page 25. The Π_2^{cc} protocol is simple: first the disprover tries to disprove that $\mathrm{LNE}_{k,l}(x, y) = 1$ by trying to find out index $i \in \{1, 2, \ldots, k\}$ such that $x^{(i)} = y^{(i)}$, then the prover tries to find out an index $j \in \{1, 2, \ldots, l\}$ such that $x_j^{(i)} \neq y_j^{(i)}$. It is easy to see from the definition of the function $\mathrm{LNE}_{k,l}$ that for every input pair (x, y), the prover has a winning strategy if and only if $\mathrm{LNE}_{k,l}(x, y) = 1$. In fact $\mathrm{LNE}_{k,l}$ is also in Σ_2^{cc}, but the proof is trickier (see [28]).

Next we define oracle query communication complexity classes:

Definition 2.3. *In a communication protocol with oracle* $\{\mathcal{O}_n^{cc}\}_{n=1}^{\infty}$*, the players Alice and Bob not only can communicate bit strings between the two of them, they can also make queries of the form* $\mathcal{O}_m(p(x), q(y))$*. That is, Alice computes a* local preprocessing *function* $p : \{0,1\}^n \to \{0,1\}^m$ *on her input* x*, and Bob computes a local* local preprocessing *function* $q : \{0,1\}^n \to \{0,1\}^m$ *on his input* y*, then they submit the output of* $p(x)$ *and* $q(y)$ *to the oracle* $\mathcal{O}$ *and immediately they both get the output of* $\mathcal{O}(p(x), q(y))$*. Such a query has associated cost of* $\log m$*, whilst a normal communication bit has associated cost of* 1 *as always.*

The $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ *class is defined to be the set of functions computable with such oracle query protocols with total cost* $\mathsf{polylog}(n)$ *and oracle* $\mathcal{O} \in \mathsf{NP}^{\mathsf{cc}}$*. Note that* $\mathsf{NP}^{\mathsf{cc}} = \Sigma_1^{cc}$ *is the set of functions with* $\mathsf{polylog}(n)$ *nondeterministic communication complexity.*

Like the deterministic communication protocols introduced in Section 2.1.1, oracle query protocols can also be presented using protocol trees. In the protocol tree of an oracle query protocol, in addition to Alice's nodes and Bob's nodes, we have a third type of nodes: *query nodes*. At each query node, Alice and Bob make a query of the form $\mathcal{O}_m(p(x), q(y))$, where p and q are Alice's and Bob's preprocessing functions respectively and $\mathcal{O}_m$ is the oracle query function for query input length m. After the query result is announced, Alice and Bob choose the child node to go to based on the query result rather than the communication bit.

Example: $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ Protocol for GT. To demonstrate the above definition, let us take a look at a $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ protocol for the Greater-Than function GT. Here we use the Not-EQual function $\mathrm{NEQ} \in \mathsf{NP}^{\mathsf{cc}}$ as our oracle.

Problem 2.4. *The* NEQ *function is the complement of the* EQ *function, for two n-bit strings x and y,* $\mathrm{NEQ}(x, y) \stackrel{def}{=} \neg\mathrm{EQ}(x, y)$.

For two n-bit strings x and y, to figure out if x is greater than y, we simply need to find the smallest index i such that $x_i \neq y_i$, then we can compare x_i and y_i and get the answer. To find this index i with the help of the NEQ oracle, we simply do a binary search: we first test if the first $\frac{n}{2}$ bits of x and the first $\frac{n}{2}$ bits of y are different with one query to NEQ, and if they are different, then we recurse on the first half of x and y, otherwise we recurse on the second half of x and y. After at most $\lceil \log n \rceil$ recursions, we will be able to find the index i we look for. The total cost of this protocol is clearly $O((\log n)^2)$, so it is a $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ protocol.

2.3 Boolean Circuits

The boolean circuit model is another non-uniform computational model like the communication complexity model.

Here, we present some basic definitions and tools from circuit complexity that are relevant to this work. For a more complete treatment, see [44].

First, the definition of a basic boolean circuit:

Definition 2.5 (Boolean Circuit)**.** *A* boolean circuit *with input length n is a directed acyclic graph with a unique sink. Each source node of the graph is labelled with an index $i \in \{1, 2, \ldots, n\}$. Each non-source node is labelled with one of the three basic boolean operations: $\wedge$ as logical-and/conjunction, $\vee$ as logical-or/disjunction, and $\neg$ as logical-not/negation. A node in the circuit is also called a* gate*, so we have input gates (source nodes), $\wedge$-gates, $\vee$-gates, $\neg$-gates and output gate (the unique sink). The* fan-in *of a gate is its in-degree. The* fan-out *of a gate is its out-degree. $\wedge$-gates and $\vee$-gates may have arbitrary fan-in, whilst $\neg$-gates must have fan-in* 1*. The* depth *of a circuit is the length of the longest path from one of its input gates to its output gate. The* size *of a circuit, denoted as $|\mathcal{C}|$ for circuit $\mathcal{C}$, is the number of gates it has.*

When evaluating a circuit on n-bit input x, each input gate with label i evaluates to x_i, each non-input gates takes input from all the gates with outgoing edges pointing at it and evaluates the boolean operation indicated by its label. The output of the circuit is the value evaluated by the output gate.

Here are some restricted classes of circuits.

Definition 2.6 (Boolean Formula)**.** *A boolean* formula *is a boolean circuit with maximum fan-out* 1*. That is, its underlying directed acyclic graph is a directed tree.*

Definition 2.7 (NC and AC)**.** *For every constant integer $k \geq 0$, the AC^k class include all circuit families $\{\mathcal{C}_n\}_{n=1}^{\infty}$ with polynomial size (in input length n), $O((\log n)^k)$ depth and unbounded fan-in.* [2] *An important special case is the class AC^0 which contains circuit families of polynomial size,* constant depth*, and unbounded fan-in.*

[2] Since the circuit model is non-uniform, to talk about asymptotics, we need to define a family of circuits $\{\mathcal{C}_n\}_{n=1}^{\infty}$, one for each input length. Here $\mathcal{C}_n : \{0,1\}^n \to \{0,1\}$.

For every constant integer $k \geq 1$, the NC^k class include all circuit families $\{\mathcal{C}_n\}_{n=1}^{\infty}$ with polynomial size (in input length n), $O((\log n)^k)$ depth and maximum fan-in 2.

Circuits are more appealing to analyze than Turing machines. Given the following connections between the circuit model and the Turing machine model, circuit complexity still provides an alternative route to the most sought-out goals in complexity theory: space and time lower bounds in the Turing machine world. These connections are very important for this thesis as well.

Theorem 2.2 (Theorem 2.8 in [44])**.** *Let $t(n) \geq n$, then if a function family $\{f_n\}_{n=1}^{\infty}$ can be computed by a Turing machine in time $t(n)$, then it can be computed by a family of circuits of size $O(t(n) \cdot \log t(n))$.*

Theorem 2.3 (Theorem 2.9 and Theorem 2.18 in [44])**.** *Let $s(n) \geq \log n$. If a function $\{f_n\}_{n=1}^{\infty}$ can be computed by a nondeterministic Turing machine in space $s(n)$, then it can be computed by a family of circuits of depth $O((s(n))^2)$ and fan-in 2.*

On the other hand, if a function $\{f_n\}_{n=1}^{\infty}$ can be computed by a family of circuits $\{\mathcal{C}_n\}_{n=1}^{\infty}$ of depth $s(n)$ and fan-in 2, then there is a space $O(s(n))$ Turing machine such that for each input length n and each input $x \in \{0,1\}^n$, the Turing machine takes x and the description of $\mathcal{C}_n$ (see Section 2.3 in [44] for formal definition of an admissible encoding scheme *for circuits) as inputs and correctly computes $f_n(x)$.*

In Definition 1.7 (page 29), we defined a model called Turing machine with local preprocessing. Now, we add the same kind of local preprocessing to circuits and make them more relevant in the communication setting.

Definition 2.8. *A boolean circuit with local preprocessing $\mathcal{C}^{cc}$ takes two n-bit inputs x and y. It is similar to a normal circuit, except that each input gate of it evaluates an arbitrary boolean function $p(x)$ on x or an arbitrary boolean function $q(y)$ on y, instead of just taking one of the n bits of x or one of the n bits of y.*

A boolean formula with local preprocessing is defined similarly.

The connections between Turing machines and boolean circuits as presented in Theorem 2.2 and Theorem 2.3 can clearly be extended to the case when both of them are augmented with arbitrary local preprocessing.

Theorem 2.4 (see also Theorem 2.8 in [44])**.** *Let $t(n) \geq n$, then if a function family $\{f_n\}_{n=1}^{\infty}$ can be computed by a Turing machine with local preprocessing in time $t(n)$, then it can be computed by a family of size $O(t(n) \cdot \log t(n))$ circuits with local preprocessing.*

Theorem 2.5 (see also Theorem 2.9 and Theorem 2.18 in [44])**.** *Let $s(n) \geq \log n$. If a function $\{f_n\}_{n=1}^{\infty}$ can be computed by a nondeterministic Turing machine with local preprocessing in space $s(n)$, then it can be computed by a family of depth $O((s(n))^2)$ fan-in 2 circuits with local preprocessing.*

On the other hand, if a function $\{f_n\}_{n=1}^{\infty}$ can be computed by a family of depth $s(n)$ fan-in 2 circuits with local preprocessing, then it can also be computed by a space $O(s(n))$ Turing machine with local preprocessing.

These two theorems can be proved in almost the exact same manner as their classical counterparts, except that whenever the classical proof mentions about an input bit, we replace it with an arbitrary local preprocessing function with boolean output. For the second part of Theorem 2.5, the space-bounded Turing machine with local preprocessing uses one of its preprocessing functions to produce the description of the required circuit description.

Boolean formulas with local preprocessing actually gives an alternative way of defining the communication complexity polynomial hierarchy.

Fact 2.6. *For $k(n) \leq \mathsf{polylog}(n)$, $\Sigma^{cc}_{k(n)}$ ($\Pi^{cc}_{k(n)}$) is exactly the set of functions computable by depth $k(n)+1$, size $2^{\mathsf{polylog}(n)}$ boolean formulas with $\vee$-gates ($\wedge$-gates) as output gates and local preprocessing.*

$\mathsf{PSPACE}^{\mathsf{cc}}$ *is exactly the set of functions computable by depth* $\mathsf{polylog}(n)$*, size* $2^{\mathsf{polylog}(n)}$*, and maximum fan-in* 2 *boolean formulas with local preprocessing.*

$\mathsf{PH}^{\mathsf{cc}}$ *is exactly the set of function computable by constant-depth, size* $2^{\mathsf{polylog}(n)}$ *boolean formulas with local preprocessing.*

This holds true because in Definition 2.1, we can simply replace an $\exists$ quantifier with an $\vee$ gate, a $\forall$ quantifier with an $\wedge$ gate, for each possible value of the quantified variable, we define a subtree/sub-formula.

With these tools, we can prove the following theorem, which gives another characterization of $\mathsf{PSPACE}^{\mathsf{cc}}$ in terms of "space". The inspiration comes from a personal discussion with Buhrman and Speelman [1]:

Theorem 2.7. $\mathsf{PSPACE}^{\mathsf{cc}}$ *is exactly the set of function families computable by space* $\mathsf{polylog}(n)$ *Turing machines with local preprocessing.*

Proof. First, we prove that every function family $\{f^{cc}_n\}^{\infty}_{n=1}$ that is computable by space $\mathsf{polylog}(n)$ Turing machine with local preprocessing is also in $\mathsf{PSPACE}^{\mathsf{cc}}$. Suppose the two preprocessing function of this Turing machine is $p(x)$ and $q(y)$. Then according to Theorem 2.3, there is a depth $\mathsf{polylog}(n)$ circuit family with local preprocessing that simulates this Turing machine, the proof is like for classical circuit families, except that our circuit family takes input bits from $p(x)$ and $q(y)$ using their own local preprocessing capability. Then according to Fact 2.6, we have $\{f^{cc}_n\}^{\infty}_{n=1} \in \mathsf{PSPACE}^{\mathsf{cc}}$.

Second, we prove that every function family $\{f^{cc}_n\}^{\infty}_{n=1}$ in $\mathsf{PSPACE}^{\mathsf{cc}}$ is computable by a space $\mathsf{polylog}(n)$ Turing machine with local preprocessing. By Fact 2.6, $\{f^{cc}_n\}^{\infty}_{n=1}$ is computable by a family of depth $\mathsf{polylog}(n)$ and fan-in 2 circuits with local preprocessing. Then by Theorem 2.5, $\{f^{cc}_n\}^{\infty}_{n=1}$ is computable by a space $O(s(n))$ Turing machine with local preprocessing. □

2.4 Branching Programs

The branching program processes its input bit-by-bit, and it has a space concept that implicitly corresponds to intermediate result storage. It is a non-uniform model like the communication complexity model and the circuit model.

Definition 2.9 (Branching Programs)**.** *A* branching program $\mathcal{B}$ *with n-bit input and boolean output is a directed acyclic graph with a unique source. Each sink of the graph is labelled with either 0 or 1. Each non-sink node is labelled with an index $i \in \{1, 2, \ldots, n\}$, and has two outgoing edges, one labelled with 0, the other labelled with 1.*

When executing this program on an n-bit input x, we traverse the nodes starting at the unique source node. Each time we traverse a non-sink node with label i, we read the i-th bit of the input x_i, and go to the node pointed by the outgoing edge of the currently traversed node labelled with x_i. When we reach a sink node, then the label of that sink node is the output of the branching program. The space *of a branching program is the logarithm of the number of its nodes.*

A branching program $\mathcal{B}$ is layered *if its nodes can be partitioned into several layers. That is, there is a family of non-empty subsets of $\mathcal{B}$'s nodes $L_0, L_1, L_2, \ldots, L_l$ such that $\cup_{j=0}^{l} L_j$ contains all of $\mathcal{B}$'s nodes, and no two different subsets L_{j_1} and L_{j_2} ($0 \leq j_1, j_2 \leq l$, $j_1 \neq j_2$) intersect with each other. And further more, for every layer L_j ($0 \leq j \leq l-1$), all outgoing edges of all the non-sink nodes in this layer should go to the next layer L_{j+1}. The* width *of such a layered branching program is $\max_j |L_j|$, here $|L_j|$ is the cardinality of L_j, that is, the number of nodes in L_j. The* length *of such a layered branching program is l, the number of layers minus 1. Such a layered branching program is* oblivious *if for every layer L_j ($0 \leq j \leq l-1$), all non-sink nodes in that layer have the same label i.*

In this work, we are mostly interested in oblivious layered branching programs of *constant width*. For a constant integer $w \geq 2$, we note that in a width-w branching program $\mathcal{B}$, if for every layer we give the nodes in that layer an order, then each layer (except for the last layer) can be modelled by a tuple $\langle i, f^{(0)}, f^{(1)} \rangle$, where $i \in \{1, 2, \ldots, n\}$, and $f^{(0)}, f^{(1)} : \{1, 2, \ldots, w\} \to \{1, 2, \ldots, w\}$. That is to say, if we reach the k-th node in this layer ($1 \leq k \leq w$), then we will go to the $f^{(x_i)}(k)$-th node in the next layer, where x_i is the i-th bit in the input x.

Fact 2.8. *For any constant integer $w \geq 2$ and $l \geq 1$, an oblivious branching program of width-w and length l can be defined as a sequence of l tuples: $\langle i_1, f_1^{(0)}, f_1^{(1)} \rangle$, $\langle i_2, f_2^{(0)}, f_2^{(1)} \rangle$, $\ldots$, $\langle i_l, f_l^{(0)}, f_l^{(1)} \rangle$, for each $1 \leq j \leq l$, $i_j \in \{1, 2, \ldots, n\}$, $f_j^{(0)}, f_j^{(1)} : \{1, 2, \ldots, w\} \to \{1, 2, \ldots, w\}$.*

For an n-bit input $x = x_1 x_2 \ldots x_n$, the output of the program on this input x is $f_l^{(x_{i_l})}(f_{l-1}^{(x_{i_{l-1}})}(\cdots f_1^{(x_{i_1})}(1)))$.

In Definition 1.7 (page 29) and Definition 2.8 (page 42), we add arbitrary local preprocessing capability to Turing machines and boolean circuits, we can do the same thing to branching programs, and make them more relevant in the communication setting.

Definition 2.10. *A branching program with local preprocessing $\mathcal{B}^{cc}$ takes two n-bit inputs x and y. It is like a normal branching program, except that each node of it branches based on the output of an arbitrary boolean function $p(x)$ on x or an arbitrary boolean function $q(y)$ on y, instead of just taking one of the n bits of x or one of the n bits of y.*

In 1989, Barrington discovered a surprising connection between bounded-depth circuit and bounded-width oblivious branching programs [8]. Here is a slightly generalized version of this theorem. We will use this in our work with circuits and branching programs augmented with local preprocessing capability.

Theorem 2.9 (Generalized Version of Barrington's Theorem)**.** *There exists a universal constant c such that if a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is computable by a depth d, maximum fan-in 2 circuit with local preprocessing $\mathcal{C}^{cc}$, then it can also be computed by a width 5, length c^d oblivious branching program with local preprocessing $\mathcal{B}^{cc}$.*

The proof follows closely the original proof given in Barrington's paper [8].

Chapter 3

The General Space-Bounded Communication Model

In this chapter we study our general space-bounded communication model. We first introduce the formal definition of this model, then we explore the connections between our model and several other related models including space-bounded Turing machines, the communicating branching programs [9] and the garden-hose model [17]. We will also prove several lower bound results in our model for both boolean and non-boolean functions.

3.1 Model Definition

First, the definition of the general space-bounded model.

Definition 3.1. *A protocol $\mathcal{P}$ in the* general space-bounded communication complexity *model for computing function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ which uses s bits of space, is defined as a pair of* transition functions*: $T_A, T_B : \{0,1\} \times \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^s \times \{\text{HALT}, 0, 1\} \times \{0, 1, *\}^m$.*

There are two players Alice and Bob. We use variables $M_A, M_B \in \{0,1\}^s$ to denote the content of Alice and Bob's s-bit space respectively. When executing the protocol, Alice and Bob receive local inputs $x, y \in \{0,1\}^n$ respectively. M_A and M_B are initialized to the string 0^s in the beginning. In each step, each player sends one bit of communication to the other player. [1] *Then Alice/Bob computes her/his transition function. Alice computes $(M'_A, h, r) \stackrel{def}{=} T_A(b, x, M_A)$, here M_A and M'_A denote the content of Alice's local memory before and after this step; b is the communication bit received from Bob; x is her local input; $h \in \{\text{HALT}, 0, 1\}$ denotes her decision whether to halt the execution of the protocol. If she decides to continue she decides which bit to send to Bob in the next step; and $r \in \{0, 1, *\}^m$ denotes her output statement of this step, for every $j \in \{1, 2, \ldots, m\}$, if the j-th bit of r $r_j \neq *$, then r_j is Alice's answer for the j-th bit in f's output. Symmetrically for Bob.*

[1] The first communicated bit is by convention 0.

We say such a protocol $\mathcal{P}$ correctly computes f if for every input pair (x, y) the following three conditions are met: (i) Alice and Bob halts in the same step; (ii) for every $j \in \{1, 2, \ldots, m\}$, the j-th bit of f's output is answered by either Alice or Bob at least once*; (iii) no output of Alice or Bob contains wrong answers.*

Note that in this model the players can only exchange single bits, not whole bit strings like in classical communication complexity. The reason is that if we allow the players to send whole bit strings, then the communication channel between them will somehow act like a temporary storage for communication bits, which contradicts our goal of quantifying the players' space requirement for storing communication bits through the concept of their local memory space. For one thing, one of the original motivations for introducing such space-bounded communication complexity models is to show that the trivial upper bound of $n + 1$ in classical communication complexity no longer holds in the space-bounded setting, as already explained in Section 1.2.1. But, if we allow the players to exchange whole bit strings, Alice would be able to simply put her entire input string x on the "communication channel", and Bob would read this string and compute the target function in one step, which defies our original goal.

Another remark we have for this definition is that we do not allow the players to give wrong answers for certain output bits in the beginning and then correct those wrong answers later; nor do we allow the players to have a "default" answer (either 0 or 1) for all output bits so that during the actual protocol execution they only need to give answers to those output bits with the opposite value. In particular, certain lower bound results in Section 3.5.2 would no longer hold if such behavior is allowed, as we will explain when we present those lower bound results.

The above is the more general definition of our model. Let us now consider some slightly restricted variants:

Definition 3.2 (General Stuttering Model)**.** *A general space-bounded communication protocol $\mathcal{P}$ with space s as defined in Definition 3.1 is said to be* stuttering*, if the output statement of each player in each step gives answer to at most one output bit. More precisely, that means the defining transition functions T_A, T_B both satisfy the following condition: for every $x \in \{0,1\}^n$, $M \in \{0,1\}^s$ and $b \in \{0,1\}$, suppose $T_{A/B}(b, x, M) = (M', h, r)$ then in the output statement $r \in \{0, 1, *\}^m$, at least $m - 1$ bits in r is $*$.*

We use the word fluent *to describe those protocols not subject to the above restriction.*

This stuttering variant of our model gives a very general information theoretic setting in which the proof techniques introduced in Paul Beame et al.'s communicating branching programs paper [9] are applicable. We will discuss this later on in Section 3.4.

Definition 3.3 (General One-Way Model)**.** *A general space-bounded communication protocol $\mathcal{P}$ with space s as defined in Definition 3.1 is said to have only* one-way *communication, if in the protocol only Alice can send bits to Bob, but Bob can not*

send bits back to Alice. More precisely, that means the function T_B in Definition 3.1 satisfies the following condition: for every $x \in \{0,1\}^n$, $M \in \{0,1\}^s$ and $b \in \{0,1\}$, suppose $T_B(b, x, M) = (M', h, r)$, then $h \in \{\text{HALT}, 0\}$. This restriction makes the communication bit computed by T_B meaning less. Since there is no way for Alice to know how far the computation has gone, we do not require Alice to halt together with Bob.

For the vast part of this work, we will actually focus on the *boolean* case, meaning the function to be computed, f, has output length $m = 1$. In this case, there is no point for Alice and Bob to continue once one of them has already given the answer. Therefore, without loss of generality we assume that such protocols always halt once the answer is given. In other words, for every $x \in \{0,1\}^n$, $M \in \{0,1\}^s$ and $b \in \{0,1\}$, suppose $T_{A/B}(b, x, M) = (M, h, r)$ and $r \neq *$ then we have $h = \text{HALT}$.

3.2 Connections to Space-Bounded Turing Machines

In Section 1.3.1, we presented two connections between our general space-bounded communication model and space-bounded Turing machines, one concerning the two-way model, the other concerning the one-way model. Here we present formal statements for them.

Theorem 3.1 (Theorem 1.10 restated [1])**.** *For any function $s(n) \geq \log n$, n being the input length, the set of boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ computable in our general (two-way) space-bounded communication model with space $O(s(n))$ is exactly the set of functions computable by a space $O(s(n))$ Turing machine augmented with local preprocessing.*

Proof. First, we prove that Turing machines with local preprocessing simulate protocols in our general two-way space-bounded communication model. Consider a protocol $\mathcal{P}$ defined in terms of transition functions T_A, T_B. We define preprocessing function $p(x)$ ($q(y)$) to be the output table (a generalization of truth table for boolean functions) of $T_A(\cdot, x, \cdot)$ ($T_B(\cdot, y, \cdot)$). That is, we enumerate all different values $b \in \{0,1\}$ and $M \in \{0,1\}^{O(s(n))}$ in lexicographic order, and concatenate the output of $T_A(b, x, M)$ ($T_B(b, y, M)$) for all these different input values together to form a "look-up table". Given these two look-up tables as input, the Turing machine can clearly simulate $\mathcal{P}$ in work space $O(s(n))$, it simply keeps track of the memory content of the two players and the communication bits in its work space and uses the look-up tables to figure out how the memory content and communication bits change. For the other direction, the two players of a space $O(s(n))$ protocol $\mathcal{P}$ cooperatively simulate a Turing machine $\mathcal{M}(p(x), q(y))$ with work space $O(s(n))$. The read-only head of the Turing machine scans back and forth on its input tape with $p(x)$ and $q(y)$ written on it side-by-side. Suppose initially the read head of the Turing machine points into the region occupied by $p(x)$, where Alice starts the simulation. Because of her unlimited local computation power she can compute every input bit the Turing machine wants to read from $p(x)$ on the fly and she can also simulate any computation done by the

Turing machine. Whenever the input tape read head of the Turing machine crosses over to the region occupied by $q(y)$, Bob takes over and Alice passes the content of the Turing machine's work tape to Bob. Every time the read head of the Turing machine moves back to the region occupied by $p(x)$, Alice takes over again, and so on. Each of Alice and Bob needs at most $O(s(n))$ space to carry out this simulation. □

Note that in Theorem 2.7 we showed that the complexity class $\mathsf{PSPACE}^{\mathsf{cc}}$ is characterized by functions computable with space $\mathsf{polylog}(n)$ Turing machines with local preprocessing. Therefore, we have the following corollary of the above theorem.

Corollary 3.2 (part of Theorem 1.7 restated)**.** $\mathsf{PSPACE}^{\mathsf{cc}}$ *is exactly the set of function families computable by space* $\mathsf{polylog}(n)$ *general two-way protocols.*

Theorem 3.3 (Theorem 1.11 restated)**.** *For any function* $f : \{0,1\}^{2n} \to \{0,1\}$ *and any* $s > \log n$*, if* f *can be computed by a Turing machine with space* s *and time* t*, then under any input partition for* f*, the resulting communication problem* $f' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *is always computable in our general space-bounded communication model with one-way communication, space* $s + \log s + 2\log n + O(1)$ *and communication* $t \cdot n$*.*

Proof. The simulation protocol proceeds in t phases, each phase corresponds to one time step of the simulated Turing machine $\mathcal{M}$. In each phase Alice sends the bits in her part of the input to Bob one-by-one in order. Bob simulates one step of $\mathcal{M}$'s computation. If the input tape read head of $\mathcal{M}$ points to Bob's part of the input, Bob simply simulates this step by himself; otherwise Bob takes the input bit from Alice and simulates this step. The total communication is clearly $t \cdot n$. Alice uses $\log n + O(1)$ space to count through her input. Bob uses s bits to keep track of the content of $\mathcal{M}$'s work tape, $\log s + O(1)$ bits for $\mathcal{M}$'s work tape read-write head position, $\log n + O(1)$ bits for $\mathcal{M}$'s input tape read head position, $O(1)$ space for $\mathcal{M}$'s internal memory and $\log n + O(1)$ to count through Alice's enumeration. Therefore the space requirement of the simulation protocol is within the stated upper bound. □

3.3 Connections to the Garden-Hose Model

We have already mentioned the garden-hose model in Section 1.3.1. This model was introduced in 2011 by Buhrman, Fehr, Schaffner and Speelman [17], with application to quantum position-verification scheme. In this model we have a water source and a number of parallel pipes running between two players Alice and Bob. Each pipe has one end on Alice's side and the other end on Bob's side. Given two inputs x and y to Alice and Bob respectively, Alice decides how to connect the pipe-ends on her side with hoses based on her input value x, and Bob does similar things based on his input value y. Alice needs to make sure that the water source is connected to at least one pipe-end, and they both make sure that each end of each pipe is connected to at most one hose. After they are done connecting the pipes and hoses, the water source will pump water into the system. The output of the system is defined to be

0 if water comes out from one of the unconnected pipe ends on Alice's side; it is 1 if water comes out on Bob's side. More formally:

Definition 3.4. *A* garden-hose *protocol with p pipes is defined by two functions $C_A : \{0,1\}^n \to P(\{0,1,2,\ldots,p\})^{\lfloor \frac{p+1}{2} \rfloor}$ and $C_B : \{0,1\}^n \to P(\{1,2,\ldots,p\})^{\lfloor \frac{p}{2} \rfloor}$, here $P(\{0,1,2,\ldots,p\})$ is the power set of $\{0,1,2,\ldots,p\}$, similarly for $P(\{1,2,\ldots,p\})$. These functions must satisfy the following conditions.*

- *for every $x \in \{0,1\}^n$ ($y \in \{0,1\}^n$), and every element $\mathcal{S}$ (a subset of $\{0,1,2,\ldots,p\}$ or $\{1,2,\ldots,p\}$) in the output of $C_A(x)$ ($C_B(y)$), either $\mathcal{S}$ is the empty set $\emptyset$ or $\mathcal{S}$ has exactly 2 elements;*
- *for every $x \in \{0,1\}^n$, there exists one element $\mathcal{S}$ (a subset of $\{0,1,2,\ldots,p\}$) in the output of $C_A(x)$ such that $0 \in \mathcal{S}$;*
- *for every $x \in \{0,1\}^n$ ($y \in \{0,1\}^n$), and every two elements $\mathcal{S}_1$ and $\mathcal{S}_2$ in the output of $C_A(x)$ ($C_B(y)$), $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$.*

On input pair (x,y), the output of the garden-hose protocol is defined as follows: find the longest integer sequence [2] *of the form $\langle a_0 = 0, a_1, a_2, \ldots, a_l \rangle$ such that $\{a_0 = 0, a_1\}$ is one of the elements in the output of $C_A(x)$, $\{a_1, a_2\}$ is one of the elements in the output of $C_B(y)$, and $\{a_2, a_3\}$ is again one of the elements in the output of $C_A(x)$, and so on. Then the output of the protocol is 0 if l is even, and it is 1 if l is odd.*

The garden-hose complexity *of a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, denoted as $GH(f)$, is defined to be the smallest integer p such that there is a garden-hose protocol with p pipes that correctly computes f.*

In their paper Buhrman et. al. proved that for any function $s(n) \geq \log n$, the set of functions with garden-hose complexity $2^{O(s(n))}$ is exactly the set of functions computable by space $O(s(n))$ Turing machines augmented with local preprocessing. Combining their result with Theorem 3.1, we have:

Corollary 3.4 (Theorem 1.12)**.** *For any function $s(n) \geq \log n$, the set of functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ with $2^{O(s(n))}$ garden-hose complexity is exactly the set of functions computable with space $O(s(n))$ general two-way communication protocols.*

3.4 Connections to Communicating Branching Programs

As mentioned in Section 1.3.1 (page 30), the communicating branching program model is one of the space-bounded communication models introduced in 1990 by Beame, Tompa and Yan [9]. In this model the authors used branching programs to model the two players in a communication complexity model. Because of this their model is not purely information theoretic like ours, as discussed in Section 1.2. Interestingly,

[2] Such a sequence is clearly unique.

when they prove their space-communication tradeoff results, the proof technique they use is in fact information theoretic. Therefore their proof technique is immediately transferable to our stronger model, and our model provides a broader setting in which their proof technique is applicable.

Let us first give a more formal definition to their model.

Definition 3.5. *A pair of communicating branching programs for computing function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^*$ is similar to a pair of standard branching programs (as defined in Definition 2.9), one operating on input x and the other operating on input y, but with the following differences:*

- *each non-sink node in the branching program can also be a* receive *node, which receives one bit of communication from the other branching program, instead of reading one bit of input. The traversal process will stall at this node until a communication bit is received, and the edge to traverse next depends on the communication bit received rather than an input bit.*

- *each edge in the branching program may also carry an optional* output statement *and an optional* send *command in addition to its 0/1 label that defines the traversal process. Each output statement may give answer to at most one bit in the output of f. Each* send *command is of the form send(0) or send(1), which sends the bit value to the other branching program. A program executing a* send *command is blocked until the other branching program reaches a* receive *node and the communication bit is actually received.*

- *since the output is produced by output statements, the sink nodes in the branching program no longer produce output, they merely halt the program.*

We say that the function f is correctly computed by such a pair of communicating branching programs if: for every input pair x, y, both branching programs halt, and each bit in the output of $f(x, y)$ is answered in at least one of the output statements executed by one of the branching programs, and none of the answers is wrong.

The authors proved several space-communication tradeoff results for communicating branching programs. For example, for matrix-vector multiplication, they have:

Theorem 3.5 (see Corollary 4.4 in [10])**.** *Any pair of communicating branching programs computing the product of an $n \times n$ matrix and an n-vector over $GF(2)$ requires communication C and space S such that $C \cdot S = \Omega(n^2)$, as long as $S = o(n/\log n)$.*

We observe that although the communicating branching program model is not information theoretic, the proof technique used in [9] to achieve the space-communication results is. Therefore, the same argument shows the following theorem:

Theorem 3.6 (Theorem 1.13 restated)**.** *Any stuttering general two-way space-bounded communication protocol computing the product of an $n \times n$ matrix and an n-vector over $GF(2)$ requires communication C and space S such that $C \cdot S = \Omega(n^2)$, as long as $S = o(n/\log n)$.*

3.5 Space Lower Bound Results

In this section we prove several space lower bound results in our general space-bounded model. These lower bounds show that certain functions are not computable at all within certain space bounds, no matter how much communication we are willing to pay.

We have obtained improved lower bound results in Section 4.5 (page 68), for the memoryless model.

3.5.1 The Boolean Case

The first space lower bound result concerns boolean functions of the form $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

Theorem 3.7 (Theorem 1.20 restated)**.** *Let $\epsilon \in (0,1)$ be a constant, The ratio of all input length n boolean functions of the form $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that can be computed by space $\epsilon \cdot n$ general two-way protocols approaches 0 as $n \to \infty$.*

We prove Theorem 3.7 using the following:

Lemma 3.8. *The number of input length n boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that can be computed by space s general two-way protocols is at most $2^{(s+2)\cdot 2^{n+s+2}}$.*

Proof. In Definition 3.1 a general two-way space-bounded protocol is defined by a pair of transition functions $T_A, T_B : \{0,1\} \times \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^s \times \{\text{HALT}, 0, 1\} \times \{0,1,*\}^m$, here the output length $m = 1$ and we can safely assume that an optimal protocol will halt whenever the output is produced. Therefore, for each of T_A and T_B, we have at most $2^{(s+2)\cdot 2^{n+s+1}}$ such transition functions. Each pair of such transition functions defines one protocol, which computes at most one boolean functions of the form $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ correctly, if it computes any boolean function at all. Therefore, the number of boolean functions computable by these protocols is upper bounded by $2^{(s+2)\cdot 2^{n+s+2}}$. □

Now we obtain Theorem 3.7 as a corollary.

Proof of Theorem 3.7. The total number of functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is $2^{2^{2n}}$, whilst the number of boolean functions computable by space $\epsilon \cdot n$ general two-way protocols is at most $2^{(\epsilon n+2)\cdot 2^{n+\epsilon n+2}}$, and

$$\lim_{n\to\infty} \frac{2^{(\epsilon n+2)\cdot 2^{n+\epsilon n+2}}}{2^{2^{2n}}} = 0$$

□

Barrier Ahead

Unfortunately, even though we can prove that almost all boolean functions require $\Omega(n)$ space to be computed in our general two-way model, explicitly constructing even one of them remains a very hard problem. According to Theorem 3.3, for an explicit function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, if we prove space lower bound $s(n)$ for it in our general one-way model, then we will also prove space lower bound $s(n)$ for it in the Turing machine world. The only natural functions we know that have interesting Turing machine space lower bounds of the form $\omega(\log n)$ are PSPACE-complete problems. Some less natural functions with such space lower bounds can be constructed using diagonalization ([42], cf. example Section 4.1.3 in [6]).

Suppose for example that we can prove space lower bound $\omega(\log n)$ for one of the PSPACE-complete problems (e.g. satisfiability of quantified boolean formulas, cf. Section 4.2 in [6]) then by Theorem 2.5 and Theorem 3.1 we can also prove that this problem does not have NC^1 circuits. In other words, we would be able to prove $\mathsf{PSPACE} \not\subseteq \mathsf{NC}^1$. The strongest result we know so far of this kind is Williams 2011 result that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$ [46]. Here NEXP is the set of functions computable by a nondeterministic Turing machine in exponential time, which is believed to be a larger set than PSPACE. ACC^0 is the set of circuits like AC^0 circuits but with additional MOD_m gates (for every possible m). ACC^0 is believed to be strictly weaker than NC^1. Therefore proving any $\omega(\log n)$ space lower bound of this kind would be a breakthrough in this direction.

3.5.2 The Non-Boolean Case

As mentioned in the boolean case, proving interesting space lower bounds for explicit functions in our general model is very challenging, even if we restrict the model to have only one-way communication. In the non-boolean case, the situation is better. We are able to prove some non-trivial space lower bounds for some naturally defined functions in the *fluent* variant of our general two-way model, which is the strongest space-bounded communication model introduced in this work.

Theorem 3.9 (Theorem 1.18 restated)**.** *The* ALL-EQ *function (Problem 1.8 on page 31) requires* $\Theta(n)$ *space to be computed in the fluent variant of our general space-bounded communication model.*

Theorem 3.10 (Theorem 1.19 restated)**.** *For every positive integer* k*,* EQ-WITH-DESIGN$_k$ *(Problem 1.9 on page 32) requires space* $\Theta(k \log n)$ *to be computed in the fluent variant of our general space-bounded communication model.*

We have already mentioned (in Section 3.1 on page 47) that we do not allow protocols in our general space-bounded model to set a "default value" for some output bits, or to have wrong answers for some output bits, only to correct them later. The reason is that, if we allowed this kind of behavior, then the above two lower bounds would no longer hold. We would have a space $\lceil \log n \rceil + 1$ protocol for both the ALL-EQ function and the EQ-WITH-DESIGN$_k$ function: the protocol sets "default

value" 1 for every input bit, in round i ($i = 1, 2, \ldots, n$) Alice sends i and x_i (the i-th bit in Alice's input x) to Bob, Bob outputs 0 for all subsets in the function definition that contain index i if x_i is not equal to y_i (the i-th bit in Bob's input y).

To prove these theorems we first introduce some notation. For two n-bit strings x and y, we have already defined the *Hamming distance* $HD(x, y)$ between these two string in Definition 1.4 on page 25 as the number of bit positions at which the corresponding bits in x and y are different. $HS(x, y) = n - HD(x, y)$, is the number of positions at which the corresponding bits are the same. $|x|_1$ is the number of 1 bits in x.

When executing a general two-way space-bounded protocol $\mathcal{P}$ on input pair (x, y), we denote the total number of 1 bits in all output statements produced by Alice (Bob) as $\|\mathcal{P}(x, y)\|_1^A$ ($\|\mathcal{P}(x, y)\|_1^B$). Note that if a bit in the output is answered several times by Alice (Bob), it will be counted multiple times in $\|\mathcal{P}(x, y)\|_1^A$ ($\|\mathcal{P}(x, y)\|_1^B$). We have the following two lemmas, one for ALL-EQ, the other for EQ-WITH-DESIGN$_k$. We will prove these lemmas later on.

Lemma 3.11. *If there is a protocol $\mathcal{P}$ that correctly computes* ALL-EQ*, then we have:*

- *either $\exists x \in \{0, 1\}^n$, such that*

$$\sum_{y \in \{0,1\}^n} \|\mathcal{P}(x, y)\|_1^A \geq \frac{3^n}{2}$$

- *or $\exists y \in \{0, 1\}^n$, such that*

$$\sum_{x \in \{0,1\}^n} \|\mathcal{P}(x, y)\|_1^B \geq \frac{3^n}{2}.$$

Lemma 3.12. *If there is a protocol $\mathcal{P}$ that correctly computes* EQ-WITH-DESIGN$_k$*, then we have*

- *either $\exists x \in \{0, 1\}^n$, such that*

$$\sum_{y \in \{0,1\}^n} \|\mathcal{P}(x, y)\|_1^A \geq 2^{p^2-p-1} \cdot p^k$$

- *or, $\exists y \in \{0, 1\}^n$, such that*

$$\sum_{x \in \{0,1\}^n} \|\mathcal{P}(x, y)\|_1^B \geq 2^{p^2-p-1} \cdot p^k.$$

With the above two lemmas we are ready to prove Theorem 3.9 and Theorem 3.10.

Proof of Theorem 3.9. For the upper bound part we note that with space $n + O(1)$ Alice can simply send her n-bit input x to Bob and rely on Bob to come up with the whole answer. Bob has enough space to store all the bits in x.

For the lower bound part we show the contrapositive. Suppose that there is a space s general two-way protocol $\mathcal{P}$ that correctly computes ALL-EQ and the transition function for Alice and Bob in $\mathcal{P}$ are T_A and T_B. [3] Assume without loss of generality that the clause of x in Lemma 3.11 is true and the input value that makes this condition true is $x_0 = 0^n$.

For every memory state $M_A \in \{0,1\}^s$ and every possible communication bit $b \in \{0,1\}$, denote the output of $T_A(b, x_0, M_A)$ by (M'_A, h, r), denote the number of 1 bits in r by $o^1(M_A, b)$, and denote the set of $y \in \{0,1\}^n$ such that the execution of $\mathcal{P}(x_0, y)$ reaches M_A with Alice receiving bit b from Bob by $Y(M_A, b)$. For every $y \in Y(M_A, b)$, $|\text{ALL-EQ}(x_0, y)|_1 \geq o^1(M_A, b)$, since $|\text{ALL-EQ}(x_0, y)|_1 = 2^{HS(x_0,y)}$, therefore $HS(x_0, y) \geq \log o^1(M_A, b)$. This means $|Y(M_A, b)| \leq 2^n / o^1(M_A, b)$.

On the other hand,

$$\begin{aligned}
& \sum_{y \in \{0,1\}^n} \|\mathcal{P}(x_0, y)\|_1^A \\
= & \sum_{M_A \in \{0,1\}^s} \sum_{b \in \{0,1\}} \sum_{y \in Y(M_A,b)} o^1(M_A, b) \\
= & \sum_{M_A \in \{0,1\}^s} \sum_{b \in \{0,1\}} |Y(M_A, b)| \cdot o^1(M_A, b) \\
\leq & \ 2^s \cdot 2 \cdot 2^n
\end{aligned}$$

Therefore $3^n/2 \leq 2^s \cdot 2 \cdot 2^n$, which implies $s \geq \log(1.5) \cdot n - 2$. □

Proof of Theorem 3.10. For the upper bound part we give the following straightforward protocol: Alice and Bob each has two counters, one $\lceil k \log p \rceil$ bits long [4], to enumerate through the p^k subsets in the function definition; another one, $\lceil \log p \rceil$ bits long, to enumerate through the bits in a particular subset. In each step the players look at the counters, and compare the corresponding bits in their inputs x and y.

For the lower bound part we also show the contrapositive. Suppose that there is a space s general two-way protocol $\mathcal{P}$ that correctly computes EQ-WITH-DESIGN$_k$. Like we did in the proof of Theorem 3.9, we assume, without loss of generality that the clause of x in Lemma 3.12 is true, and the input value that makes this condition true is $x_0 = 0^n$.

$$\sum_{y \in \{0,1\}^n} \|\mathcal{P}(x_0, y)\|_1^A \geq 2^{p^2 - p - 1} \cdot p^k \tag{3.1}$$

For every state $M_A \in \{0,1\}^s$ and every possible communication bit $b \in \{0,1\}$, we likewise define $o^1(M_A, b)$ and $Y(M_A, b)$ as in the proof of Theorem 3.9.

[3] See Definition 3.1.

[4] Note in the definition of EQ-WITH-DESIGN$_k$ (see Theorem 1.19) we have $p^2 = n$.

$$
\begin{aligned}
& \sum_{y\in\{0,1\}^n} \|\mathcal{P}(x_0,y)\|_1^A \\
= & \sum_{M_A\in\{0,1\}^s} \sum_{b\in\{0,1\}} \sum_{y\in Y(M_A,b)} o^1(M_A,b) \\
= & \sum_{M_A\in\{0,1\}^s} \sum_{b\in\{0,1\}} |Y(M_A,b)|\cdot o^1(M_A,b) \qquad (3.2)
\end{aligned}
$$

Suppose the family of subsets of $\{1,2,\ldots,n\}$ we use to define EQ-WITH-DESIGN$_k$ are $\{I_i\}_{i=1,2,\ldots,p^k}$. We define

$$\alpha(t) = \min_{\substack{i_1,i_2,\ldots,i_t\in\{1,2,\ldots,p^k\}\\ i_1,i_2,\ldots,i_t \text{ are all different}}} \left|\bigcup_{j=1}^{t} I_{i_j}\right|.$$

For every $M_A\in\{0,1\}^s$ and every $b\in\{0,1\}$,

$$o^1(M_A,b)\cdot|Y(M_A,b)| \le \max_{t\in\{1,2,\ldots,p^k\}} t\cdot 2^{p^2-\alpha(t)} \qquad (3.3)$$

And clearly we have:

First, $\alpha(t)$ is non-decreasing for $t\in\{1,2,\ldots,p^k\}$. In particular, for $t\in\{1,2,\ldots,\lceil p/2k\rceil\}$, $\alpha(t)$ is strictly increasing. Second, for every $t\in\{1,2,\ldots,p^k\}$, $\alpha(t)\ge tp-\binom{t}{2}k$.

Therefore for $t\in\{1,2,\ldots,\lceil p/2k\rceil\}$, $t\cdot 2^{p^2-\alpha(t)}$ is strictly decreasing.

$$
\begin{aligned}
& \max_{t\in\{1,2,\ldots,p^k\}} t\cdot 2^{p^2-\alpha(t)} \\
= & \max\Big(\max_{t\in\{1,2,\ldots,\lceil p/2k\rceil\}} t\cdot 2^{p^2-\alpha(t)}, \max_{t\in\{\lceil p/2k\rceil,\lceil p/2k\rceil+1,\ldots,p^k\}} t\cdot 2^{p^2-\alpha(t)}\Big) \\
\le & \max\Big(\Big(t\cdot 2^{p^2-\alpha(t)}\Big)|_{t=1}, \Big(\max_{t\in\{\lceil p/2k\rceil,\lceil p/2k\rceil+1,\ldots,p^k\}} t\Big)\cdot\Big(\max_{t\in\{\lceil p/2k\rceil,\lceil p/2k\rceil+1,\ldots,p^k\}} 2^{p^2-\alpha(t)}\Big)\Big) \\
= & \max\Big(2^{p^2-p}, p^k\cdot\Big(2^{p^2-\alpha(t)}\Big)|_{t=\lceil p/2k\rceil}\Big) \\
\le & \max\big(2^{p^2-p}, p^k\cdot 2^{p^2(1-3/8k)}\big) \\
= & 2^{p^2-p}
\end{aligned}
$$

Combine this with (3.1), (3.2), and (3.3), we have:

$$\begin{aligned} 2^{s+1} \cdot 2^{p^2-p} &\geq 2^{s+1} \cdot \max_{t \in \{1,2,\ldots,p^k\}} t \cdot 2^{p^2-\alpha(t)} \\ &\geq \sum_{M_A \in \{0,1\}^s} \sum_{b \in \{0,1\}} |Y(M_A, b)| \cdot o^1(M_A, b) \\ &= \sum_{y \in \{0,1\}^n} \|\mathcal{P}(x_0, y)\|_1^A \\ &\geq 2^{p^2-p-1} \cdot p^k \end{aligned}$$

Therefore, $2^{s+1} \geq p^k/2$, $s = \Omega(k \log n)$. [5] □

Proofs of Lemmas

Proof of Lemma 3.11. It is easy to see that

$$|\text{ALL-EQ}(x, y)|_1 = 2^{HS(x,y)}$$

By definition, if a protocol $\mathcal{P}$ correctly computes ALL-EQ then for every possible pair of inputs (x, y),

$$\|\mathcal{P}(x, y)\|_1^A + \|\mathcal{P}(x, y)\|_1^B \geq |\text{ALL-EQ}(x, y)|_1$$

Since $\mathcal{P}$ correctly computes ALL-EQ then we have

$$\sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} \left(\|\mathcal{P}(x, y)\|_1^A + \|\mathcal{P}(x, y)\|_1^B\right) \geq \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} |\text{ALL-EQ}(x, y)|_1$$

On the other hand:

$$\begin{aligned} & \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} |\text{ALL-EQ}(x, y)|_1 \\ = & \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} 2^{HS(x,y)} \\ = & \sum_{x \in \{0,1\}^n} \sum_{j=0}^{n} \left(\sum_{y \in \{\bar{y} | HS(x,\bar{y})=j\}} 2^{HS(x,y)} \right) \\ = & \sum_{x \in \{0,1\}^n} \sum_{j=0}^{n} \binom{n}{j} 2^j \\ = & \sum_{x \in \{0,1\}^n} 3^n \\ = & \ 2^n \cdot 3^n \end{aligned}$$

[5] Note in the definition of EQ-WITH-DESIGN$_k$ (see Theorem 1.19), we have $p^2 = n$.

Thus,

$$\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}\|\mathcal{P}(x,y)\|_1^A+\sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}\|\mathcal{P}(x,y)\|_1^B\geq 2^n\cdot 3^n$$

Therefore:

- either
$$\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}\|\mathcal{P}(x,y)\|_1^A\geq 2^{n-1}\cdot 3^n$$
- or,
$$\sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}\|\mathcal{P}(x,y)\|_1^B\geq 2^{n-1}\cdot 3^n$$

Then, by averaging we have:

- either $\exists x\in\{0,1\}^n$, such that
$$\sum_{y\in\{0,1\}^n}\|\mathcal{P}(x,y)\|_1^A\geq\frac{3^n}{2}$$
- or, $\exists y\in\{0,1\}^n$, such that
$$\sum_{x\in\{0,1\}^n}\|\mathcal{P}(x,y)\|_1^B\geq\frac{3^n}{2}$$

□

Proof of Lemma 3.12. Since $\mathcal{P}$ correctly computes EQ-WITH-DESIGN$_k$, we have

$$\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}\left(\|\mathcal{P}(x,y)\|_1^A+\|\mathcal{P}(x,y)\|_1^B\right)=\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}|\text{EQ-WITH-DESIGN}_k(x,y)|_1$$

Suppose the family of subsets of $\{1,2,\ldots,n\}$ we use to define EQ-WITH-DESIGN$_k$ is $\{I_i\}_{i=1,2,\ldots,p^k}$. We have

$$\begin{aligned}
&\sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}|\text{EQ-WITH-DESIGN}_k(x,y)|_1\\
=&\sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}|\{i\mid EQ_{I_i}(x,y)=1\}|\\
=&\sum_{y\in\{0,1\}^n}\sum_{i\in\{1,2,\ldots,p^k\}}|\{x\mid EQ_{I_i}(x,y)=1\}|\\
=&\sum_{y\in\{0,1\}^n}\sum_{i\in\{1,2,\ldots,p^k\}}2^{p^2-p}\\
=&\ 2^n\cdot p^k\cdot 2^{p^2-p}
\end{aligned}$$

By averaging like we did in the proof of Lemma 3.11, we have

- either $\exists x \in \{0,1\}^n$, such that

$$\sum_{y \in \{0,1\}^n} \|\mathcal{P}(x,y)\|_1^A \geq 2^{p^2-p-1} \cdot p^k$$

- or, $\exists y \in \{0,1\}^n$, such that

$$\sum_{x \in \{0,1\}^n} \|\mathcal{P}(x,y)\|_1^B \geq 2^{p^2-p-1} \cdot p^k$$

□

Chapter 4

Overlays and the Memoryless Communication Model

In Section 1.2.2 we introduced our memoryless communication model, and in Section 1.3.1 we discussed the fact that this model fully characterizes $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$. This is the communication complexity analog of the oracle Turing machine complexity class P^{NP}. In addition, we presented a new combinatorial tool called rectangle overlay that greatly facilitates the study of the memoryless model and the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ class. In particular, strong lower bounds can be shown through rectangle overlays, in very intuitive ways.

In this chapter, we will first present formal definitions of the memoryless model and of the rectangle overlay concept, and then explore the aforementioned connections in detail.

4.1 Definitions

4.1.1 The Memoryless Model

Definition 4.1 (The Memoryless Model)**.** *A* one-way memoryless *protocol with maximum message length s is defined by two functions: $A : \mathbb{Z}^+ \times \{0,1\}^n \to \{0,1\}^s$, which defines Alice's behavior; and $B : \{0,1\}^n \times \{0,1\}^s \to \{0,1,\perp\}$, which defines Bob's behavior.*

Alice receives input $x \in \{0,1\}^n$ and Bob receives input $y \in \{0,1\}^n$. The protocol proceeds in rounds. In round i ($i = 1, 2, \ldots$), Alice computes a message $\alpha \stackrel{def}{=} A(i,x)$ and sends it to Bob. Bob, upon receiving this message, computes $\beta \stackrel{def}{=} B(y,\alpha)$. If $\beta \in \{0,1\}$, he outputs β and the protocol ends. If $\beta = \perp$, he does not output anything and the protocol proceeds to round $i+1$.

$\mathsf{SPACE}_{\mathsf{LESS}}[s]$ *is the set of functions computable by a one-way memoryless protocol with maximum message length s. The* memoryless complexity *of a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, denoted by $S(f)$, is the smallest s such that $f \in \mathsf{SPACE}_{\mathsf{LESS}}[s]$.*

A few remarks are in order about the above definition: first, throughout this work,

a memoryless protocol always only has one-way communication (from Alice to Bob), therefore most of the time, we will simply refer to the one-way memoryless model as the memoryless model, omitting the word "one-way"; second, in a memoryless protocol, it is understood from the definition that at the end of each round, if Bob chooses to continue to the next round without any output, then he will always forget Alice's message for this round completely and start anew in the next round, that is exactly the reason why we call this model "memoryless"; and thirdly, given the limited message length s, Alice has at most 2^s different messages to send to Bob, once she has exhausted all her possible messages there is no point to continue the protocol. Therefore, without loss of generality we assume that every memoryless protocol with message length s always halts within 2^s rounds.

We further note that just like when we talk about the general communication models in the last chapter, when we talk about the memoryless model we mostly focus on whether a function is computable at all with certain message length (space bound), without caring too much about the total communication cost.

Alternative Definition in Terms of Oblivious Space

The memoryless model can also be defined as a restricted variant of the general one-way model (see Definition 3.3 on page 49), in which all of Bob's space is "oblivious" and can only be used to compress information received from Alice without mixing-in any information about his own input.

Definition 4.2 (The One-Way Oblivious Model)**.** *The one-way oblivious communication model with s bits of space is a special case of the general one-way model defined in Definition 3.3, in which Bob's* transition function *T_B is split into two: $T_B^o : \{0,1\}^s \times \{0,1\} \to \{0,1\}^s$ for updating the content of Bob's oblivious space, and $T_B^f : \{0,1\} \times \{0,1\}^n \times \{0,1\}^s \to \{0,1,\perp\}$ for making halting/output decision (here we only deal with the case where output length $m = 1$).*

We use M_A and M_B^o to denote the content of Alice's local space and Bob's local oblivious *space. In each step of the protocol, Alice executes her transition function as usual (by definition of one-way protocols, Definition 3.3, the communication bit she receives from Bob in each step is always 0). Bob executes $M_B^{o\prime} = T_B^o(M_B^o, b)$ to update the content of his oblivious space, where M_B^o and $M_B^{o\prime}$ denote the old and new content of Bob's oblivious space before and after this step and b is the communication bit received from Alice. In addition, Bob executes $\beta = T_B^f(b, y, M_B^o)$ (here y is Bob's local input) to make halting/output decision: if $\beta \in \{0,1\}$, Bob outputs β and the protocol ends; if $\beta = \perp$, Bob does not output anything and the protocol proceeds to the next step.*

The basic idea is that Bob does not mix-in any information about his own input y into the content of his oblivious space. Therefore in each step of the protocol execution, the content of Bob's oblivious space is always just a compressed version of Alice's input x. In other words, in each step during the oblivious protocol execution, Alice always knows exactly what's in Bob's oblivious space. Therefore an oblivious protocol $\mathcal{P}$ with s bits of space (as defined in Definition 4.2) can always be simulated

by a memoryless protocol $\mathcal{P}'$ with message length $s+1$ (as defined in Definition 4.1): in each round of $\mathcal{P}'$ Alice simply computes the content of Bob's oblivious space and the communication bit according to $\mathcal{P}$, and sends that to Bob in her message. On the other hand, a memoryless protocol $\mathcal{P}'$ with message length s can always be simulated by an oblivious protocol $\mathcal{P}$ with $s + \lceil \log s \rceil$ bits of space: Alice simply sends her messages in $\mathcal{P}'$ bit-by-bit to Bob. This shows that Definition 4.1 and Definition 4.2 are equivalent for all the cases we deal with in this thesis. We prefer Definition 4.1 because it is easier to handle and it it is more intuitive.

Because the whole point of the oblivious model is to keep the content Bob's oblivious space free from information about Bob's local input. It is important that we only have one-way communication. If we have two-way communication, any meaningful communication from Bob to Alice needs to carry information about Bob's local input. Once Alice learns partial information about Bob's input and keeps that information in her local space, Alice can mix-in that information into her communication to Bob, thus defeating the original point of the model. Consequently we also only have one-way communication in the memoryless model.

4.1.2 Rectangle Overlay

Next, we define our new combinatorial tool, rectangle overlay, and a related complexity measure, *overlay complexity*.

Definition 4.3 (Rectangle Overlay)**.** *For a positive integer n, a* combinatorial rectangle $\mathcal{R}$ *is defined to be the Cartesian product of any two subsets $X, Y \subseteq \{0,1\}^n$: $\mathcal{R} \stackrel{def}{=} X \times Y$. And a* rectangle overlay *(or simply an* overlay*) of length l is defined to be an ordered sequence of tuples $(\mathcal{R}_1, b_1)$, $(\mathcal{R}_2, b_2)$, ..., $(\mathcal{R}_l, b_l)$, where for every $i \in \{1, 2, \ldots, l\}$, $\mathcal{R}_i$ is a combinatorial rectangle, and $b_i \in \{0,1\}$. This sequence of tuples must satisfy: $\cup_{i=1}^{l} \mathcal{R}_i = \{0,1\}^n \times \{0,1\}^n$.*

An overlay defines/computes a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ in a natural way: for each input pair (x, y), suppose $i_0 = \min \{i \mid (x, y) \in \mathcal{R}_i\}$. We define the value of $f(x, y)$ to be b_{i_0}.

For a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we define the overlay number *of f, denoted by $RO(f)$, as* $\min \{l \mid$ *there is an overlay of length l that correctly computes $f\}$*

The "rectangle overlay" name stems from the intuitive notion of forming an "overlay" of "combinatorial rectangles" in the communication matrix of a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$; as demonstrated in Section 1.3.1. We have also observed in that section that our rectangle overlay concept is obviously a generalization of the classical rectangle partition and rectangle cover concepts. In particular, we have:

Fact 4.1. *For any positive integer n and any function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,*

$$RO(f) \leq \min \left(RC^1(f), RC^0(f) \right) + 1 \ .$$

Like classical communication complexity, we can also introduce public-coin randomness into our memoryless model.

Definition 4.4 (The Public-Coin Randomized Memoryless Model)**.** *In a public-coin randomized memoryless protocol $\mathcal{P}$ with maximum message length s, Alice and Bob start by drawing random bits r from a shared/public random source, therefore we call these random bits* public-coins*. Then, they execute a public-coin specific* deterministic *memoryless protocol $\mathcal{P}_r$ with maximum message length s. We say that $\mathcal{P}$ satisfactorily approximates a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ if for every input pair $(x,y) \in \{0,1\}^n \times \{0,1\}^n$, we have $\Pr_r[\mathcal{P}_r(x,y) = f(x,y)] \geq \frac{2}{3}$. That is to say, for every possible input pair (x,y), the probability that the protocol chosen according to the random results of public-coin tosses makes a mistake should be smaller than $\frac{1}{3}$.*

The randomized memoryless complexity *of a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, denoted as $S^R(f)$, is the smallest s such that there is a public-coin randomized memoryless protocol with message length s that satisfactorily approximates f.*

Just as in classical communication complexity a public-coin randomized memoryless protocol can be identified by a probability distribution over a set of deterministic memoryless protocols.

4.2 Memory Hierarchy Theorems

The memoryless model may seem rather weak at a first glance, but the next theorem hints the opposite: with slightly more space, the memoryless model can actually beat the general two-way model.

Theorem 4.2 (see also Theorem 1.15)**.** *For any $n > 40$ and $0 < s(n) < \frac{n}{5}$, the number of boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ computable with space $s(n)$ general two-way protocols is always smaller than the number of boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ computable with message length $s(n) + \lceil \log n \rceil$ one-way memoryless protocols, and the ratio of these two numbers approaches 0 as $n \to \infty$.*

Proof. It is easy to see that a boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that depends only on the first s bits of its first n-bit input can always be computed with a one-way memoryless protocol with message length s. This is because Alice can send the first s bits of her input in one message to Bob and Bob can compute the function in one round. The number of distinct such functions is $2^{2^{n+s}}$. Therefore, the number of boolean functions that can be computed by message length s one-way memoryless protocols is at least $2^{2^{n+s}}$.

On the other hand, by Lemma 3.8, the number of boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that can be computed by space s two-way general protocols is at most $2^{(s+2)\cdot 2^{n+s+2}}$.

For $n > 40$ and $0 < s(n) < \frac{n}{5}$, clearly $2^{2^{n+s(n)+\lceil \log n \rceil}} > 2^{(s(n)+2)\cdot 2^{n+s(n)+2}}$, and

$$\lim_{n\to\infty} \frac{2^{(s(n)+2)\cdot 2^{n+s(n)+2}}}{2^{2^{n+s(n)+\lceil \log n \rceil}}} = 0$$

□

We note that a one-way memoryless protocol $\mathcal{P}$ with message length s can always be simulated by a general two-way protocol $\mathcal{P}'$ with space $s+\lceil \log s \rceil$. In $\mathcal{P}'$, Alice uses an s-bit counter to keep track of the round number, and Bob uses s bits of memory space to simulate the s bit message buffer in $\mathcal{P}$. In addition, each uses a $\lceil \log s \rceil$ bit long counter to communicate each of Alice's messages bit-by-bit to Bob. Therefore we have:

Corollary 4.3 (Corollary 1.16 restated)**.** *For any $n > 40$ and $0 < s < \frac{n}{5}$, there exist boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ computable with space $s + \lceil \log n \rceil + \lceil \log s \rceil$ general two-way protocols, but not computable with space s general two-way protocols.*

Corollary 4.4 (Corollary 1.17 restated)**.** *For any $n > 40$ and $0 < s < \frac{n}{5}$, there exist boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ computable with message length $s + \lceil \log n \rceil + \lceil \log s \rceil$ one-way memoryless protocols, but not computable with message length s one-way memoryless protocols. In other words,* $\mathsf{SPACE}_{\mathsf{LESS}}[s] \subsetneq \mathsf{SPACE}_{\mathsf{LESS}}[s + \lceil \log n \rceil + \lceil \log s \rceil]$.

4.3 Overlays and the Memoryless Model

Now, we are ready to prove that a function's *overlay complexity* fully characterizes its *memoryless complexity*.

Theorem 4.5 (Theorem 1.1 restated)**.** *For any positive integer n and any function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we have $S(f) \leq \lceil \log (RO(f)) \rceil \leq 2S(f) + 1$.*

Proof. We first prove the easier inequality: $S(f) \leq \lceil \log (RO(f)) \rceil$.

Let $l = RO(f)$. By definition there should be an overlay $(\mathcal{R}_1, b_1), (\mathcal{R}_2, b_2), \ldots, (\mathcal{R}_l, b_l)$ that computes f. Based on this we can construct a one-way memoryless protocol with messages of length at most $s \stackrel{\text{def}}{=} \lceil \log l \rceil$. For each $1 \leq i \leq l$, by definition we can write $R_i = X_i \times Y_i$, where $X_i, Y_i \subseteq \{0,1\}^n$. Given input $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, we enumerate the set of all X_i's such that $x \in X_i$ as $\{X_{i_1}, X_{i_2}, \ldots, X_{i_{l'}}\}$, here $i_1 < i_2 < \ldots < i_{l'}$. We define our protocol using the following two functions: $A(j, x) \stackrel{\text{def}}{=} i_j$ for Alice and

$$B(y, i) \stackrel{\text{def}}{=} \begin{cases} b_i & \text{if } y \in Y_i \\ \perp & \text{if } y \notin Y_i \end{cases}$$

for Bob. By Definition 4.1, this defines a memoryless protocol with message length at most $\lceil \log l \rceil$ because obviously $l' \leq l$.

Second, we prove the other inequality: $\lceil \log (RO(f)) \rceil \leq 2S(f) + 1$.

Suppose that there is a memoryless protocol with maximum message length $s \stackrel{\text{def}}{=} S(f)$ that computes f. We show that there is a rectangle overlay of length at most 2^{2s+1} that computes f. By Definition 4.1, there are functions A and B such that in the i-th round, Bob computes $B(y, A(i, x)) \in \{0, 1, \perp\}$ to determine whether to

give the final answer of either 0 or 1, or continue to the next round. Furthermore, we can always assume that the protocol will halt within 2^s rounds without loss of generality. For each tuple $(i, \alpha, b) \in \{1, 2, \ldots, 2^s\} \times \{0,1\}^s \times \{0,1\}$, we define $X_{i,\alpha,b} \stackrel{\text{def}}{=} \{x \mid A(i,x) = \alpha\}$, $Y_{i,\alpha,b} \stackrel{\text{def}}{=} \{y \mid B(y,\alpha) = b\}$ and $R_{i,\alpha,b} \stackrel{\text{def}}{=} X_{i,\alpha,b} \times Y_{i,\alpha,b}$.

Note that for any fixed i, the rectangles $R_{i,\alpha,b}$ are disjoint for different values of α and b. We order the rectangles $\{R_{i,\alpha,b}\}_{(i,\alpha,b)}$ in the increasing order of i and color each rectangle $R_{i,\alpha,b}$ with color b. It is easy to verify that such a rectangle overlay correctly computes f, and it clearly has length 2^{2s+1}. □

4.4 $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ and the Memoryless Model

Now, let us prove that the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ complexity class defined in Definition 2.3 is fully characterized by the memoryless complexity introduced above.

Theorem 4.6 (Theorem 1.2 restated)**.** $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}} = \mathsf{SPACE}_{\mathsf{LESS}}[\mathsf{polylog}(n)]$.

For the proof, we need to define the following function.

Problem 4.5 (Intersect, INT)**.** *For two n-bit strings $x = x_1 x_2 \ldots x_n$ and $y = y_1 y_2 \ldots y_n$, $\mathrm{INT}(x,y) = 1$ if and only if there is an index $i \in \{1, 2, \ldots, n\}$ such that $x_i = y_i = 1$.*

This function is complete for NP^{cc} under the so-called "rectangle reduction". Please refer to [7] for more details.

Proof. We start with the easier direction. Suppose there is a function $f = \{f_n^{cc}\}_{n=1}^{\infty}$ such that $f \in \mathsf{SPACE}_{\mathsf{LESS}}[\mathsf{polylog}(n)]$, or in other words $S(f) \in \mathsf{polylog}(n)$. We show that such a function f is always in $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ by constructing a $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ protocol $\mathcal{P}$ for f. The oracle we will use in $\mathcal{P}$ is the $\mathsf{NP}^{\mathsf{cc}}$-complete function INT defined above.

By Theorem 4.5 each f_n^{cc} has a rectangle overlay $(\mathcal{R}_1, b_1), (\mathcal{R}_2, b_2), \ldots, (\mathcal{R}_{\ell(n)}, b_{\ell(n)})$ where $\ell(n) \leq 2^{\mathsf{polylog}(n)}$. Each rectangle can be represented by a Cartesian product: $\mathcal{R}_i = X_i \times Y_i$ for each $i \in \{1, 2, \ldots, \ell(n)\}$. In the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ protocol $\mathcal{P}$ we construct, Alice and Bob first preprocess their respective inputs x and y into two $\ell(n)$-bit strings $\hat{x}, \hat{y} \in \{0,1\}^{\ell(n)}$: for each index $i \in \{1, 2, \ldots, \ell(n)\}$, set $\hat{x}_i$ to be 1 if $x \in X_i$ and 0 otherwise; and we define $\hat{y}$ analogously. Given these two $\ell(n)$-bit strings, Alice and Bob need to find the smallest index i such that $\hat{x}_i = \hat{y}_i = 1$ and output b_i. They can find this i using binary search, by querying INT at most $\log(\ell(n))$ times, and each query has length at most $\ell(n)$. Given that $\ell(n) \leq 2^{\mathsf{polylog}(n)}$, our protocol $\mathcal{P}$ is a proper $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ protocol ($\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ is defined in Definition 2.3 on page 40).

For the other direction, suppose $f = \{f_n^{cc}\}_{n=1}^{\infty} \in \mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$. We show that $f \in \mathsf{SPACE}_{\mathsf{LESS}}[\mathsf{polylog}(n)]$ by constructing a one-way memoryless protocol with maximum message length $\mathsf{polylog}(n)$.

For each f_n^{cc}, let $\mathcal{P}$ be the corresponding $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ protocol with $\mathcal{T}$ as its protocol tree. According to Definition 2.3, the depth of $\mathcal{T}$ is at most $\mathsf{polylog}(n)$, and every query node in $\mathcal{P}$ makes an oracle query to an $\mathsf{NP}^{\mathsf{cc}}$-function of input length at most $2^{\mathsf{polylog}(n)}$. As already discussed in Section 2.1.4, for every $\mathsf{NP}^{\mathsf{cc}}$-function Q, there is a

1-cover of Q consisting of at most $2^{\mathsf{polylog}(n)}$ 1-monochromatic rectangles $\{\mathcal{R}_i\}$, such that for each input pair (x, y), $Q(x, y) = 1$ if and only there is a rectangle $\mathcal{R}_i$ in the cover such that $(x, y) \in \mathcal{R}_i$. The index i plays the role of a *witness* for (x, y). Each witness can clearly be encoded into a bit string of length at most $\mathsf{polylog}(n)$.

For each input pair (x, y) to f_n^{cc}, we can describe the computational history the original $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ protocol $\mathcal{P}$ will follow in $\mathcal{T}$ as follows: first we use a $\mathsf{polylog}(n)$-bit string p to denote all the communication bits and query answers along the way, from the root of $\mathcal{T}$ down to one of its leaf nodes; then we scan this path denoted by p, for each query that answers 1 along the way, we concatenate one of the witnesses for (x, y). This gives a string $(p, w_1, w_2, \ldots, w_t)$, in which $p, w_1, w_2, \ldots, w_t$ are all of $\mathsf{polylog}(n)$ length and $t = O(\mathsf{polylog}(n))$. Therefore, the length of the whole string is at most $\mathsf{polylog}(n)$. This string can be independently verified by Alice and Bob individually.

Now, let H be the set of all possible computational histories for all possible input pairs (x, y). We construct our one-way memoryless protocol $\mathcal{P}'$ for f as follows: Alice enumerates through all computational histories in H that are compatible with her input x in lexicographically decreasing order of p. That is, if $h = (p, w_1, \ldots, w_t)$ and $h' = (p', w'_1, \ldots, w'_{t'})$ are two histories from H with $p < p'$ (in lexicographic order), then Alice enumerates h' before she enumerates h. In particular, for a query node, Alice enumerates all the paths that take its 1 child before she moves on to the paths that take its 0 child.

Upon receiving a computational history $h \in H$ from Alice, Bob checks to see if this computational history h is also compatible with his input y. If so, he outputs the label of the leaf node in $\mathcal{T}$ (the protocol tree of the original $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ protocol $\mathcal{P}$) specified by h, otherwise he just continues.

Clearly, this protocol $\mathcal{P}'$ is in $\mathsf{SPACE}_{\mathsf{LESS}}[\mathsf{polylog}(n)]$. All we need to do now is to prove that f is correctly computed by $\mathcal{P}'$. Consider an input pair (x, y) for f. Let p be root-to-leaf path in $\mathcal{T}$ followed by the original protocol $\mathcal{P}$ on input (x, y), and let h be one of the computational histories in H that are compatible with p as defined above. Let $h^* = (p^*, w_1^*, \ldots, w_t^*)$ be the computational history that is actually accepted by Alice and Bob in protocol $\mathcal{P}'$. We prove that $p = p^*$ by contradiction. Suppose $p \neq p^*$. Let v be the last node that is common to both path p and p^*. That is to say, path p^* deviates away from path p at node v.

- First, v cannot be a communication node, otherwise the owner of node v (either Alice or Bob of course) would reject p^* as incompatible with her input. She would see that the communication bit she would send at node v based on the local computation result on given input should be the one contained in p, not the one contained in p^*.

- So v must be a query node.

  - Suppose the 0-child of v is taken in p whilst the 1-child of v is taken in p^*. This is impossible for the following reason: since p is the correct path actually taken by the original protocol $\mathcal{P}$, the query answer obtained at v on the given input pair (x, y) should be 0. That means either Alice or Bob

should reject the purported 1-witness in h^* for the query performed at v as incompatible to their input;

– Suppose it is the other way around, the 1-child of v is take by p whilst 0-child of v is taken by p^*. Then $p > p^*$ in lexicographical order. This is also impossible because in that case Alice would enumerate h before h^*, and since h is a correct computational history that is compatible with the input pair (x, y), Alice and Bob would have accepted h before they even consider h^*.

That means $p = p^*$ and $\mathcal{P}'$ is correct for an arbitrarily chosen input pair (x, y).

Thus, we can conclude $f \in \mathsf{SPACE}_{\mathsf{LESS}}[\mathsf{polylog}(n)]$. □

Corollary 4.7. *The complexity class* $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ *contains exactly the set of functions* $f = \{f_n^{cc}\}_{n=1}^{\infty}$ *with quasi-polynomial overlay number, in other words,* $RO(f) = 2^{\mathsf{polylog}(n)}$

4.5 Overlay Lower Bounds

We showed before that rectangle overlays fully characterize both the memoryless complexity and the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ complexity class. A major application of this characterization is an intuitive combinatorial lower bound technique provided by rectangle overlays.

4.5.1 Combinatorial Lower Bound Technique

Theorem 4.8 (complete version of Theorem 1.4)**.** *Consider any boolean function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *and any product probability distribution* μ *on* $\{0,1\}^n \times \{0,1\}^n$*. Let* $\epsilon_\mu \stackrel{def}{=} \max_{\mathcal{R}} \mu(\mathcal{R})$*, where* $\mathcal{R}$ *ranges over all monochromatic rectangles in the communication matrix of* f*. Then,* $RO(f) \geq \frac{1}{\sqrt{\epsilon_\mu}}$*, and* $S(f) \geq \frac{1}{4}\log\left(\frac{1}{\epsilon_\mu}\right) - \frac{1}{2}$.

Proof. Let $\ell = RO(f)$. Then according to the definition of overlay number, there is a rectangle overlay $(\mathcal{R}_1, b_1)$, $(\mathcal{R}_2, b_2)$, …, $(\mathcal{R}_\ell, b_\ell)$ that computes f. And according to Theorem 4.5, $S(f) \geq \frac{\log(\ell)}{2} - \frac{1}{2}$.

Next we show the desired lower bound by constructing a sequence of rectangles $T_0, T_1, T_2, \ldots, T_\ell$ with the following properties:

1. $T_0 = \{0,1\}^n \times \{0,1\}^n$;
2. $T_\ell = \emptyset$;
3. $\mathcal{R}_j \cap T_i = \emptyset$ for all $1 \leq j \leq i \leq \ell$;
4. for every $i \in \{1, 2, \ldots, \ell\}$, we have $T_i \subseteq T_{i-1}$, and
5. $\mu(T_i) \geq \mu(T_{i-1}) - \sqrt{\epsilon_\mu}$.

The existence of such a sequence clearly implies that $\ell \geq \frac{1}{\sqrt{\epsilon_\mu}}$ and $S(f) \geq \frac{\log(\ell)}{2} - \frac{1}{2} \geq \frac{1}{4}\log\left(\frac{1}{\epsilon_\mu}\right) - \frac{1}{2}$.

Now, we construct the sequence $\{T_i\}$. First, we set $T_0 = \{0,1\}^n \times \{0,1\}^n$. Then, define $T_1, T_2, \ldots, T_\ell$ inductively. For every $i \in \{1, 2, \ldots, \ell\}$, define $\bar{R}_i \stackrel{\text{def}}{=} \mathcal{R}_i \cap T_{i-1}$. $\bar{R}_i$ is clearly a combinatorial rectangle, and by property 3 above, it is disjoint from all previous $\mathcal{R}_1, \ldots, \mathcal{R}_{i-1}$. Thus, the function defined by the rectangle overlay outputs color b_i for all $(x, y) \in \bar{R}_i$. In other words, $\bar{R}_i$ is monochromatic. Write $T_{i-1} = X_{i-1} \times Y_{i-1}$ and $\bar{R}_i = \bar{X}_i \times \bar{Y}_i$.

Since μ is a product distribution, we can write it as a product of two distributions $\mu = \mu_X \times \mu_Y$, where μ_X and μ_Y are both distributions on $\{0,1\}^n$. We have $\mu(\bar{R}_i) = \mu_X(\bar{X}_i) \cdot \mu_Y(\bar{Y}_i)$ and $\min\{\mu_X(\bar{X}_i), \mu_Y(\bar{Y}_i)\} \leq \sqrt{\mu(\bar{R}_i)} \leq \sqrt{\epsilon_\mu}$. Then we define

$$T_i \stackrel{\text{def}}{=} \begin{cases} (X_{i-1} \setminus \bar{X}_i) \times Y_{i-1} & \text{if } \mu_X(\bar{X}_i) \leq \mu_Y(\bar{Y}_i) \\ X_{i-1} \times (Y_{i-1} \setminus \bar{Y}_i) & \text{if } \mu_X(\bar{X}_i) > \mu_Y(\bar{Y}_i) \end{cases}$$

In other words, we construct T_i by cutting a piece away from T_{i-1} to ensure that T_i and R_i are disjoint. Thus, T_i possesses property 3 and property 4. In addition, we note that the piece we cut away from T_{i-1} has weight at most $\sqrt{\epsilon_\mu}$, therefore property 5 is also satisfied. Since per the definition of rectangle overlays, $\cup_{i=1}^{2^m} R_i = \{0,1\}^n \times \{0,1\}^n$, it holds that $T_\ell = \emptyset$. This completes the construction of the rectangle sequence $\{T_i\}$ and our proof is done. □

4.5.2 Applications of the Lower Bound Technique

In this section we apply the overlay lower bound technique presented above to several boolean functions and obtain tight lower bounds for their memoryless complexity.

Theorem 4.9 (see also Theorem 1.5)**.** $\frac{n}{4} - \frac{1}{2} \leq S(\mathrm{IP}) \leq n$*, and* $\mathrm{IP} \notin \mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$*. Here* IP *is the Inner-Product function of two* n*-dimensional vectors over* $GF(2)$*.*

Proof. For the upper bound part we observe that every boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ has a memoryless protocol with message length n: Alice simply sends her entire input to Bob in one single message and Bob computes the answer.

For the lower bound part we note that it is a well-known fact that every monochromatic rectangle in the communication matrix of IP has size at most 2^n (measured in the number of entries in the rectangle). See for example Example 1.25 in Kushilevitz and Nisan's book on communication complexity [27]. That is to say, if we choose the distribution μ in Theorem 4.8 to be the uniform distribution over $\{0,1\}^n \times \{0,1\}^n$, then we have $\epsilon_\mu \leq 2^{-n}$. The lower bound above follows. □

Theorem 4.10 (see also Theorem 1.5)**.** $\frac{1}{2}\sqrt{n} - \frac{1}{4}\log n - \frac{1}{2} \leq S(\mathrm{LNE}_{\sqrt{n},\sqrt{n}}) \leq \lceil\sqrt{n}\rceil + \lceil\log n\rceil$*, and* $\mathrm{LNE}_{\sqrt{n},\sqrt{n}} \notin \mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$*, where* $\mathrm{LNE}_{k,l}$ *is the List-Non-Equality function defined in Section 1.3.1.*

Proof. For the upper bound part, we have the following memoryless protocol: for $i \in \{1, 2, \ldots, \sqrt{n}\}$, in round i, Alice sends the index i and the i-th block of her input

x (consisting of $\sqrt{n}$ consecutive bits) to Bob, and Bob compares that to the i-th block of his input y.

For the lower bound part, we choose the distribution μ in Theorem 4.8 to be the uniform distribution on $\{0,1\}^n \times \{0,1\}^n$. We wish to show that the corresponding ϵ_μ in Theorem 4.8 is upper bounded by $\max\{k^2 \cdot 2^{-2l}, 2^{-2k}\}$ for $\mathrm{LNE}_{k,l}$.

For every 0-monochromatic rectangle $\mathcal{R}_0 = X_0 \times Y_0$ in the communication matrix of $\mathrm{LNE}_{k,l}$ (here $X_0, Y_0 \subseteq \{0,1\}^n$) if either X_0 or Y_0 is empty then $\mu(\mathcal{R}_0) = 0$. If neither of them is empty then we can take $x_0 \in X_0$ and $y_0 \in Y_0$. For every $y \in Y_0$, x_0 and y must be the same for one of the k blocks, therefore $|Y_0| \leq k \cdot 2^{n-l}$. Similarly we have $|X_0| \leq k \cdot 2^{n-l}$. Therefore, $\mu(\mathcal{R}_0) \leq \max\{0, \frac{(k \cdot 2^{n-l})^2}{2^{2n}}\} = k^2 \cdot 2^{-2l}$.

For every 1-monochromatic rectangle $\mathcal{R}_1 = X_1 \times Y_1$ in the communication matrix of $\mathrm{LNE}_{k,l}$ (here $X_1, Y_1 \subseteq \{0,1\}^n$), and every $i \in \{1, 2, \ldots, k\}$, we define

$$X_1^{(i)} = \{\bar{x} \in \{0,1\}^l \mid \exists x \in X_1 \text{ such that the } i\text{-th block in } x,\ x^{(i)} = \bar{x}\}$$

and

$$Y_1^{(i)} = \{\bar{y} \in \{0,1\}^l \mid \exists y \in Y_1 \text{ such that the } i\text{-th block in } y,\ y^{(i)} = \bar{y}\}$$

Clearly, for every $i \in \{1, 2, \ldots, k\}$, $X_1^{(i)} \cap Y_1^{(i)} = \emptyset$. Because otherwise we can choose $r \in X_1^{(i)} \cap Y_1^{(i)}$, then there are $x \in X_1$ and $y \in Y_1$ such that $x^{(i)} = r = y^{(i)}$. Then $\mathrm{LNE}_{k,l}(x, y) = 0$. This contradicts the fact that $\mathcal{R}_1$ is a 1 monochromatic rectangle. Therefore since $X_1^{(i)}, Y_1^{(i)} \subseteq \{0,1\}^l$, we have $\left|X_1^{(i)}\right| + \left|Y_1^{(i)}\right| \leq 2^l$, and this implies that

$$\left|X_1^{(i)}\right| \cdot \left|Y_1^{(i)}\right| = \frac{\left(\left|X_1^{(i)}\right| + \left|Y_1^{(i)}\right|\right)^2 - \left(\left|X_1^{(i)}\right| - \left|Y_1^{(i)}\right|\right)^2}{4} \leq \frac{\left(\left|X_1^{(i)}\right| + \left|Y_1^{(i)}\right|\right)^2}{4} \leq 2^{2(l-1)}$$

Therefore

$$|\mathcal{R}_1| = |X_1| \cdot |Y_1| \leq \prod_{i=1}^{k} \left|X_1^{(i)}\right| \cdot \prod_{i=1}^{k} \left|Y_1^{(i)}\right| = \prod_{i=1}^{k} \left|X_1^{(i)}\right| \cdot \left|Y_1^{(i)}\right| \leq 2^{2(n-k)}$$

That means $\mu(\mathcal{R}_1) \leq 2^{-2k}$.

Thus, we can conclude that $\epsilon_\mu \leq \max\{k^2 \cdot 2^{-2l}, 2^{-2k}\}$ for $\mathrm{LNE}_{k,l}$. Substitute in $k = l = \sqrt{n}$ in Theorem 4.8, we get the desired lower bound. □

Note that this lower bound, together with the fact that $\mathrm{LNE}_{\sqrt{n},\sqrt{n}} \in \Sigma_2^{cc} \cap \Pi_2^{cc}$ [28], gives a separation $\mathsf{P}^{\mathsf{NP}^{cc}} \subsetneq \Sigma_2^{cc} \cap \Pi_2^{cc}$. This separation was first obtained by Impagliazzo and Williams in 2010 [21]. With our new overlay characterization, the proof becomes more intuitive. The following lower bound for Gap-Hamming-Distance we present gives a somewhat stronger separation.

Theorem 4.11. *For every full function extension $f = \{f_n^{cc}\}_{n=1}^{\infty}$ of the Gap-Hamming-Distance function, $\frac{1-H(1/3)}{2} n - \frac{1}{4} \log n - \frac{1}{2} \leq S(f) \leq n$, and $f \notin \mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$, where $H(\cdot)$ is the binary entropy function, for $\lambda \in (0,1)$, $H(\lambda) \stackrel{def}{=} \lambda \cdot \log\left(\frac{1}{\lambda}\right) + (1-\lambda) \cdot \log \frac{1}{1-\lambda}$.*

For proving this theorem, we will need the following combinatorial lemma. The proof of this lemma is a little bit technical and we will delay it to the next section.

Lemma 4.12. *Suppose for $X, Y \subseteq \{0,1\}^n$, every $x \in X$ and every $y \in Y$ satisfy that $HD(x,y) > n/3$. Then $|X| \cdot |Y| \leq n \cdot 4^{H(1/3)n}$.*

Proof of Theorem 4.11. With the help of Lemma 4.12, we can show that all monochromatic rectangles of all full function extensions $f = \{f_n^{cc}\}_{n=1}^{\infty}$ of Gap-Hamming-Distance are very small.

Now, take an arbitrary 1-monochromatic rectangle $\mathcal{R}_1 = X_1 \times Y_1$ in the communication matrix of f_n^{cc}, where $X_1, Y_1 \subseteq \{0,1\}^n$. By definition of the Gap-Hamming-Distance function, it is clear that for every $x \in X_1$ and every $y \in Y_1$, we have $HD(x,y) > n/3$. Therefore, by Lemma 4.12, it holds that $|\mathcal{R}_1| = |X_1| \cdot |Y_1| \leq n \cdot 4^{H(1/3)n}$.

Next, we take an arbitrary 0-monochromatic rectangle $\mathcal{R}_0 = X_0 \times Y_0$ in the communication matrix of f_n^{cc}, where $X_0, Y_0 \subseteq \{0,1\}^n$. Again it is clear that for every $x \in X_0$ and every $y \in Y_0$, we have $HD(x,y) < 2n/3$. Now, for an n-bit string $x \in \{0,1\}^n$, we define the *complement* of x to be another n-bit string such that: for every $i \in \{1,2,\ldots,n\}$, if the i-th bit in x is 0, then the i-th bit in x's complement is defined to be 1; on the other hand, if the i-th bit in x is 1, then the i-th bit in x's complement is defined to be 0. We define $\bar{X_0} \stackrel{\text{def}}{=} \{\bar{x} \mid \text{the complement of } \bar{x} \text{ is in } X_0\}$. It is easy to see that for all $x, y \in \{0,1\}^n$, suppose the complement of x is $\bar{x}$, then $HD(x,y) + HD(\bar{x},y) = n$. Therefore, for all $(\bar{x},y) \in \bar{X_0} \times Y_0$, we have $HD(\bar{x},y) > n/3$. We apply Lemma 4.12 to the sets $\bar{X_0}$ and Y_0, we get $|\mathcal{R}_0| = |X_0| \cdot |Y_0| = \left|\bar{X_0}\right| \cdot |Y_0| \leq n \cdot 4^{H(1/3)n}$.

Finally, we apply Theorem 4.8 to f_n^{cc}, a full function extension of Gap-Hamming-Distance, and choose the distribution μ in the theorem to be the uniform distribution on $\{0,1\}^n$ then we get $\epsilon_\mu \leq n \cdot 4^{-(1-H(1/3))n}$, and the lower bound part in Theorem 4.11 follows. □

Together with the following theorem we present a stronger separation between $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ and Σ_2^{cc}/Π_2^{cc} than Impagliazzo and Williams [21], using the Gap-Hamming-Distance function.

Theorem 4.13. *There are functions $f = \{f_n^{cc}\}_{n=1}^{\infty}$ and $g = \{g_n^{cc}\}_{n=1}^{\infty}$, such that both are full function extensions of the Gap-Hamming-Distance function* GHD*, and $f \in \Sigma_2^{cc}$ and $g \in \Pi_2^{cc}$.*

We will use the following boolean function in the proof:

Problem 4.6. *the* Approximate-Majority *function,* AppMaj *is a partial function, for an n-bit input x.*

$$\text{AppMaj}(x) \stackrel{def}{=} \begin{cases} 0 & \text{if } |x|_1 \leq \frac{n}{3} \\ 1 & \text{if } |x|_1 \geq \frac{2}{3}n \\ * & \text{otherwise} \end{cases}$$

here $*$ means unspecified value.

Proof. We observe that once we find function g with the desired properties, we can simply set $f(x,y) \stackrel{\text{def}}{=} \neg g(\bar{x},y)$, in which $\bar{x}$ is the complement of x, that is, we flip every bit in x to get $\bar{x}$, replace every 0 with 1, and every 1 with 0. It is easy to see that f would also have the desired properties. Therefore, we can focus on constructing g, which amounts to constructing g_n^{cc} for every fixed input length n.

To construct the desired function g_n^{cc}, we consider a function closely related to Gap-Hamming-Distance, the AppMaj function defined above. It is easy to see that for two n-bit strings x and y, $\text{GHD}(x,y) = \text{AppMaj}(x \oplus y)$, where $\oplus$ denotes the exclusive-or operation.

Ajtai [3] proved that there is a polynomial-size depth-3 circuit family $\mathcal{C} = \{\mathcal{C}_n\}_{n=1}^{\infty}$ that computes a full function extension of AppMaj. Furthermore, $\mathcal{C}$'s output gate is an $\wedge$-gate; its middle layer consists of $\vee$-gates; and its bottom gates are $\wedge$-gates of fan-in $O(\log(n))$.

Now we replace every input gate of $\mathcal{C}_n$ which takes the i-th bit in the input with $x_i \oplus y_i$, this gives us circuit $\mathcal{C}_n^{cc}$ which computes GHD on two n-bit strings. In $\mathcal{C}_n^{cc}$, every $\wedge$-gate that is just above the bottom level takes $O(\log(n))$ inputs, therefore can be rewritten as a DNF formula. Furthermore, the top $\vee$-gate of every such DNF formula can be merged with one of the middle level $\vee$-gates. As a result, we get a polynomial size depth-3 circuit with an $\wedge$-gate as its output gate. According to Fact 2.6 $\mathcal{C}^{cc} = \{\mathcal{C}_n^{cc}\}_{n=1}^{\infty}$ is in Π_2^{cc}. We define $g_n^{cc} \stackrel{\text{def}}{=} \mathcal{C}_n^{cc}$ and we are done. □

4.5.3 Proof of Lemma 4.12

In this section we prove (for an independent proof of a similar statement, see [19]) the combinatorial lemma that is used above to prove the memoryless complexity lower bound of the Gap-Hamming-Distance function. To show this we need to use the well-known *vertex isoperimetric inequality*. Before getting to this inequality, we first introduce some new notation for this section: For integer $n \in \mathbb{Z}^+$ and real number $r \in [0,n]$, we define:

$$\binom{n}{\leq r} \stackrel{\text{def}}{=} \sum_{i=0}^{\lfloor r \rfloor} \binom{n}{i}$$

For integer $n \in \mathbb{Z}^+$, real number $r \geq 0$ and $X \subseteq \{0,1\}^n$, we define:

$$NB(X,r) \stackrel{\text{def}}{=} \{x' \in \{0,1\}^n \mid \exists x \in X \text{ such that } HD(x',x) \leq r\}$$

It is easy to see that:

- $NB(X,0) = X$;
- for any real number $r \geq 0$, $NB(NB(X,r),1) = NB(X,r+1)$;
- for any real number $r_1 \geq r_2 \geq 0$, $NB(X,r_1) \supseteq NB(X,r_2)$.

Now we can present the vertex isoperimetric inequality using the above notations:

Lemma 4.14 (Vertex Isoperimetric Inequality, see e.g. Theorem 5, Chapter 16, page 128 in Bollobás [12]). *Let $n \in \mathbb{Z}^+$, $A \subseteq \{0,1\}^n$, and let $0 \leq r \leq n-1$ be an integer. If $|A| \geq \binom{n}{\leq r}$, then $|NB(A,1)| \geq \binom{n}{\leq r+1}$.*

Next we prove the following auxiliary lemma using the vertex isoperimetric inequality:

Lemma 4.15 (Far Sets). *Let $X, Y \subseteq \{0,1\}^n$ be two non-empty sets. Let $d \in \mathbb{Z}^+$ be an integer such that for all $x \in X$ and $y \in Y$, $HD(x,y) \geq d$. Then there exists an integer $r \in \{0,1,2,\ldots,n-1\}$ such that $|X| \leq \binom{n}{\leq r+1}$ and $|Y| \leq \binom{n}{\leq n-(r+d)}$.*

Proof. It is easy to see that for a non-empty set $X \subseteq \{0,1\}^n$, there always exists an integer $r \in \{0,1,2,\ldots,n-1\}$ such that $\binom{n}{\leq r} \leq |X| \leq \binom{n}{\leq r+1}$. Next we prove that this r has the desired properties as specified in the Lemma. We have two cases to consider: either $d-1 \geq n-r$, or $d-1 < n-r$.

- case 1: $d-1 \geq n-r$. In this case, we apply the vertex isoperimetric inequality in Lemma 4.14 $n-r$ times to X, and get $|NB(X, n-r)| \geq \binom{n}{\leq n} = 2^n$. This means $NB(X, n-r) = \{0,1\}^n$. Since Y is non-empty, we can pick $y \in Y$. Clearly y is also in $NB(X, n-r)$. Based on the definition of $NB(X, n-r)$, there exists $x \in X$ such that $HD(x,y) \leq n-r \leq d-1$. This contradicts the premise of the theorem which says for all $x \in X$ and $y \in Y$, $HD(x,y) \geq d$! This means this case would never happen. It is always true that $d-1 < n-r$.

- case 2: $d-1 < n-r$. In this case, we apply the vertex isoperimetric inequality $d-1$ times to X, and get $|NB(X, d-1)| \geq \binom{n}{\leq r+d-1}$.

  We observe that $NB(X, d-1) \cap Y = \emptyset$. Otherwise, pick any $y \in Y \cap NB(X, d-1)$. Based on the definition of $NB(X, d-1)$, there exists $x \in X$ such that $HD(x,y) \leq d-1$, contradicting the premise of the theorem.

  Since $NB(X, d-1)$ and Y are disjoint, we conclude that $|NB(X, d-1)| + |Y| \leq 2^n$. From this we obtain

$$\begin{aligned}
|Y| &\leq 2^n - |NB(X, d-1)| \\
&\leq 2^n - \binom{n}{\leq r+d-1} \\
&= 2^n - \sum_{i=0}^{r+d-1} \binom{n}{i} \\
&= \sum_{i=r+d}^{n} \binom{n}{i} \\
&= \sum_{j=0}^{n-(r+d)} \binom{n}{j} = \binom{n}{\leq n-(r+d)}
\end{aligned}$$

This concludes the proof. □

Now we are ready to prove the main lemma of this section, Lemma 4.12.

Proof of Lemma 4.12. We apply Lemma 4.15 to sets X and Y with $d = \lceil n/3 \rceil$. (Note if either X or Y is empty, the conclusion of the theorem is already true.) We claim that there is an integer $r \in \{0, 1, 2, \ldots, n-1\}$ such that $|X| \leq \binom{n}{\leq r+1}$ and $|Y| \leq \binom{n}{\leq n-(r+\lceil n/3 \rceil)} \leq \binom{n}{\leq n-(r+n/3)}$.

For any real number $x \in [0, 1]$, define function $h(x)$ as follows:

$$h(x) \stackrel{\text{def}}{=} \begin{cases} H(x) & \text{if } 0 \leq x < \frac{1}{2} \\ 1 & \text{if } \frac{1}{2} \leq x \leq 1 \end{cases}$$

It is easy to see that the function h is continuous, monotonically increasing, and concave within its domain $[0, 1]$.

It is well-known (see for example Lemma 5.6 in [45]) that $\binom{n}{\leq xn} \leq 2^{h(x)n}$ for all $x \in [0, 1]$.

Note that $|X| \leq \binom{n}{\leq r+1} \leq n\binom{n}{\leq r}$. Write $r = \rho n$ (since $r \in \{0, 1, 2, \ldots, n-1\}$, $\rho \in [0, 1)$). Then

$$|X| \cdot |Y| \leq n \cdot 2^{h(\rho)n} \cdot 2^{h(1-(\rho+1/3))n} = n \cdot 2^{(h(\rho)+h(1-\rho-1/3))n}$$

Since h is concave, we can apply Jensen's inequality and obtain

$$n \cdot 2^{(h(\rho)+h(1-\rho-1/3))n} \leq n \cdot 2^{2h\left(\frac{\rho+1-\rho-1/3}{2}\right)n} = n \cdot 4^{h\left(\frac{1-1/3}{2}\right)n} = n \cdot 4^{H\left(\frac{1}{3}\right)n}$$

This finishes the proof of Lemma 4.12. □

4.6 Protocol Composition

In this section, we present a technique for compositing multiple memoryless protocols into one.

Theorem 4.16 (Theorem 1.21 restated)**.** *For positive integers n, c and functions $f_1, f_2, \ldots, f_c : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $h : \{0,1\}^c \to \{0,1\}$, suppose for each $i \in \{1, 2, \ldots, c\}$, f_i can be computed by a message length s_i memoryless protocol $\mathcal{P}_i$, then the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined as $f(x, y) \stackrel{def}{=} h(f_1(x, y), f_2(x, y), \ldots, f_c(x, y))$ is computable by a message length $\sum_{i=1}^{c} s_i$ memoryless protocol*

Because a memoryless protocol is, as their name suggests, "memoryless", it cannot just simulate the other memoryless protocols one-by-one and try to remember their outputs for later use. We instead run several memoryless protocols together "in parallel".

Proof of Theorem 4.16. For each $i \in \{1, 2, \ldots, c\}$, we denote the two defining functions of protocol $\mathcal{P}_i$ as A_i (for Alice) and B_i (for Bob) [1]. As discussed in Section 4.1.1

[1] See Definition 4.1 on page 61

(page 61), without loss of generality we assume that protocol $\mathcal{P}_i$ halts within 2^{s_i} rounds for every $i \in \{1, 2, \ldots, c\}$.

Given input pair $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, the "composite" protocol runs in $2^{s_1} \cdot 2^{s_2} \cdot \ldots \cdot 2^{s_c}$ rounds. In each round, Alice and Bob has a "working hypothesis" $(j_1, j_2, \ldots, j_c) \in \{1, 2, \ldots, 2^{s_1}\} \times \{1, 2, \ldots, 2^{s_2}\} \times \ldots \times \{1, 2, \ldots, 2^{s_c}\}$ that for each $i \in \{1, 2, \ldots, c\}$ the protocol $\mathcal{P}_i$ halts in round j_i when executed on input (x, y). They sort these different "working hypotheses" in lexicographic order and verify them one-by-one. To verify a "working hypothesis" $(j_1, j_2, \ldots, j_c)$, Alice sends one message $(\alpha_1, \alpha_2, \ldots, \alpha_c) \stackrel{\text{def}}{=} (A_1(j_1, x), A_2(j_2, x), \ldots, A_c(j_c, x))$ to Bob. Upon receiving this message, Bob computes $b_i \stackrel{\text{def}}{=} B_i(y, \alpha_i)$ for all $i \in \{1, 2, \ldots, c\}$. If all b_i's are in the set $\{0, 1\}$ then the current working hypothesis is accepted and Bob computes and outputs $h(b_1, b_2, \ldots, b_c)$, otherwise Alice and Bob proceed to the next "working hypothesis".

This memoryless protocol uses messages of length $\sum_{i=1}^{c} s_i$. We prove the correctness of the protocol by contradiction. The protocol makes a mistake only when Alice and Bob accept a wrong "working hypothesis". Suppose for each $i \in \{1, 2, \ldots, c\}$ protocol $\mathcal{P}_i$ actually halts in round j_i^* when executed on given input, whilst in the composite protocol Alice and Bob accept the "working hypothesis" $(j_1, j_2, \ldots, j_c)$. Find the smallest index i such that $j_i \neq j_i^*$, either $j_i < j_i^*$ or $j_i > j_i^*$.

- Suppose $j_i < j_i^*$, then since protocol $\mathcal{P}_i$ actually halts in round j_i^*, $B_i(y, A_i(j_i, x))$ outputs $\perp$, the composite protocol will never accept working hypothesis $(j_1, j_2, \ldots, j_c)$.
- Suppose $j_i > j_i^*$. Alice and Bob enumerate working hypothesis $(j_1^*, j_2^*, \ldots, j_c^*)$ before $(j_1, j_2, \ldots, j_c)$, and they will accept the former hypothesis before they even see the later one.

This contradiction concludes our proof. $\square$

Chapter 5

The Limited-Memory Communication Model

Our limited-memory communication model lies in between our memoryless model and our general model. An impressive thing about this model is that it is capable of simulating our general two-way model with moderate overhead. Its most interesting application so far is its connections to the communication complexity polynomial hierarchy, as already presented in Section 1.3.1 (page 26). These connections provide new characterizations for the complexity classes in the hierarchy. Another notable feature of this model is its similarity to bounded-width branching programs. We rely on tools provided by branching programs to prove the aforementioned connections between our limited-memory model and our general two-way model, and also the connections between our limited-memory model and the communication polynomial hierarchy.

In this chapter, we will first give formal definition to our limited-memory model and some techniques for constructing non-trivial protocols in this model. Then, we will explore the various connections this model has with other related computation models and complexity classes one by one.

5.1 Definitions

Definition 5.1 (The Limited-Memory Model)**.** *A* one-way limited-memory *protocol has two complexity parameters: the* message length *s, and the* number of memory states *in Bob's* permanent *memory w. The protocol is defined by two functions: $A : \mathbb{Z}^+ \times \{0,1\}^n \to \{0,1\}^s$, which defines Alice's behavior; and $B : \{0,1\}^n \times \{0,1\}^s \times \{1,2,\ldots,w\} \to \{0,1,\perp\} \times \{1,2,\ldots,w\}$, which defines Bob's behavior.*

Alice receives input $x \in \{0,1\}^n$ and Bob receives input $y \in \{0,1\}^n$ as usual. Bob's initial memory state is 1. The protocol proceeds in rounds. In round i ($i = 1,2,\ldots$), Alice computes a message $\alpha \stackrel{def}{=} A(x,i)$ and sends it to Bob. Bob, upon receiving this message, computes $(\beta, q') \stackrel{def}{=} B(y,\alpha,q)$, where q is Bob's current memory state, and q' is his designated new memory state. If $\beta \in \{0,1\}$, he outputs β and the protocol ends. If $\beta = \perp$, he switches to memory state q' without outputting anything, and the

protocol proceeds to round $i+1$. There is a round number limit: if a protocol does not halt within $w \cdot 2^s$ rounds, then it is forced to halt and output 0 by default.

$\mathsf{SPACE}_{\mathsf{LTD}}[s, w]$ *is the set of functions computable by a one-way limited-memory protocol with maximum message length s and w memory states for Bob.*

As discussed in Section 1.2.2 (page 19), we measure Bob's permanent memory in terms of the number of memory *states* instead of the number of memory *bits* in order to have a finer analysis of the computational power Bob's permanent memory has. First, with 5 memory states (fewer than what can be represented by 3 memory bits) the limited-memory model can already simulate the general two-way model with moderate overhead (see Theorem 5.7 on page 84). Number of memory bits is too coarse a measure for Bob's permanent memory.

Again, we mostly focus on the issue of whether a function is computable at all within a certain space bound (the combination of a certain message length bound and a certain number of memory states), without caring too much about the total communication cost.

A feature in the definition is that there must be a round number limit in the model. Without this limit the model will become too strong: every boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ will be computable with message length $\lceil \log(n+1) \rceil + 1$ and 2 memory states. This is analogous to the (easy) fact that every boolean function can be computed by a width-2 branching program with exponential length. The protocol can be partitioned into 2^n phases: in the i-th phase ($i \in \{1, 2, \ldots, 2^n\}$), Alice assumes Bob's input y to be the i-th element in the set $\{0,1\}^n$ (basically that is $(i-1)$ encoded in binary), computes the corresponding output $f(x, y)$. Then Bob verifies Alice's assumption and if it is correct he takes Alice's answer as the final output and halt. At the beginning of each phase Bob initializes his memory state to $\textsc{ok} \stackrel{\text{def}}{=} 1$. Each of the first n messages from Alice in this phase consists of an index $j \in \{1, 2, \ldots, n\}$ and the j-th bit in the input value y Alice assumes for this phase. Whenever Bob discovers an inconsistency between the input y Alice assumes and the input y he actually gets, he switches to state $\textsc{failed} \stackrel{\text{def}}{=} 2$. At the end of each phase, Alice sends index $(n+1)$ together with her assumed answer $f(x, y)$ to Bob, Bob takes Alice's answer as the final output if he is still in state $\textsc{ok}$, otherwise he initializes his memory state back to $\textsc{ok}$ and continues to the next phase.

It is easy to see that a memoryless protocol as defined in Definition 4.1 is merely a restricted limited-memory protocol with just 1 memory state. A careful reader may notice that we do not have a round number limit in the definition of our memoryless model (Definition 4.1). That is because as discussed in Section 4.1 (where Bob is memoryless), the model will not gain extra power even when the round number limit is left out. Any memoryless protocol with message length s that runs for more than 2^s rounds can always be optimized to a one that always halts within 2^s rounds.

5.1.1 Alternative Definition in Terms of Semi-Oblivious Space

Like the memoryless model, the limited-memory model can also be defined as a restricted variant of the general one-way model (see Definition 3.3 on page 49), in which

part of Bob's space is "oblivious" and can only be used to compress information received from Alice without mixing-in any information about his own input, whilst the "non-oblivious" part of Bob's space can be used freely by Bob.

Definition 5.2 (The One-Way Semi-Oblivious Model)**.** *The one-way semi-oblivious communication model with s bits of oblivious space and w bits of non-oblivious space is a special case of the general one-way model defined in Definition 3.3, in which Alice has a total number of $(s+w)$ bits of space, and Bob's* transition function *T_B is split into two: $T_B^o : \{0,1\}^s \times \{0,1\} \to \{0,1\}^s$ for updating the content of Bob's oblivious space, and $T_B^f : \{0,1\} \times \{0,1\}^n \times \{0,1\}^s \times \{0,1\}^w \to \{0,1,\perp\} \times \{0,1\}^w$ for making halting/output decision (here we only deal with the case where output length $m = 1$) and updating the content of Bob's non-oblivious space.*

We use M_A, M_B^o, and M_B^f to denote the content of Alice's local space, Bob's local oblivious *space and Bob's local* non-oblivious *space respectively. In each step of the protocol, Alice executes her transition function as usual (by definition of one-way protocols, Definition 3.3, the communication bit she receives from Bob in each step is always 0). Bob executes $M_B^{o\prime} = T_B^o(M_B^o, b)$ to update the content of his oblivious space, where M_B^o and $M_B^{o\prime}$ denote the old and new content of Bob's oblivious space before and after this step, and b is the communication bit received from Alice. In addition, Bob executes $(\beta, M_B^{f\prime}) = T_B^f(b, y, M_B^o, M_B^f)$ (here y is Bob's local input, M_B^f and $M_B^{f\prime}$ denote the old and new content of Bob's non-oblivious space before and after this step) to make halting/output decision and updating the content of his non-oblivious space: if $\beta \in \{0,1\}$, Bob outputs β and the protocol ends; if $\beta = \perp$, Bob updates the content of his non-oblivious space to $M_B^{f\prime}$ without outputting anything, and the protocol proceeds to the next step.*

Using arguments similar to the ones in Section 4.1.1 (page 62), we can show that this definition is practically equivalent to the definition of the limited-memory model (Definition 5.1), and it is important to have only one-way communication instead of two-way communication in these model definitions.

5.1.2 Randomized Variants

A *public-coin* randomized limited-memory protocol, just like its classical and memoryless counterparts (see Section 2.1.3 and Section 4.1), is defined as a probability distribution over a set of deterministic limited-memory protocols.

Definition 5.3. *In a public-coin randomized limited-memory protocol $\mathcal{P}$ with maximum message length s and w memory states, Alice and Bob start by drawing some random bits r from a random source. These random bits are shared between the two of them by default and are therefore* public-coins*. Then, they execute a public-coin specific* deterministic *limited-memory protocol $\mathcal{P}_r$ with maximum message length s and w memory states. We say that $\mathcal{P}$ satisfactorily approximates a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ if for every input pair $(x,y) \in \{0,1\}^n \times \{0,1\}^n$, we have $\Pr_r[\mathcal{P}_r(x,y) = f(x,y)] \geq \frac{2}{3}$. That is to say, for every possible input pair (x,y),*

the probability that the protocol chosen according to the random results of public-coin tosses makes a mistake should be smaller than $\frac{1}{3}$.

Furthermore, in some part of this work, we will also work with another type of public-coin randomized limited-memory protocols, in which Bob may end the protocol by saying "I do not know" without giving a definitive answer. But whenever he gives a definitive answer of either 0 or 1, the answer should never be wrong. In addition, Bob should not dodge the question by saying "I do not know" all the time.

Definition 5.4. *A public-coin randomized* zero-error *limited-memory protocol is like a "normal" public-coin randomized limited-memory protocol as defined above in Definition 5.3, except that in each round in the public-coin specific deterministic protocol, Bob may say "I do not know" and halt the protocol without giving a definitive answer. But, whenever Bob gives an answer in the set* $\{0,1\}$*, it should never be wrong. Furthermore, for every input pair* $(x,y) \in \{0,1\}^n \times \{0,1\}^n$

$$\Pr_r[\textit{when executing } \mathcal{P}_r(x,y)\textit{, Bob says "I do not know"}] \leq \frac{1}{2}$$

To distinguish with zero-error protocols, we use the term bounded-error *to describe the "normal" kind of public-coin randomized protocols as defined in Definition 5.3.*

5.2 Techniques for Constructing Efficient Protocols

In Section 1.3.3 we talked about two techniques for constructing non-trivial protocols for functions with certain special structure. Now we present them in detail.

Theorem 5.1 (Theorem 1.22 restated)**.** *For a communication problem* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$*, suppose* $f(x,y)$ *can be decomposed into* $g(h_1(x,y), h_2(x,y), \ldots, h_k(x,y))$*, where each* $h_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *depends on at most* l *bits in* x *(*$i \in \{1,2,\ldots,k\}$*), then* $f \in \mathsf{SPACE}_{\mathsf{LTD}}[l + k + \lceil \log k \rceil, 2]$.

Proof. Suppose that the function $f(x,y)$ can be decomposed as specified. Then, we construct a protocol $\mathcal{P}$ with message length $l + k + \lceil \log k \rceil$ and 2 memory states for Bob as follows:

The protocol $\mathcal{P}$ is divided in 2^k phases. In the i-th phase, Alice and Bob have the "working hypothesis" that the outputs of the functions $h_1(x,y), h_2(x,y), \ldots, h_k(x,y)$ are simply the bits in the binary encoding of $(i-1)$, denoted as $z \in \{0,1\}^k$. That is to say, for every $j \in \{1,2,\ldots,k\}$, $h_j(x,y) = z_j$, where z_j is the j-th bit in z. They try to cooperatively verify this "working hypothesis". If they find the "working hypothesis" to be true, Bob can give the final output $g(z_1, z_2, \ldots, z_k)$ and halt.

At the beginning of i-th phase ($i \in \{1,2,\ldots,2^k\}$) Bob always initializes his memory state to $\textsc{ok} \stackrel{\text{def}}{=} 1$. They verify their "working hypothesis" for the current phase in k rounds. In the j-th round ($j \in \{1,2,\ldots,k\}$), Alice sends Bob 3 things in her message: phase index i, round index j, and the input bits in x that h_j actually depends

on. Based on the definition of i and j, and the fact that each h_j depends on at most l bits in x, the message length does not exceed $l + k + \lceil \log k \rceil$ as specified. Bob, upon receiving the message, computes $h_j(x, y)$ and compares it with z_j, the j-th bit in the binary encoding of $(i-1)$. If any inconsistency is found, Bob switches to memory state FAILED $\stackrel{\text{def}}{=} 2$ to denote that the working hypothesis of the current phase is invalidated, otherwise he continues to the next round remaining in the current state. After the verification for round k is done, Bob checks to see if he is still in memory state OK, and if so, he outputs $g(z_1, z_2, \ldots, z_k)$ and halts, otherwise he initializes his memory state back to OK and continues to the next phase. □

Using a very similar strategy, we can also prove the following protocol construction technique for bounded size formulas. This result was inspired by Valiant [43].

Theorem 5.2 (Theorem 1.25 restated)**.** *If a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ can be computed by a boolean formula of size S and fan-in 2 ($S \geq 4$), then $f \in \mathsf{SPACE}_{\mathsf{LTD}}[4\left\lceil\sqrt{S}\right\rceil + \left\lceil\frac{\log S}{2}\right\rceil + 4, 2]$*

To prove this theorem we need the following lemma concerning how to break up a bounded size formula into more manageable pieces. The proof of this lemma follows the "1/3-2/3 principle" and is presented in full in Section 5.2.1.

Lemma 5.3. *Given a directed binary tree $\mathcal{T}$ of size S ($S \geq 4$), with edges leading from leaves to root, we have a way of deleting at most $\left\lceil\sqrt{S}\right\rceil + 1$ edges of $\mathcal{T}$, such that the remaining graph is a forest consisting of at most $\left\lceil\sqrt{S}\right\rceil + 2$ trees, and each tree is of size between $\left\lfloor\sqrt{S}\right\rfloor$ (inclusive) and $3\left\lceil\sqrt{S}\right\rceil$ (inclusive).*

Proof of Theorem 5.2. Let the boolean formula of size S that can compute function f be $\mathcal{C}$, according to Lemma 5.3, we can delete a set of edges E ($|E| \leq \left\lceil\sqrt{S}\right\rceil + 1$), such that the remaining is a set of sub-formulas $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$, $m \leq \left\lceil\sqrt{S}\right\rceil + 2$, and for every $i \in \{1, 2, \ldots, m\}$, $\left\lfloor\sqrt{S}\right\rfloor \leq |\mathcal{C}_i| \leq 3\left\lceil\sqrt{S}\right\rceil$ (here we ignore the input gates corresponding to the deleted edges).

Now, we can present a limited-memory protocol $\mathcal{P}$ with the desired complexity parameter in terms of these sub-formulas. The protocol is divided into $2^{|E|+1}$ phases. Alice and Bob enumerate all the elements $(v_E, o) \in \{0,1\}^{|E|} \times \{0,1\}$ one by one. In each phase they take one element (v_E, o) from $\{0,1\}^{|E|} \times \{0,1\}$ as their "working hypothesis": they assume the bits in v_E are the bits computed by $\mathcal{C}$ on the edges in E and o is the output of $\mathcal{C}$. They verify this hypothesis and if the verification passes, they output o as the final answer.

In the beginning of each phase, Bob always initializes his memory state to OK $\stackrel{\text{def}}{=} 1$. Then they verify their current working hypothesis in m rounds. In the i-th round of each phase ($i \in \{1, 2, \ldots, m\}$), Alice sends the following information to Bob in her message: the current working hypothesis (v_E, o), the round number i, and the input bits in x that $\mathcal{C}_i$ depends on. Then, Bob takes all input bits to $\mathcal{C}_i$, which must be

among the bits in x, y and v_E, computes the output of $\mathcal{C}_i$, and compares his result with the assumed output value in the current working hypothesis (the output gate of $\mathcal{C}_i$ should either be the output gate of $\mathcal{C}$, or corresponds to one of the edges in E). If any inconsistency is found, Bob switches to memory state FAILED $\stackrel{\text{def}}{=} 2$, otherwise he remains in the current state and continues to the next round. After all the m sub-formulas are checked, if Bob is still in the OK memory state, then he announces the current working hypothesis (v_E, o) to be true and takes o as the final answer. Otherwise, he initializes his memory state back to OK and they proceed to the next phase with the next working hypothesis.

We note that each sub-formula $\mathcal{C}_i$ has size at most $3\left\lceil\sqrt{S}\right\rceil$. In particular it has at most $3\left\lceil\sqrt{S}\right\rceil$ input gates, which means it will depend on at most $3\left\lceil\sqrt{S}\right\rceil$ bits from x. Then, counting in $|E| \leq \left\lceil\sqrt{S}\right\rceil + 1$ and $m \leq \left\lceil\sqrt{S}\right\rceil + 2$, we conclude that in protocol $\mathcal{P}$, each of Alice's messages can be encoded into bit strings of length at most $4\left\lceil\sqrt{S}\right\rceil + \left\lceil\frac{\log S}{2}\right\rceil + 4$ and Bob uses only two memory states $\{\text{OK}, \text{FAILED}\}$. □

5.2.1 Proof of Lemma 5.3

For completeness we provide the proof of Lemma 5.3.

To prove Lemma 5.3, we need yet another lemma, which we prove using the "1/3-2/3 principle". We denote the size of a tree $\mathcal{T}$ (the number of nodes in $\mathcal{T}$) as $|\mathcal{T}|$. And for a node v in tree $\mathcal{T}$, we denote the subtree of $\mathcal{T}$ rooted at v as $\mathcal{T}_v$.

Lemma 5.4. *For any integer $S \geq 4$, and any binary tree $\mathcal{T}$ such that $|\mathcal{T}| > 2\left\lceil\sqrt{S}\right\rceil$, there is a node v in $\mathcal{T}$ such that $\left\lfloor\sqrt{S}\right\rfloor \leq |\mathcal{T}_v| \leq 2\left\lceil\sqrt{S}\right\rceil$.*

Proof. We prove this by contradiction. Suppose there is no such node v. Now let us set v_0 to be the root of the tree, suppose the two children of v_0 are l_0 and r_0, without loss of generality, we can assume that $|\mathcal{T}_{l_0}| > |\mathcal{T}_{r_0}|$, then we set $v_1 \stackrel{\text{def}}{=} l_0$. Again we look at the two children of v_1 and choose v_2 similarly. We do this repeatedly until we have reached a leaf node. (This process will clearly end in finite steps because we have only S nodes in the tree.)

Suppose the node sequence we get is $v_0, v_1, \ldots, v_k$. The size of $|\mathcal{T}_{v_0}| > 2\left\lceil\sqrt{S}\right\rceil$, whilst $|\mathcal{T}_{v_k}| = 1 < \left\lfloor\sqrt{S}\right\rfloor$ (given $S \geq 4$). Based on our original assumption, there is no $i \in \{0, 1, \ldots, k\}$ such that $\left\lfloor\sqrt{S}\right\rfloor \leq |\mathcal{T}_{v_i}| \leq 2\left\lceil\sqrt{S}\right\rceil$. And $|\mathcal{T}_{v_i}|$ must be a strictly monotonically decreasing function in i. Therefore, there must be a $i \in \{0, 1, \ldots, k-1\}$ such that $|\mathcal{T}_{v_i}| > 2\left\lceil\sqrt{S}\right\rceil$, whilst $\left|\mathcal{T}_{v_{i+1}}\right| < \left\lfloor\sqrt{S}\right\rfloor$. But based on the way we choose v_i's, we should have $|\mathcal{T}_{v_i}| \leq 1 + 2\left|\mathcal{T}_{v_{i+1}}\right| < 1 + 2\left\lfloor\sqrt{S}\right\rfloor$. This is impossible! That concludes our proof. □

Proof of Lemma 5.3. Now suppose we maintain a forest $\mathcal{F}$ which can always be obtained by deleting several edges in $\mathcal{T}$ and collecting the resulting subtrees. Originally,

$\mathcal{F}$ has just one tree, that is $\mathcal{T}$ itself. Each time, we pick one tree $\mathcal{T}^*$ in $\mathcal{F}$ with size strictly bigger than $3\left\lceil\sqrt{S}\right\rceil$, find a subtree $\mathcal{T}'$ of it with size between $\left\lfloor\sqrt{S}\right\rfloor$ (inclusive) and $2\left\lceil\sqrt{S}\right\rceil$ (inclusive), and delete the edge connecting the root node of $\mathcal{T}'$ to its parent. (The existence of $\mathcal{T}'$ is guaranteed by the above lemma.) We do this repeatedly until no such tree $\mathcal{T}^*$ exists.

Now, we note that there will never be a tree in $\mathcal{F}$ with size strictly smaller than $\left\lfloor\sqrt{S}\right\rfloor$. Because originally $\mathcal{T}$ has size greater than $\left\lfloor\sqrt{S}\right\rfloor$. And each time we cut up a tree $\mathcal{T}^*$, it is easy to see that neither $\mathcal{T}'$ nor $\mathcal{T}^* \setminus \mathcal{T}'$ has size strictly smaller than $\left\lfloor\sqrt{S}\right\rfloor$.

Secondly we note that this process will end after at most $\left\lceil\sqrt{S}\right\rceil + 1$ steps. Because each time we cut up a tree in $\mathcal{F}$, the size of $\mathcal{F}$ increment by 1. And the size of $\mathcal{F}$ can not exceed $\left\lceil\sqrt{S}\right\rceil + 2$, because otherwise the total size of the trees in $\mathcal{F}$ would exceed $(\sqrt{S}-1)(\sqrt{S}+2) \geq S$, which is impossible.

All in all, that means we will end up with a forest $\mathcal{F}$ in which all the trees have the desired size. □

5.3 Comparison with Bounded-Width Branching Programs

The reader may have already noticed the similarity between our limited-memory protocols and bounded-width oblivious branching programs with local preprocessing as defined in Definition 2.9 and Definition 2.10. In fact it is not hard to prove that a width-w oblivious branching program with local preprocessing can always be simulated by a limited-memory communication protocol with w memory states for Bob:

Fact 5.5. *If a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ can be computed with a width-w and length l oblivious branching program with local preprocessing, then it can be computed with a limited-memory communication protocol with message length $\lceil \log l \rceil + 1$ and w memory states for Bob.*

Proof. The protocol simulates the branching program layer by layer. Bob uses his w memory states to keep track of the index of the traversed node in the current layer. For each layer, if the underlying input function is a preprocessing function on x, Alice sends the layer number together with the output of the preprocessing function to Bob; if the underlying input function is a preprocessing function on y, then Alice sends the layer number with a placeholder bit 0 to Bob, just to trigger Bob to move forward one layer by himself. □

Interestingly, the simulation in the reverse direction is not always possible.

Theorem 5.6 (Theorem 1.24 restated)**.** *The shortest width-2 branching program that correctly computes* IP_3 [1] *has length* $\Omega\left(\left(\frac{3}{2\sqrt{2}}\right)^n\right)$*, whilst there is a limited-memory protocol with message length* $O(\sqrt{n})$ *and* 2 *memory states that correctly computes* IP_3*, that is,* $\mathrm{IP}_3 \in \mathsf{SPACE}_{\mathsf{LTD}}[O(\sqrt{n}), 2]$.

Proof. The width-2 branching program length lower bound was proved by Shearer [40], using the combinatorial tool called striped cube introduced by Borodin, Dolev, Fich and Paul [14].

We prove the upper bound in our limited-memory model with the help of Theorem 5.1. We note that we can cut the input bits of each of x and y into $\lceil\sqrt{n}\rceil$ consecutive blocks, each block of length at most $\lceil\sqrt{n}\rceil$. Then for each $i \in \{1, 2, \ldots, \lceil\sqrt{n}\rceil\}$, we define the $h_i(x, y)$ to be the $GF(3)$ inner product of the i-th block in x and the i-th block in y. We further define g to be the sum of $h_1, h_2, \ldots, h_{\lceil\sqrt{n}\rceil}$ over $GF(3)$. We apply Theorem 5.1 and conclude that IP_3 has a 2 memory state limited-memory protocol with message length $O(\sqrt{n})$. □

5.4 Comparison with the General Two-way Model

In Section 4.2, we showed that the memoryless model is not as weak as it seems to be according to its seemingly restricted definition. With a little bit more space the memoryless model can beat the (strong) general two-way model in terms of the number of boolean functions that can be computed in each of these two models. In this section we will show that actually if you give the limited-memory model just 5 memory states (instead of just 1 memory state in the case of the memoryless model), then it will be able to completely simulate the general two-way model with moderate overhead. In particular, we have [2]:

Theorem 5.7 (Theorem 1.14 restated)**.** *For any* $s(n) > \log n$*, if a function family* $\{f_n^{cc}\}_{n=1}^{\infty}$ *can be computed by a space* $s(n)$ *general two-way communication protocol, then it is also computable by a limited-memory protocol with message length* $O(s(n)^2)$ *and* 5 *memory states.*

Proof. Suppose the function f can be computed by a space $s(n)$ general two-way protocol, then according to Theorem 3.1, it can also be computed by a space $O(s(n))$ Turing machine with local preprocessing. Furthermore according to Theorem 2.5, this function f can also be computed by a depth $O((s(n))^2)$ circuit family with local preprocessing.

Next we take into account our generalized version of Barrington's theorem Theorem 2.9, the function f can also be computed with a length $2^{O((s(n))^2)}$ width 5 oblivious branching program with local preprocessing.

Finally we note that according to Fact 5.5, such a branching program can always be simulated by a limited-memory protocol with message length $O((s(n))^2)$ and 5 memory states for Bob. This concludes the proof. □

[1] See Problem 1.2 on page 25 for definition.

[2] Inspired by discussion with Buhrman and Speelman [1].

5.5 Connections to the Communication Complexity Polynomial Hierarchy

In Section 1.3.1, we introduced several connections between our limited-memory model and the communication complexity polynomial hierarchy. The connection between our memoryless model and the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ complexity class that we examined in detail in Section 4.4 is just one of them. Now we examine the other two in the following sections.

5.5.1 3 Memory States and $\mathsf{PH}^{\mathsf{cc}}$

Theorem 5.8 (Theorem 1.6 restated)**.** *For any constant k, if a function family $\{f_n^{cc}\}_{n=1}^{\infty}$ is in Σ_k^{cc}, then $\{f_n^{cc}\}_{n=1}^{\infty}$ can also be computed in our limited-memory model with message length $\mathsf{polylog}(n)$ and 3 memory states. In other words, $\mathsf{PH}^{\mathsf{cc}} \subseteq \mathsf{SPACE}_{\mathsf{LTD}}[\mathsf{polylog}(n), 3]$.*

The proof of this theorem follows a storyline similar to Allender and Hertrampf's work on circuit reduction [4]. The idea is that we first use Razborov and Smolensky's approximation [38, 41] to give a public-coin bounded-error randomized protocol for any function in $\mathsf{PH}^{\mathsf{cc}}$. This protocol uses messages of length $\mathsf{polylog}(n)$ and has 2 memory states for Bob. Then we borrow the "error-indicator" idea from Braverman [15] to make the protocol zero-error. We note that this technique of converting a public-coin bounded-error randomized limited-memory protocol into a zero-error one is specific to our proof for $\mathsf{PH}^{\mathsf{cc}}$, we are not sure how to do this bounded-error to zero-error conversion in the general case. Finally, we use Newman's technique [31] to reduce the total number of public-coin tosses needed and derandomize the zero-error protocol with one more memory state for Bob. That gives us the protocol in $\mathsf{SPACE}_{\mathsf{LTD}}[\mathsf{polylog}n, 3]$ we need.

The Bounded-error Protocol

First let us look at Razborov and Smolensky's idea about how to approximate constant depth circuits with polynomials over $GF(2)$. Since it is quite straightforward to compute polynomials over $GF(2)$ with our 2-state limited-memory protocols, and the complexity class $\mathsf{PH}^{\mathsf{cc}}$ contains only constant-depth circuits augmented with local preprocessing (see Fact 2.6), this gives us a way to approximate functions in $\mathsf{PH}^{\mathsf{cc}}$ with 2-state randomized limited-memory protocols.

Lemma 5.9 (Razborov and Smolensky [38, 41])**.** *Suppose $\mathcal{C}$ is a depth d size S circuit on the basis $\{\vee, \oplus, \neg\}$ ($\oplus$ is exclusive-or gate) with unbounded fan-in, and it takes an m-bit string $x \in \{0,1\}^m$ as input. Then, we can choose a random string r uniformly from $\{0,1\}^{S^2 \log(3S)}$ and assign a polynomial $p_{r,g}$ over $GF(2)$ to each gate g in the circuit based on the value of r, such that:* [3]

[3] Here we use the additive identity 0 in $GF(2)$ to represent the boolean value false, and we use the multiplicative identity 1 to represent the boolean value true.

- *for each value* $r \in \{0,1\}^{S^2 \log(3S)}$ *and each gate* g *in circuit* $\mathcal{C}$*, the degree of the corresponding polynomial* $p_{r,g}$ *is at most* $(\log(3S))^d$*;*

- *for every input value* $x \in \{0,1\}^m$*, the probability that all the polynomials correctly compute the output values of the corresponding gates on input* x *is at least* $\frac{2}{3}$*, or in other words,*

$$\forall x \in \{0,1\}^m \quad \Pr_r[\forall g \in \mathcal{C} \quad p_{r,g}(x) = g(x)] \geq \frac{2}{3}$$

  here, $g(x)$ *is the output of gate* g *given input* x*;*

- *if* g *is an input gate with input* x_i*, that is, it takes the* i*-th bit in* x *as its output (*$i \in \{1, 2, \ldots, m\}$*), then for every* $r \in \{0,1\}^{S^2 \log(3S)}$*,* $p_{r,g} = x_i$*;*

- *if* g *is a* $\neg$*-gate with* c *as its only child, then for every* $r \in \{0,1\}^{S^2 \log(3S)}$*,* $p_{r,g} = p_{r,c} + 1$*;*

- *if* g *is an* $\oplus$*-gate with children* $c_1, c_2, \ldots, c_t$*, then for every* $r \in \{0,1\}^{S^2 \log(3S)}$*, we have* $p_{r,g} = \sum_{i=1}^{t} p_{r,c_i}$*;*

- *if* g *is an* $\vee$*-gate with children* $c_1, c_2, \ldots, c_t$ *and* $p_{r,c_1}(x) = \cdots = p_{r,c_t}(x) = 0$ *for a particular* $x \in \{0,1\}^m$ *and* $r \in \{0,1\}^{S^2 \log(3S)}$*, then* $p_{r,g}(x) = 0$*, too.*

Note that in the above lemma the circuit $\mathcal{C}$ can only have $\vee$, $\oplus$ and $\neg$ gates, but no $\wedge$ gates. Since we can always replace every $\wedge$ gate with $\vee$ and $\neg$ gates using De Morgan's laws, and convert a depth d size S circuit $\mathcal{C}_1$ with $\wedge$ gates into a depth $3d$ size $2S$ circuit $\mathcal{C}_2$ without $\wedge$ gates, so this is not a real problem.

Theorem 5.10. *Let* $f = \{f_n^{cc}\}_{n=1}^{\infty} \in \mathsf{PH}^{cc}$*. There exists a public-coin bounded-error randomized limited-memory protocol family with* $\mathsf{polylog}(n)$ *message length and* 2 *memory states that can satisfactorily approximate* f

Proof of Theorem 5.10. Let $f = \{f_n^{cc}\}_{n=1}^{\infty} \in \mathsf{PH}^{cc}$. By Fact 2.6, there is a constant-depth formula family $\mathcal{C}^{cc} = \{\mathcal{C}_n^{cc}\}_{n=1}^{\infty}$ with local preprocessing that can compute f, and this formula family has quasi-polynomial (that is, $2^{\mathsf{polylog}(n)}$) size.

We first apply De Morgan's laws and transform $\mathcal{C}^{cc}$ into another formula family $\mathcal{C}^{cc*}$ without $\wedge$ gates maintaining constant depth and quasi-polynomial size.

Then we take the formula for input length n, ${\mathcal{C}^{cc*}}_n$, from $\mathcal{C}^{cc*}$. Suppose the preprocessing function used by ${\mathcal{C}^{cc*}}_n$ are $p(x)$ and $q(y)$. According to Lemma 5.9, there is a randomized polynomial p_{r,g_o} (based on randomness r) over $GF(2)$ with degree $\mathsf{polylog}(n)$ that can approximate the output gate g_o of ${\mathcal{C}^{cc*}}_n$. Such a polynomial p has at most $2^{\mathsf{polylog}(n)}$ terms when expanded. Therefore Alice and Bob can use a public-coin bounded-error randomized limited-memory protocol with $\mathsf{polylog}(n)$ message length and 2 memory states to compute this polynomial p. First they use their public-coin randomness to sample r uniformly from $\{0,1\}^{S^2 \log(3S)}$, then they compute p_{r,g_o} on inputs $p(x)$ and $q(y)$ term-by-term. In each message, Alice sends Bob the index of a

term and the product of all the factors in that term that are bits from $p(x)$. Bob uses his 2-state memory $\{0,1\}$ to keep the partial sum of already processed terms, and upon receiving each of Alice's message, he computes the product in the same term that are bits from $q(y)$, multiplies it with Alice's product, and add that to his partial sum. □

The Zero-error Protocol

We now want to make the above protocol zero-error. Since the circuit we are dealing with has only $\vee$, $\oplus$ and $\neg$ gates, according to the last four bullet points of Lemma 5.9 we can conclude that in this approximation, errors can only be introduced at $\vee$-gates, and only if there is some $\vee$-gate g with children $c_1, c_2, \ldots, c_t$ such that for some $x \in \{0,1\}^m$ and $r \in \{0,1\}^{S^2 \log(3S)}$, there exists $i \in \{1,2,\ldots,t\}$, $p_{r,c_i}(x) = 1$ but $p_{r,g}(x) = 0$. Alternatively, an error is introduced at such an $\vee$-gate g only if the following "error indicator"

$$\mathcal{E}_{r,g}(x) \stackrel{\text{def}}{=} \bigvee_{i=1}^{t} (\neg p_{r,g}(x)) \wedge p_{r,c_i}(x)$$

evaluates to 1. We can define an "error indicator" for the whole circuit as follows:

$$\mathcal{E}_r(x) \stackrel{\text{def}}{=} \bigvee_{\substack{\vee\text{-gates } g \\ c \text{ is a child of } g}} (\neg p_{r,g}(x)) \wedge p_{r,c}(x)$$

With this error indicator, we are now ready to present our zero-error protocol.

Theorem 5.11. *Let $f = \{f_n^{cc}\}_{n=1}^{\infty} \in \mathsf{PH}^{cc}$. There exists a public-coin zero-error randomized limited-memory protocol family with $\mathsf{polylog}(n)$ message length and 2 memory states that can satisfactorily approximate f*

Proof. The zero-error protocol works as follows: first Alice and Bob use their public-coin randomness to sample r uniformly from $\{0,1\}^{S^2 \log(3S)}$, then their r-specific deterministic protocol proceeds in two phases: an "error checking" phase, in which Alice and Bob evaluate the error indicator $\mathcal{E}_r$, and after than an "conclusion" phase, in which Alice and Bob evaluates the polynomial that corresponds to the output gate of the circuit. Whenever the error indicator evaluates to 1, Bob says "I do not know" and halts the protocol. Therefore whenever they proceed to the second phase, the polynomial of the output gate will give the correct answer.

We note that every $(\neg p_{r,g}(x)) \wedge p_{r,c}(x)$ part in the error indicator, where g is an $\vee$-gate, and c is a child of g, can be rewritten as a polynomial of degree $\mathsf{polylog}(n)$. Furthermore, the size of the circuit is quasi-polynomial ($2^{\mathsf{polylog}(n)}$), so the number of such (g,c) pairs we need to enumerate is at most quasi-polynomial ($2^{\mathsf{polylog}(n)}$). Therefore in the error checking phase, Alice and Bob can enumerate over all such polynomials and evaluate each of them term-by-term using just $\mathsf{polylog}(n)$ message length and 2 memory states for Bob, like how we evaluate degree $\mathsf{polylog}(n)$ polynomials in the bounded-error protocol in the proof of Theorem 5.10.

In the "conclusion" phase, we again evaluate the polynomial corresponding to the output gate of the circuit like we did it in the proof of Theorem 5.10.

To establish the protocol described above as a proper zero-error protocol, we need to also show that Bob does not say "I do not know" too often. We observe that according to Lemma 5.9, the probability that an error is introduced somewhere is at most $\frac{1}{3}$, and the error indicator will evaluate to 1 only if some error is introduced somewhere, therefore the probability that Bob says "I do not know" is at most $\frac{1}{3}$. This concludes our proof. □

Derandomization of Zero-Error Protocols

In this section we show a generic mechanism for derandomizing zero-error limited-memory protocols.

Theorem 5.12 (Derandomization of Zero-Error Protocols)**.** *Suppose there is a zero-error protocol computing $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ with maximum message length s and w memory states. Then there is a deterministic protocol computing f with maximum message length $s + \lceil \log(3n) \rceil$ and $w + 1$ memory states.*

The main idea is that if we can find a small set of random strings R such that for every input pair (x, y), there is at least one random string $r \in R$ for which the original zero-error protocol will give a definitive answer for (x, y), then we can construct a deterministic protocol by simply enumerating all random strings in R and simulate the original zero-error protocol for each random string in turn. We have an additional memory state for Bob called "I do not know", which Bob switches to whenever the original protocol tries to say "I do not know". And whenever the original zero-error protocol tries to give a definitive answer, Bob takes it as the final answer.

The existence of such a small set of random strings R can be proven using techniques first introduced by Newman [31]; cf. Theorem 3.14 of [27].

Proof of Theorem 5.12. Suppose the original zero-error protocol is $\mathcal{P}_0$. We denote the output of this protocol on input (x, y) and random string r as $\mathcal{P}_0(x, y, r)$. By definition $\mathcal{P}_0(x, y, r) \in \{0, 1, \text{"I do not know"}\}$. We say a random string r is *good for input pair* (x, y) if $\mathcal{P}_0(x, y, r) \neq$ "I do not know". By definition, if we draw r according to the distribution take by $\mathcal{P}_0$, then for each input pair (x, y), the probability that r is not good for (x, y) is at most $\frac{1}{2}$. That means for each input pair (x, y), if we draw $3n$ random strings $r_1, r_2, \ldots, r_{3n}$ independently, all according to the distribution in $\mathcal{P}_0$, then the probability that none of them is good for (x, y) drops to 2^{-3n}. Therefore since $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, and $|\{0,1\}^n \times \{0,1\}^n| = 2^{2n}$, by a union bound we have

$$\Pr[\exists (x, y) \in \{0,1\}^n \times \{0,1\}^n \quad \text{none of } r_1, r_2, \ldots, r_{3n} \text{ is good for } (x, y)] \leq 2^{-n} < 1$$

That is to say, there exists a fixed choice of strings $R = \{r_1, r_2, \ldots, r_{3n}\}$ such that for every (x, y) there is at least one r_i ($i \in \{1, 2, \ldots, 3n\}$) that is good for (x, y).

With this set of random strings R, our deterministic protocol $\mathcal{P}$ works as follows: we use w memory states to simulate the w memory states in the original zero-error

protocol $\mathcal{P}_0$, and we give Bob one additional memory state called "I do not know". Given input pair (x, y), our protocol $\mathcal{P}$ proceeds in $3n$ phases. In the i-th phase ($i \in \{1, 2, \ldots, 3n\}$), $\mathcal{P}$ initializes Bob memory state to the same state as $\mathcal{P}_0$, and then $\mathcal{P}$ simulates the execution of $\mathcal{P}_0(x, y, r_i)$. Whenever $\mathcal{P}_0(x, y, r_i)$ tries to say "I do not know", $\mathcal{P}$ switches Bob's state to the "I do not know" state and makes Bob to stay in that state until the phase ends. Whenever $\mathcal{P}_0(x, y, r_i)$ tries to give a definitive answer, $\mathcal{P}$ takes that as the final answer. Each message Alice sends to Bob in $\mathcal{P}$ is the original message in $\mathcal{P}_0$ plus the round number i. Since $|R| \leq 3n$, the maximum message length in $\mathcal{P}$ will not exceed $s + \lceil \log n \rceil$. That concludes the proof. □

Proof of Theorem 5.8. This theorem is clearly a corollary of Theorem 5.11 and Theorem 5.12. □

Implication for Circuits

By Fact 2.6 (page 43), it is easy to see that the AC^0 circuit class is a subclass of the $\mathsf{PH}^{\mathsf{cc}}$ class, therefore Theorem 5.8 implies:

Corollary 5.13 (Theorem 1.8 restated)**.** *If a boolean function family $f = \{f_n\}_{n=1}^{\infty}$ ($f_n : \{0,1\}^{2n} \to \{0,1\}$) is computable by an AC^0 circuit family, then under every input partition (to split the $(2n)$-bit input of every f_n into two n-bit inputs), the resulting communication problem is computable by a limited-memory protocol family with message length $\mathsf{polylog}(n)$ and 3 memory states.*

5.5.2 5 Memory States and $\mathsf{PSPACE}^{\mathsf{cc}}$

Theorem 5.14 (part of Theorem 1.7 restated)**.** *For any integer constant $w \geq 5$, the complexity class defined by message length $\mathsf{polylog}(n)$ and w memory states in our limited-memory model is exactly equivalent to $\mathsf{PSPACE}^{\mathsf{cc}}$. In other words, for every $w \geq 5$, $\mathsf{PSPACE}^{\mathsf{cc}} = \mathsf{SPACE}_{\mathsf{LTD}}[\mathsf{polylog}(n), w]$.*

Proof. First, we prove that $\mathsf{SPACE}_{\mathsf{LTD}}[\mathsf{polylog}(n), w] \subseteq \mathsf{PSPACE}^{\mathsf{cc}}$ for every integer $w \geq 5$. We simply observe that the complexity class $\mathsf{SPACE}_{\mathsf{LTD}}[\mathsf{polylog}(n), w]$ is contained in the complexity class defined by space $\mathsf{polylog}(n)$ in our general two-way communication model. The later one is shown to be equivalent to $\mathsf{PSPACE}^{\mathsf{cc}}$ in Corollary 3.2.

Second, we prove that $\mathsf{PSPACE}^{\mathsf{cc}} \subseteq \mathsf{SPACE}_{\mathsf{LTD}}[\mathsf{polylog}(n), w]$ for every integer $w \geq 5$. We first note that according to Corollary 3.2, $\mathsf{PSPACE}^{\mathsf{cc}}$ contains exactly the set of functions computable with space $\mathsf{polylog}(n)$ general two-way protocols. Furthermore, according to Theorem 5.7, every space $\mathsf{polylog}(n)$ general two-way protocol can be simulated by a limited-memory protocol with message length $\mathsf{polylog}(n)$ and 5 memory states. Therefore for every $w \geq 5$, we have

$$\mathsf{PSPACE}^{\mathsf{cc}} \subseteq \mathsf{SPACE}_{\mathsf{LTD}}[\mathsf{polylog}(n), 5] \subseteq \mathsf{SPACE}_{\mathsf{LTD}}[\mathsf{polylog}(n), w]$$

That concludes our proof. □

Again because of the structural similarity between the $\mathsf{PSPACE}^{\mathsf{cc}}$ class and the NC^1 circuit class (see Fact 2.6 on page 43), we can prove an analog theorem for NC^1:

Theorem 5.15 (restated and proved as Theorem 5.15)**.** *If a boolean function family $f = \{f_n\}_{n=1}^{\infty}$ ($f_n : \{0,1\}^{2n} \to \{0,1\}$) is computable by an NC^1 circuit family, then under every input partition (way of splitting the $(2n)$-bit input of every f_n into two n-bit inputs), the resulting communication problem is computable by a limited-memory protocol family with message length $O(\log n)$ and 5 memory states.*

Proof. The proof is very similar to the proof of Theorem 5.14, except that the circuits involved have no local preprocessing capability, and certain $\mathsf{polylog}(n)$ parameters need to be replaced by $O(\log n)$. □

Chapter 6

Some Remarks on Randomized Memoryless Lower Bound

In this chapter, we present results that may constitute the beginning of some future works: proving lower bounds in the public-coin randomized variant of our memoryless model. We will present mostly concrete and rigorous results, together with some speculative remarks concerning possible future research directions. We include these results in the hope of triggering future researches that continue this work.

In section 4.5.2, we proved a linear message length lower bound for the IP function in our memoryless model. Intuitively, we believe there should be some strong lower bound even in the randomized case. We conjecture a super-polylogarithmic lower bound:

Conjecture 6.1. *The Inner Product over $GF(2)$ function* IP *requires $\omega(\mathsf{polylog}(n))$ message length to be computed in the public-coin randomized memoryless communication model.*

Note that in the communication complexity community, proving explicit lower bounds in the so-called Arthur-Merlin communication model, the public-coin randomized version of $\mathsf{NP}^{\mathsf{cc}}$, is widely considered to be a very hard problem. The only lower bound results we know so far in this direction is achieved in a weaker model called the Merlin-Arthur communication model [24, 36, 26]. See for example [26] for definitions of these communication models. Since we have an exponential separation between the complexity classes $\mathsf{NP}^{\mathsf{cc}}$ and $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ (see Section 1.3.1 page 28), and our memoryless model gives a full characterization for the $\mathsf{P}^{\mathsf{NP}^{\mathsf{cc}}}$ class. This means proving randomized lower bounds in our memoryless model is even harder than proving corresponding lower bounds in the Arthur-Merlin model. Therefore progress in proving randomized lower bounds in our memoryless model will lead to breakthrough in this direction.

In section 4.1, we define a public-coin randomized memoryless communication protocol as a probability distribution over a set of deterministic memoryless protocols. This enables us to use Yao's min-max lemma on our public-coin model ([48], see also section 3.4 in Kushilevitz and Nisan's book [27]). Therefore, in order to prove a strong lower bound in our public-coin model for some explicit function, all we need to do is

to find a "hard" input distribution for this function, and prove a strong distributional lower bound for every deterministic protocol that satisfactorily approximates the function under that input distribution. More precisely, we define the memoryless complexity in the distributional setting below.

Definition 6.1. *Given probability distribution μ on $\{0,1\}^n \times \{0,1\}^n$, we say a deterministic memoryless protocol $\mathcal{P}$ satisfactorily approximates a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ under input distribution μ if*

$$\Pr_{(x,y)\sim\mu}[\mathcal{P}(x,y) = f(x,y)] \geq \frac{2}{3}$$

The distributional memoryless complexity *of function f under input distribution μ, denoted as $S^D_\mu(f)$, is the smallest integer s such that there is a deterministic memoryless protocol with message length s that satisfactorily approximates f.*

A straightforward application of Yao's min-max lemma [48] yields the following.

Lemma 6.1. *For every function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we have $S^R(f) = \max_\mu S^D_\mu(f)$, where μ ranges over all probability distributions over $\{0,1\}^n \times \{0,1\}^n$*

We want to use Fourier analysis of boolean functions as a tool to prove the conjectured randomized lower bound for IP, so we first summarize the relevant results from Fourier analysis.

6.1 Prerequisites: Fourier Analysis of Boolean Functions

Here we only mention some of the concepts and tools in boolean function Fourier analysis that are most relevant to this work. For a more complete treatment, we refer the reader to O'Donnell's survey [32] and online book [33] on this topic.

Note that in the field of boolean function Fourier analysis, it is more convenient to represent the boolean domain with the two elements $\{1,-1\}$ instead of the usual $\{0,1\}$, think of -1 as the usual 1, and 1 as the usual 0, since $-1 = (-1)^1$ and $1 = (-1)^0$. We will use this convention for the remaining part of this chapter. Thus a computational function will be of the form $f : \{1,-1\}^n \to \{1,-1\}$, and a communication problem will be of the form $g : \{1,-1\}^n \times \{1,-1\}^n \to \{1,-1\}$.

Definition 6.2. *For two real-valued functions defined on the boolean cube $f, g : \{1,-1\}^n \to \mathbb{R}$, we define their* inner product $\langle f, g\rangle$ *as follows*

$$\langle f, g\rangle \stackrel{def}{=} \frac{1}{2^n} \sum_{x\in\{0,1\}^n} f(x)g(x)$$

The norm *of a function $f : \{1,-1\}^n \to \mathbb{R}$, denoted as $\|f\|$, is defined to be*

$$\|f\| \stackrel{def}{=} \sqrt{\langle f, f\rangle}$$

By definition, every boolean function $f : \{1,-1\}^n \to \{1,-1\}$ is normalized, that is to say:

Fact 6.2. *For every boolean function $f : \{1,-1\}^n \to \{1,-1\}$, $\|f\| = 1$.*

The linear combination of a set of functions is defined in the natural way.

Definition 6.3. *For a set of real-valued functions $f_1, f_2, \ldots, f_l : \{1,-1\}^n \to \mathbb{R}$ and a set of real numbers $\lambda_1, \lambda_2, \ldots, \lambda_l \in \mathbb{R}$, their linear combination function $f = \sum_{i=1}^{l} \lambda_i \cdot f_i$ is defined to be*

$$\forall x \in \{1,-1\}^n \quad f(x) \stackrel{def}{=} \sum_{i=1}^{l} \lambda_i \cdot f_i(x)$$

Definition 6.4. *The* Fourier space *is the 2^n dimensional linear space over $\mathbb{R}$ formed by the set of all real-valued functions $f : \{1,-1\}^n \to \mathbb{R}$.*

The set of *parity* functions play an important role in Fourier analysis.

Definition 6.5. *For a subset $S \subseteq \{1,2,\ldots,n\}$, the* parity function *on S, denoted as χ_S, is defined as*

$$\forall x = x_1 x_2 \ldots x_n \in \{1,-1\}^n \quad \chi_S(x) \stackrel{def}{=} \prod_{i \in S} x_i$$

As a special case, for every $i \in \{1,2,\ldots,n\}$, we define the dictatorship function *χ_i to be $\chi_{\{i\}}$.*

By convention, $\chi_\emptyset$ is the constant 1 function.

Definition 6.6. *For a set of parity functions $\chi_{S_1}, \chi_{S_2}, \ldots, \chi_{S_l}$, their linear combination χ is defined to be $\chi_{S_1 \ominus S_2 \ominus \ldots \ominus S_l}$, where $\ominus$ denotes the symmetric difference operation over sets. In this work, we call the n dimensional linear space formed by all the parity functions under this linear combination operation as the* parity space*. It is an n-dimensional linear space over $GF(2)$.*

Fact 6.3. *The set of n dictatorship functions form a complete basis of the parity space.*

Fact 6.4. *The set of all 2^n parity functions form an orthonormal basis of the Fourier space.*

6.2 Lower Bounds in Restricted Cases

Given the hardness of proving randomized lower bound in our memoryless model like Conjecture 6.1, we try to first prove lower bounds in some more restricted settings. As shown in Lemma 6.1, proving randomized lower bounds amounts to proving distributional lower bounds under "hard" input distributions. Therefore in this section

we prove some distributional lower bounds for the IP function when only considering some restricted classes of deterministic memoryless protocols.

In the classes of memoryless protocols we consider, we restrict what Alice can send in her messages. In each of Alice's messages, there are two parts: first there is the round number i ($i = 1, 2, \ldots$), then there is a part that contains information about Alice's input x. Our restriction is that every bit in this second part of Alice's message must be computable by a boolean function $p(x)$ that comes from a prescribed "repertoire".

Definition 6.7. *For a set of boolean functions $P = \{p_1, p_2, \ldots, p_t\}$, in which for every $i \in \{1, 2, \ldots, t\}$, $p_i : \{1, -1\}^n \to \{1, -1\}$, we say a memoryless protocol to be a P-protocol if each of Alice's messages in this protocol contains exactly two parts: a round number i ($i = 1, 2, \ldots$), and a bit string of the form $p_{i_1}(x)p_{i_2}(x)\ldots p_{i_m}(x)$, where $i_1, i_2, \ldots, i_m \in \{1, 2, \ldots, t\}$, and x is Alice's input.*

We can prove that if each of Alice's messages simply consists of the round number and a subset of bits from her input x, then we have a $\Omega(\sqrt{n})$ lower bound for IP.

Theorem 6.5. *Define $P_1 = \{\chi_1, \chi_2, \ldots, \chi_n\}$ to be the set of dictatorship functions. Then there exists an input distribution under which* IP *requires message length at least $\Omega(\sqrt{n})$ to be approximated by any deterministic P_1-protocol.*

In fact, we can prove even a more generalized version:

Theorem 6.6. *Define P_2 to be an arbitrarily chosen set of linearly independent parity functions in the parity space. Then there exists an input distribution under which* IP *requires message length at least $\Omega(\sqrt{n})$ to be approximated by any deterministic P_2-protocol.*

Now we present a proof for the more generalized theorem.

First, let us make an observation that the behavior of our memoryless protocols can be simulated by depth-3 circuits.

Lemma 6.7. *Given a deterministic memoryless P-protocol with message length m (here $P = \{p_i\}$ is any set of boolean functions $p_i : \{1, -1\}^n \to \{1, -1\}$), suppose the function it computes exactly is $f : \{1, -1\}^n \times \{1, -1\}^n \to \{1, -1\}$ then for every fixed input y_0 for Bob there exists a depth-3 circuit of size $2^{O(m)}$ which computes the function $f(x, y_0)$ on the single input x. Every input gate of this circuit is a* literal *of form either $p(x)$ or $\neg p(x)$, where $p \in P$.*

Proof. Now we fix an input value y_0 for Bob, let us analyze the behavior of this protocol.

First, we observe that by Definition 4.1, in a memoryless protocol, when Bob's input y and the round number i is fixed, we can categorize all messages $(i, \alpha) \in \{1, -1\}^m$ that Bob may receive from Alice in the i-th round into three categories: 1-messages, these messages will trigger Bob to output 1 and halt if received in the i-th round, that is, $B(y, i, \alpha) = 1$; (-1)-messages, these messages will trigger Bob to output -1 and halt if received in the i-th round, that is, $B(y, i, \alpha) = -1$; and

$\perp$-messages, these messages will trigger Bob to continue if received in the i-th round, that is $B(y, i, \alpha) = \perp$.

That is to say, Bob outputs (-1) in the protocol if he receives a (-1)-message in a certain round, and he does not receive any of the 1-messages in previous rounds. Since every bit in every message Alice sends except for the round number part is computed based on some function p from P, that means checking a particular message Alice sent in a certain round against a particular 1-message (or (-1)-message) for that round can be done with a *term*, that is, a conjunction of literals. Therefore, it is easy to see that to simulate Bob's behavior with a fixed input value y, we just need a depth 3 circuit. Furthermore, since Alice can send at most 2^m messages, and each message is of length at most m, the size of the circuit can be bounded by $2^{O(m)}$.

In fact, if we define function $B_y : \{1, -1\}^m \to \{0, 1, \perp\}$, $B_y(i, \alpha) = B(y, i, \alpha)$, then the structure of the depth-3 circuit described above is completely determined by this function B_y. □

Next, we need to argue that such depth 3 circuits cannot compute too complicated functions. The following result from Boppana's 1997 paper gives us what we need. It gives a nice upper bound on the average sensitivity of bounded-depth circuits.

Lemma 6.8 (Main Result of [13])**.** *An unbounded fan-in circuit of depth d and size s has average sensitivity at most $O((\log s)^{d-1})$.*

If we talk about P_1-protocols (P_1 being the set of all dictatorship functions, as defined in Theorem 6.5), then we can apply Boppana's lemma directly. But once we start to talk about arbitrary parity functions, the simulation circuits we use are no longer depth-3 circuits in the traditional sense.

But we observe that in fact every complete basis in the parity space works just fine like the "dictatorship basis". The key point is that given a complete basis P_2^* in the parity space, which consists of n linearly independent parity functions in the parity space, we can devise a new encoding for the input space $\{1, -1\}^n$. For each input $x \in \{1, -1\}^n$, we simply use the output of these n parity function on x as the new "encoding" for x! It is easy to verify that this mapping is one-to-one, therefore our proposed encoding is valid. Let us call this encoding P_2^*-encoding.

Fact 6.9. *Given a complete basis P_2^* of the parity space, every parity function can be written as a linear combination of a subset of basis functions from P_2^* by definition. Suppose a parity function χ can be written as a linear combination of t basis functions from P_2^* then χ has sensitivity t on every input under the P_2^* encoding.*

The "traditional" encoding of the boolean cube is merely a special case of P-encoding in which P is chosen to be the set of the n dictatorship functions.

It is straightforward to generalize Boppana's lemma to a general P-encoding:

Lemma 6.10 (Generalized Version of Boppana's Lemma)**.** *Given any complete basis P of the parity space, a depth d and size s unbounded-fanin circuit with input gates coming from literals from P has average sensitivity at most $O((\log s)^{d-1})$ under P-encoding.*

Now we are ready to prove Theorem 6.6:

Proof of Theorem 6.6. Given a set of linearly independent functions P_2 in the parity space, we can always augment it to form a complete basis P_2^* of the parity space. Now we take the linear combination of all the n basis functions in P_2^* in the parity space, we get a parity function χ_S, where S is a subset of $\{1, 2, \ldots, n\}$. χ_S has average sensitivity n in P_2^*-encoding. We define $\bar{y}$ as follows

$$\bar{y}_i \stackrel{\text{def}}{=} \begin{cases} -1 & \text{if } i \in S \\ 1 & \text{if } i \notin S \end{cases}$$

Clearly for every $x \in \{1, -1\}^n$, $\mathrm{IP}(x, \bar{y}) = \chi_S(x)$.

We define input distribution μ to be a uniform distribution over all $(x, \bar{y}) \in \{1, -1\}^n \times \{\bar{y}\}$ and 0 everywhere else. Suppose there is a memoryless P_2-protocol $\mathcal{P}$ of message length m that approximates IP satisfactorily under distribution μ then denote the function the protocol computes on x given fixed input $\bar{y}$ as $f : \{1, -1\}^n \to \{1, -1\}$.

1. According to Lemma 6.7, f can be simulated with a depth-3 circuit of size $2^{O(m)}$. The output of this circuit has average sensitivity at most $O(m^2)$ under P_2^*-encoding. This is a corollary of the generalized version of Boppana's lemma (Lemma 6.10).

2. Since the protocol $\mathcal{P}$ satisfactorily approximates IP under distribution μ, based on the definition of distribution μ, we have

$$\Pr_{x \sim U(\{1,-1\}^n)}[f(x) = \chi_S(x)] \geq \frac{2}{3}$$

   here $U(\{1, -1\}^n)$ is the uniform distribution over $\{1, -1\}^n$. This implies that the average sensitivity of f under P_2^*-encoding must be at least $\Omega(n)$.

That means $m = \Omega(\sqrt{n})$. □

6.3 Going Forward

To achieve our final goal of proving Conjecture 6.1, we need to consider bigger and bigger set of functions P and prove lower bounds for IP among these P-protocols. For example, we may consider the following sets of functions in order, each being a superset of the previous one:

1. P_3, the set of all parity functions;

2. P_4, an arbitrarily chosen orthonormal basis for the Fourier space that consists of only boolean functions;

3. P_5, the set of all boolean functions;

A distributional lower bound for IP among P_5-protocols clearly implies our main conjecture, Conjecture 6.1. Therefore the sets P_3, P_4 and P_5 give a possible step-by-step roadmap towards achieving our goal.

Now we make some speculative remarks on how to finish each step.

6.3.1 P_3-Protocols: Repertoire Containing All Parity Functions

We first observe that in the IP function, Bob, given his input y, is simply computing a parity function on x. Each different input value for y corresponds to a different parity function on x. Overall, Bob needs to be prepared to compute every possible parity function on every possible x. We actually already used this fact in the proof of Theorem 6.6.

We observe that in theorem 6.6, once the n parity function repertoire is fixed, we can immediately find one parity function χ on x, which corresponds to one specific input value for y. And this χ is very hard for this chosen repertoire. Even if our repertoire P_3 contains all 2^n parity functions, for every specific protocol $\mathcal{P}$ with message length m, the number of functions that will actually be used by Alice to form her messages in the protocol is limited to $m \cdot 2^m$. There is still hope that we can always find a hard parity function χ against these $m \cdot 2^m$ parity functions when $m = O(\mathsf{polylog}(n))$.

We further observe that in each round of the protocol, Bob can get the output of at most m parity functions. And he will be able to compute every parity function that is contained in the subspace spanned by these parity functions in the parity space. In each round, the number of such parity functions is at most 2^m, and overall the number of such parity functions that can be computed this way during the whole protocol execution is bounded by 2^{2m}, we denote this set of parity functions as $Q_{\mathcal{P}}$. All the other parity functions that are not in $Q_{\mathcal{P}}$ seem to be hard. Because Bob has very limited ability to carry information across different rounds, therefore intuitively he has very limited means to correlate the parity functions Alice sends to him in different rounds to approximate the parity functions outside $Q_{\mathcal{P}}$ well.

To make things more concrete, we propose the following conjecture:

Conjecture 6.2. *If $m = O(\mathsf{polylog}(n))$, given an arbitrary way of choosing $m \cdot 2^m$ functions from P_3 (the set of all 2^n parity functions) to form Alice's messages, there always exists one parity function χ, which corresponds to the input value $y = \bar{y}$, such that no matter how we define the function $B_{\bar{y}}$ as in the proof of Lemma 6.7, the Fourier spectrum (over the parity basis) of the depth-3 circuit determined by $B_{\bar{y}}$ will always have a sub-constant $o(1)$ coefficient in front of χ.*

6.3.2 P_4-protocols: Repertoire Containing an Arbitrary Fourier Basis

Intuitively, we do not believe that the parity basis is that special among all orthonormal basis for the Fourier space. Therefore, what holds true for the parity basis should

also hold true for any arbitrarily chosen basis P_3 to some extent. In particular, we propose the following conjecture connecting the parity basis to any other Fourier basis that consists entirely of boolean functions of the the form $p : \{1, -1\}^n \to \{1, -1\}$.

Conjecture 6.3. *Suppose $P_3 = \{p_1, p_2, \ldots, p_{2^n}\}$ is an orthonormal basis for the Fourier space such that for every $i \in \{1, 2, \ldots, 2^n\}$, $p_i : \{1, -1\}^n \to \{1, -1\}$ then there exists bijective functions $\psi : \{1, 2, \ldots, 2^n\} \to P(\{1, 2, \ldots, n\})$ [1], $\phi : \{1, -1\}^n \to \{1, -1\}^n$ and functions $g_1 : \{1, -1\}^n \to \{1, -1\}$ and $g_2 : \{1, 2, \ldots, 2^n\} \to \{1, -1\}$, such that for every $i \in \{1, 2, \ldots, 2^n\}$ and every $x \in \{1, -1\}^n$, we have*

$$p_i(x) = \chi_{\psi(i)}(\phi(x)) \cdot g_1(x) \cdot g_2(i)$$

6.3.3 P_5-protocols: Arbitrary Repertoire

Here we deal with general memoryless protocols with message length m. The basic intuition is that there may exist a way to simulate every memoryless protocol $\mathcal{P}$ with some P_4-protocol $\mathcal{P}'$ with message length $\mathsf{poly}(m)$, where P_4 is some orthonormal basis of the Fourier space specifically chosen for $\mathcal{P}$. We may use tools similar to the standard Gram-Schmidt orthonormalization process to construct P_4. If such a universal simulation scheme exists, then the lower bound for P_4-protocols would imply lower bound for P_5-protocols.

[1] $P(\{1, 2, \ldots, n\})$ is the power set of $\{1, 2, \ldots, n\}$.

Bibliography

[1] personal discussion with H. Buhrman and F. Speelman on January 14, 2014.

[2] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. *TOCT*, 1(1), 2009.

[3] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

[4] E. Allender and U. Hertrampf. Depth reduction for circuits of unbounded fan-in. *Inf. Comput.*, 112(2):217–238, 1994.

[5] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. Preliminary version in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 20–29, 1996.

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

[7] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory (preliminary version). In *FOCS*, pages 337–347. IEEE Computer Society, 1986.

[8] D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

[9] P. Beame, M. Tompa, and P. Yan. Communication-space tradeoffs for unrestricted protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 420–428, 1990.

[10] P. Beame, M. Tompa, and P. Yan. Communication-space tradeoffs for unrestricted protocols. *SIAM J. Comput.*, 23(3):652–661, 1994.

[11] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity*, pages 210–220, 2011.

[12] B. Bollobás. *Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability*. Cambridge University Press, 1986.

[13] R. B. Boppana. The average sensitivity of bounded-depth circuits. *Information Processing Letters*, 63(5):257–261, 1997.

[14] A. Borodin, D. Dolev, F. E. Fich, and W. J. Paul. Bounds for width two branching programs. *SIAM J. Comput.*, 15(2):549–560, 1986.

[15] M. Braverman. Poly-logarithmic independence fools bounded-depth boolean circuits. *Commun. ACM*, 54(4):108–115, 2011.

[16] J. Brody, S. Chen, P. A. Papakonstantinou, H. Song, and X. Sun. Space-bounded communication complexity. In R. D. Kleinberg, editor, *ITCS*, pages 159–172. ACM, 2013.

[17] H. Buhrman, S. Fehr, C. Schaffner, and F. Speelman. The garden-hose game: A new model of computation, and application to position-based quantum cryptography. *CoRR*, abs/1109.2563, 2011.

[18] H. Buhrman, N. K. Vereshchagin, and R. de Wolf. On computation and communication with small bias. In *IEEE Conference on Computational Complexity*, pages 24–32. IEEE Computer Society, 2007.

[19] P. Frankl and Z. Füredi. A short proof for a theorem of harper about hamming-spheres. *Discrete Mathematics*, 34(3):311–313, 1981.

[20] R. Impagliazzo, V. Kabanets, and A. Kolokolova. An axiomatic approach to algebrization. In M. Mitzenmacher, editor, *STOC*, pages 695–704. ACM, 2009.

[21] R. Impagliazzo and R. Williams. Communication complexity with synchronized clocks. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity*, pages 259–269, 2010.

[22] P. Indyk and D. Woodruff. Tight lower bounds for the distinct elements problem. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science*, pages 283–289, 2003.

[23] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 539–550, 1988.

[24] H. Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, pages 118–134, 2003.

[25] H. Klauck. Quantum and classical communication-space tradeoffs from rectangle bounds. In *Proceedings of the 24th International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 384–395, 2004.

[26] Hartmut Klauck. On Arthur Merlin games in communication complexity. In *IEEE Conference on Computational Complexity*, pages 189–199. IEEE Computer Society, 2011.

[27] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[28] T. W. Lam and W. L. Ruzzo. Results on communication complexity classes. *J. Comput. Syst. Sci.*, 44(2):324–342, 1992.

[29] T. W. Lam, P. Tiwari, and M. Tompa. Tradeoffs between communication and space. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 217–226, 1989.

[30] T. W. Lam, P. Tiwari, and M. Tompa. Trade-offs between communication and space. *J. Comput. Syst. Sci.*, 45(3):296–315, 1992.

[31] I. Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991.

[32] R. O'Donnell. Some topics in analysis of boolean functions. In *Proceedings of the 40th Annual ACM Symposium on Theory of computing*, pages 569–578. ACM, 2008.

[33] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[34] P. A. Papakonstantinou, D. Scheder, and H. Song. Overlays and limited-memory communication mode(l)s. 2014.

[35] S. Ramanujan. A proof of bertrand's postulate. *Journal of the Indian Mathematical Society*, 11:181–182, 1919.

[36] R. Raz and A. Shpilka. On the power of quantum proofs. In *Proceedings of the 19th Annual IEEE Conference on Computational Complexity*, pages 260–274, 2004.

[37] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM*, 39(3):736–744, 1992.

[38] A. A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition (russian). *Matematicheskie Zametki*, 41(4):598–607, 1.

[39] C. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal*, 28:59–98, 1949.

[40] Shearer. Lower bound for width-2 branching programs. Private communication with Paul Beame.

[41] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In A. V. Aho, editor, *STOC*, pages 77–82. ACM, 1987.

[42] R. E. Stearns, J. Hartmanis, and P. M. Lewis II. Hierarchies of memory limited computations. In *SWCT (FOCS)*, pages 179–190. IEEE Computer Society, 1965.

[43] L. G. Valiant. Graph-theoretic arguments in low-level complexity. In J. Gruska, editor, *MFCS*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 1977.

[44] H. Vollmer. *Introduction to Circuit Complexity - A Uniform Approach*. Springer, 1999.

[45] Emo Welzl. Boolean satisfiability – combinatorics and algorithms (lecture notes), 2005.

[46] R. Williams. Nonuniform acc circuit lower bounds. *J. ACM*, 61(1):2, 2014.

[47] A. C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, pages 209–213, 1979.

[48] A. C. Yao. Lower bounds by probabilistic arguments. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 420–428, Los Alamitos, CA, USA, 1983. IEEE Computer Society.