Juraj Hromkovič

Communication Complexity and Parallel Computing

Springer

Texts in Theoretical Computer Science
An EATCS Series

Springer-Verlag Berlin Heidelberg GmbH

Juraj Hromkovič

Communication Complexity and Parallel Computing

With 40 Figures

Springer

Prof. Dr. Juraj Hromkovič
Institut für Informatik und Praktische Mathematik
Universität Kiel
Olshausenstr. 40
D-24098 Kiel, Germany

Series Editors

Prof. Dr. Wilfried Brauer
Institut für Informatik, Technische Universität München
Arcisstrasse 21, D-80333 München, Germany

Prof. Dr. Grzegorz Rozenberg
Institute of Applied Mathematics and Computer Science
University of Leiden, Niels-Bohr-Weg 1, P.O. Box 9512
2300 RA Leiden, The Netherlands

Prof. Dr. Arto Salomaa
Data City
Turku Centre for Computer Studies
FIN-20520 Turku, Finland

CR Subject Classification (1991): F.1.2-3, F.2.3, G.2.2, B.7.1, C.2.1, F.4.3

Library of Congress Cataloging-in-Publication Data

Hromkovič, Juraj, 1958-
Communication complexity and parallel computing / Juraj Hromkovič.
p. cm. -- (Texts in theoretical computer science)
Includes bibliographical references and index.

1. Parallel processing (Electronic computers) 2. Computational complexity. I. Title. II. Series.
QA76.58.H76 1997
005.2'75--dc21 96-29508
CIP

ISBN 978-3-642-08185-9 ISBN 978-3-662-03442-2 (eBook)
DOI 10.1007/978-3-662-03442-2

Originally published by Springer-Verlag Berlin Heidelberg New York in 1997.
Softcover reprint of the hardcover 1st edition 1997

Typesetting: Camera ready by author
Cover Design: *design & production* GmbH, Heidelberg
SPIN: 10083911 45/3142-5 4 3 2 1 0 – Printed on acid-free paper

Preface

The communication complexity of two-party protocols is an only 15 years old complexity measure, but it is already considered to be one of the fundamental complexity measures of recent complexity theory. Similarly to Kolmogorov complexity in the theory of sequential computations, communication complexity is used as a method for the study of the complexity of concrete computing problems in parallel information processing. Especially, it is applied to prove lower bounds that say what computer resources (time, hardware, memory size) are necessary to compute the given task. Besides the estimation of the computational difficulty of computing problems the proved lower bounds are useful for proving the optimality of algorithms that are already designed. In some cases the knowledge about the communication complexity of a given problem may be even helpful in searching for efficient algorithms to this problem.

The study of communication complexity becomes a well-defined independent area of complexity theory. In addition to a strong relation to several fundamental complexity measures (and so to several fundamental problems of complexity theory) communication complexity has contributed to the study and to the understanding of the nature of determinism, nondeterminism, and randomness in algorithmics. There already exists a non-trivial mathematical machinery to handle the communication complexity of concrete computing problems, which gives a hope that the approach based on communication complexity will be instrumental in the study of several central open problems of recent complexity theory.

This book presents the basic concepts in the study of communication complexity and in its application in proving lower bounds on distinct fundamental complexity measures. In the applications we focus on the complexity measures of realistic, parallel computing models like combinational (Boolean) circuits, VLSI circuits and interconnection networks. The book is written at a level accessible to advanced undergraduates and to graduate students. Since it gives a survey on the study of communication complexity and formulates a lot of research problems we expect it will also prove to be a reference for researchers in this area.

First of all I would like to thank Grzegorz Rozenberg and Arto Salomaa, who encouraged me to write this book, and to Burkhard Monien and Wolfgang Thomas for the good working atmosphere in Paderborn and Kiel during the work on the book. I am indebted to Ingeborg Mayer, Andrew Ross and Hans

Wössner of Springer-Verlag for their editorial help, comments and suggestions which essentially contributed to the quality of the presentation of the book.

I am grateful for the comments and materials received from several people, including Ivana Černá, Josef Gruska, Ivana Höltring, Oliver Matz, Dana Pardubská, Anna Slobodová, Ondrej Sýkora, Imrich Vrťo, Juraj Waczulík, and Avi Wigderson. Special thanks go to Martin Dietzfelbinger and Georg Schnitger for an intensive research cooperation, which during the work on the book led to the solutions of some open problems enabling to present a complete picture of some research directions on communication complexity in this book. I would like to thank Sebastian Seibert for carefully reading of the whole manuscript. I am indebted to Heidi Luca-Gottschalk, Oliver Matz, Walter Unger, and Thomas Wilke for the help with LaTeX and to Marion Küpper and Gerti Rosenfeld for typing some parts of the manuscript.

My deepest thanks go to Tanja, not only for carefully typing of several parts of this book.

Kiel, January 1997 Juraj Hromkovič

Table of Contents

1. Introduction

1.1 Motivation and Aims

Parallel data processing has become a reality. Large numbers of processors (computers) cooperating to compute a given task have brought an essential speed-up of classical sequential computations. Many computing problems requiring too much time to be solved in real time by sequential machines can be computed in parallel very quickly. Because there are many computing tasks requiring a real-time solution in industry, the investigation of parallel computing is becoming one of the central parts of theoretical computer science and computer engineering. In the theory the study ranges from the study of abstract parallel computing models (complexity measures) and limits to parallel computations (classification of computing problems in two classes: problems that allow quick parallel solution by using a realistic number of processors and problems for which no efficient parallel algorithm exists) to the development of parallel programming languages, communication algorithms for different parallel architectures, and automatic design of VLSI circuits.

This book is devoted to the abstract theory of parallel complexity measures, where one tries to measure the computational difficulty of a computing problem as an inherent property of the problem. We are mainly interested in proving some lower bounds that say what computer resources (time, hardware, memory size) are necessary to compute the given task (problem). The lower bound proofs are usually connected with a detailed analysis of the given computing problems and so they provide knowledge about the nature of the problem. Besides the classification of the computational difficulty of the computing problems the proved lower bounds may be usefull in searching for efficient algorithms for the problem as well as for proving the optimality of algorithms that are already designed.

This book concentrates on only one complexity measure – communication complexity of two-party protocols. Informally, a *two-party protocol* (shortly, *protocol*) is a computing model consisting of two abstract computers C_I and C_{II} with unlimited power. In this book we call C_I the first (left) computer and C_{II} the second (right) computer (see Figure 1.1).

At the beginning one considers that an input w is divided in a balanced way between C_I and C_{II} (for instance, C_I has the first half of the input w and C_{II} obtains the second half of w). The aim is to compute the output for the

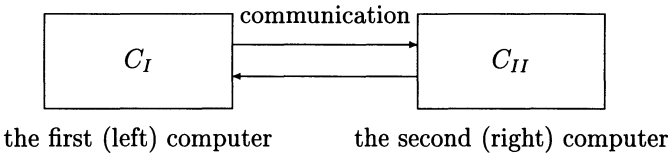

Fig. 1.1. The model of two-party communication protocols

given input w. To achieve this aim C_I and C_{II} are allowed to communicate by exchanging some binary messages. The communication complexity of the computation on w is the sum of the lengths of all messages exchanged between C_I and C_{II}. The communication complexity of a protocol computing a finite problem is the maximum of the communication complexities over all inputs of the problem.

Some natural questions arise: Why write a book about one complexity measure only? Why study a simple model, where the parallelism is restricted to two computers only? The reason is that this communication complexity is useful for parallel computing in a similar way as Kolmogorov complexity has been shown to be useful for the study of sequential computations. The communication complexity measure is so close to a lot of realistic parallel (and even sequential) complexity measures that it can be used as method for proving lower bounds on the complexity measures of fundamental computing models like VLSI circuits, combinatorial circuits, interconnection networks, Turing machines, etc. Moreover, it enables one to achieve new results in the principal topic of theoretical computer science dealing with the comparison of the computational power of deterministic, nondeterministic, and probalistic computations.

The main aim of this book is to provide an overview of the study of communication complexity and its application to parallel computing. The book is written for students as an introduction to this topic as well as for use by researchers interested in working in this area. For these reasons the book includes a lot of exercises of varying difficulty as well as the formulations of research (open) problems of interest.

More precisely, the central aims of the book are the following.

(i) To fix the formalism in the definition of two-party protocols (note that a few distinct formalisms have been considered so far in the literature) and to define several versions of communication complexity depending on the partitions of the input, because different applications require consideration of different sets of input partitions.

(ii) To give an overview of the basic theoretical properties of the communication complexity measure.

(iii) To give an overview of the methods for proving lower bounds on the communication complexity of concrete computing problems. This part is of most interest because the lower bounds on the communication complexity of concrete problems are applied to get lower bounds on the complexity measures of realistic models of parallel computations.

(iv) To give an overview of the main results concerning nondeterministic and probabilistic protocols and their communication complexity. For communication complexity one can establish results that people try but fail to establish for time complexity of sequential computations. For instance, nondeterminism and Monte Carlo probabilism can be much more powerful than determinism for two-party protocols, but deterministic communication complexity is polynomially related to Las Vegas communication complexity.

(v) To give an overview of the use of communication complexity for proving lower bounds on different complexity measures of distinct computing models. The main emphasis is placed on the relation between communication complexity and area-time complexity measures of distinct circuit models.

Recently, a lot of results about communication complexity have appeared in the literature. We are not able to present all of them. The ones chosen for presentation in this book are explained very carefully and their proofs usually contain more details than their counterparts in the journal or proceedings publications. Some of the results from the literature are formulated in the exercises or mentioned in the bibliographical remarks only. Because the central idea of the book is to give methods for proving lower bounds on parallel complexity measures by applying communication complexity, the overviews connected with the aims (iii) and (v) above are more exhaustive than the other ones. The book does not contain any result connected with relativized communication complexity classes and oracles, whose study represents a research direction in communication complexity theory too.

Proving non-trivial lower bounds on the complexity of concrete computing problems and especially the development of mathematical proof methods enabling one to achieve them is probably one of the few central topics of recent theoretical computer science. The reason why we are not able to solve the P versus NP question, which is perhaps the single most important problem of theoretical computer science, is the lack of powerful lower bounds methods in complexity theory. The two-party communication protocol model has enabled us to prove a sequence of important lower bounds and so to contribute essentially to complexity theory. This model provides several lower bound techniques based on information theoretic arguments and is elegant in use. There already exists non-trivial mathematical machinery to handle the communication complexity of concrete problems. From these reasons we hope that the approach based on communication protocols will be instrumental in solving further important open

problems in theoretical computer science. Belief in the future and the fruitfulness of the approach was the author's main reason starting to write this book. We hope that the book will be a modest contribution to motivating to research work on this topic and providing a sound basis for further investigations.

1.2 Concept and Organization

This account of communication complexity is divided into four chapters. Chapter 2 deals with communication complexity as an abstract complexity measure. Chapter 3 relates communication complexity to the complexity measures of Boolean circuits. Chapter 4 relates communication complexity to the complexity of VLSI computations and to interconnection networks. Chapter 5 is devoted to the applications of communication complexity for some sequential models.

More precisely, Chapter 2 is divided into seven sections. Section 2.1 provides definitions of some basic notions connected with formal language theory, Boolean function theory, and graph theory. Section 2.2 presents the formal model of two-party protocols and defines the communication complexity of computing problems according to a fixed partition of the input. Basic methods for proving lower bounds on communication complexity are presented and compared here. In Section 2.3 the general communication complexity is defined as the minimum of communication complexities of all protocols over all input partitions dividing the input in a balanced way (or almost balanced way). This communication complexity is the measure with the closest connection to the parallel complexity measures studied in Chapters 3 and 4. Section 2.3 also provides some fundamental theoretical properties of communication complexity and relates the communication complexity of language recognition to the Chomsky hierarchy in formal language theory. In Section 2.4 one-way protocols are defined as protocols with a restriction on the communication allowing only one message flowing from C_I to C_{II} during the whole computation. After C_{II} has received this message, it must immediately compute the result. Based on one-way protocols, one-way communication complexity is defined. Basic lower bounds methods for one-way communication complexity are presented there and the communication complexity and the one-way communication complexity of concrete problems are compared. Section 2.5 is devoted to the definitions of nondeterministic and randomized protocols as well as to the definitions of the corresponding complexity measures. The computational powers of deterministic, nondeterministic, and randomized protocols are compared in this section. Section 2.6 proposes an improved model of protocols which captures better than the standard model the definition of a complexity measure describing the communication complexity of a computing problem as an inherent property of this problem. It is shown that this communication model can be applied to prove reasonable lower bounds on parallel computations even if the standard model fails to do so. The last section of this chapter is devoted to bibliographical remarks.

Chapter 3 is divided into seven sections and it is devoted to the relation between communication complexity and the complexity measures of Boolean circuits. Section 3.1 provides the introduction to this chapter. Section 3.2 gives the definitions of fundamental Boolean circuit models and their complexity measures. Section 3.3 shows how the communication complexity of a computing problem P provides lower bounds on the layout area of any Boolean circuit computing P. Section 3.4 deals with the classical open problem of proving a superlinear lower bound on the combinational complexity (number of processors of Boolean circuits) of a concrete Boolean function. It is shown that this nonlinear lower bound can be proved if the circuit has a sublinear separator. Special attention is devoted to lower bounds on the size of planar Boolean circuits. A proof of a version of the Planar Separator theorem is given in a well-structured, detailed form. Section 3.5 is devoted to lower bounds on Boolean circuits with unbounded fan-in and fan-out. Section 3.6 uses the communication protocol model for the computations of relations instead of functions in order to develop a lower bound method on the depth of Boolean circuits. Bibliographical remarks close Chapter 3.

Chapter 4 consists of six sections. Section 4.1 provides an introduction and Section 4.2 contains the definitions of VLSI circuit models and their complexity measures. Section 4.3 shows that one-way communication complexity provides a lower bound on the area of VLSI circuits and how communication complexity can be used to obtain lower bounds on the tradeoffs of area and time complexity measures of VLSI circuits. How to get still higher lower bounds on the VLSI complexity measures if the topology of circuits is restricted in some way is shown here too. In Section 4.4 a general model of interconnection networks with powerful processors is defined. It is shown how to apply communication complexity to obtain lower bounds on time and size of concrete interconnection networks. Section 4.5 is devoted to so-called multilective VLSI circuits, which are a generalization of the basic VLSI circuit model. More elaborate combinatorial considerations than for standard VLSI circuits are needed to show how communication complexity can be used to get lower bounds on multilective VLSI computations. Section 4.6 is devoted to bibliographical remarks.

Chapter 5 deals with the relations between communication complexity and some complexity measures of sequential computation. It consists of five sections. Section 5.1 is an introduction. Section 5.2 introduces a uniform model of one-way communication protocols and shows that the corresponding uniform one-way communication complexity is strongly related to the size of deterministic finite automata. Section 5.3 relates communication complexity to the time and space complexity of Turing machines. Section 5.4 shows how communication complexity can be used to obtain lower bounds on the size and the depth of decision trees and branching programs. Section 5.5 contains bibliographical remarks.

Each chapter is organized according to the same pattern. It starts with an introduction giving the basic information about the main motivations, aims, and content of the chapter. And it ends with some bibliographical remarks.

Here the original sources of the main assertions presented in the chapter are given. Here too one can find more information and discussions about concepts, facts, and ideas close to the content of the chapter, but not presented there. The structure of the sections is uniform too. They are divided into subsections. The first subsection is an introduction giving information about the content of the section. Usually, the section finishes with the subsections Exercises and Research Problems. The exercises may be used to learn to work with the formalism used, to better understand the proof methods presented in the section, to extend the ideas of the section, and to find formutions of further results connected with the topic of the section but not presented and proved there. The exercises are divided into three classes according to their difficulty. The exercises without any star are for simple training in proof techniques presented in the section. The exercises marked with one star require nontrivial proofs and the doubly starred exercises are usually assertion from published papers requiring nontrivial proof ideas which cannot be considered as straightforward extensions of proof techniques presented in the book. The Research Problems subsections provide useful ideas for further investigation in the research direction of the corresponding section. The known open problems considered to be really hard are labelled by one or two stars.

1.3 How to Read the Book

The book contains more material than one can present in one course. If somebody is interested to learn communication complexity as an abstract complexity measure it is sufficient to read Chapter 2 only. If one follows the main goal of the book (to use communication complexity for proving lower bounds on some complexity measures of parallel computations), then it is not necessary to read the whole of Chapter 2 before starting to read the topic of proper interest in Chapters 3 and 4. It suffices to read Sections 2.1 and 2.2, Sections 2.3.1, 2.3.2, and 2.3.3 of Section 2.3, Section 2.4, and Sections 2.6.1, 2.6.2, 2.6.3, and 2.6.4 of Section 2.6. How much from Chapters 3 and 4 has to be read depends on one's interest. The first two sections of each of these two chapters provides the basic information about standard circuit models and their relations to communication complexity. Almost all ideas and proofs are relatively simple in these parts. The following sections usually contain specialized results requiring more involved ideas and proofs than those in the first two sections.

2. Communication Protocol Models

2.1 Basic Notions

2.1.1 Introduction

The aim of Section 2.1 is to define some standard basic notions from formal language theory and Boolean function theory used in this book. The reader is advised that this section does not give a detailed exposition (including illustrative examples of objects defined or some theorems and proofs about the defined objects) of the topics covered, but rather a setting of the notations and concepts which will be freely used throughout the book. Formal definitions are given for all notions which will be formally handled in the next chapters. Formal definitions of the fundamental notions (for instance, for finite automata or context-free grammars here) are not included, because their formal descriptions are not needed to prove any results presented in this book. Nevertheless we shall use these notions to discuss the consequences of the presented results for other theories or applications, and sometimes to formulate statements. In these cases, we assume that the reader is familiar with the fundamentals of formal language theory, Boolean function theory, graph theory and combinatorics.

Section 2.1 is organized as follows. The basic notions of formal language theory including alphabets, words, languages, and the operations on words and languages are presented in Section 2.1.2. Some fundamentals of Boolean function theory, including the representation of Boolean functions by Boolean matrices and formulas, are given in Section 2.1.3. Section 2.1.4 contains a formal description of the representation of computing (algorithmical) problems used as well as some definitions of the basic notions connected with undirected graphs.

The last subsection contains some exercises connected with the notions defined in Section 2.1, involving some fundamental observations about the objects defined. It is assumed that the statements included in the exercises are familiar to the reader.

2.1.2 Alphabets, Words, and Languages

Here, we fix the notations of some basic notions of formal language theory. We start with the notations for sets and operations on sets.

Definition 2.1.2.1 *Let A be a set. Then $|A|$ denotes the* **cardinality of A**. *For any two sets A and B,*

(i) $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$ *is the* **union of A and B**

(ii) $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$ *is the* **intersection of A and B**

(iii) $A - B = \{x \mid x \in A \text{ and } x \notin B\}$ *is the* **difference between A and B**

(iv) $A \oplus B = (A \cup B) - (A \cap B) = \{x \mid (x \in A \text{ and } x \notin B) \text{ or } (x \in B \text{ and } x \notin A)\}$ *is the* **symmetric difference between A and B**

(v) $A \times B = \{(x, y) \mid x \in A \text{ and } y \in B\}$ *is the* **Cartesian product of the sets A and B**

(vi) $A \subseteq B$ *denotes the fact that* **A is a subset of B** *(i.e., every element of A is also element of B)*

(vii) $A = B$ *denotes the fact that* **A and B are identical** *(i.e., $A \subseteq B$ and $B \subseteq A$)*

(viii) $A \subsetneq B$ *(or $A \subset B$) denotes the fact that* **A is a proper subset of B** *(i.e., $A \subseteq B$ and $B - A$ contains at least one element)*

(ix) $A \not\subseteq B$ *denotes the fact that* **A is not a subset of B** *(i.e., there is at least one element in A which is not in B)*

(x) $A \neq B$ *denotes the fact that* **A and B are two different sets** *(i.e., $A \not\subseteq B$ or $B \not\subseteq A$).*

$\mathbb{N}$ *stands for the* **set of all nonnegative integers**, *and $\emptyset$ stands for the* **empty set** *(set containing no element). $\boldsymbol{R}$ denotes the* **set of all reals**, *and $\boldsymbol{Q}$ denotes the* **set of all rational numbers**.

Definition 2.1.2.2 *Any non-empty, finite set is called an* **alphabet**. *Any element of an alphabet Σ is called a* **symbol (letter)** *of Σ. A* **word over the alphabet $\boldsymbol{\Sigma}$** *is any finite sequence of symbols from Σ. The set of all words over the alphabet Σ is denoted by Σ^*. The* **length of a word w**, *denoted $|w|$, is the number of symbols in w. The* **empty word λ** *is the only word consisting of zero symbols. The set of all non-empty words over the alphabet Σ is denoted by $\Sigma^+ = \Sigma^* - \{\lambda\}$. For any word w over Σ, and any symbol $a \in \Sigma$, $\#_a(w)$ denotes the number of occurrences of the symbol a in the word w.*

In what follows, we will mostly use the two-letter alphabet $\{0, 1\}$. Note that for every word w over an alphabet Σ

$$|w| = \sum_{a \in \Sigma} \#_a(w).$$

Despite the fact that words are defined as sequences, we will omit the commas in our descriptions. So, we will use 011010 to denote the word 0, 1, 1, 0, 1, 0 over the alphabet $\{0, 1\}$.

Definition 2.1.2.3 *Given two words v and w over Σ, we define the* **concatenation of v and w**, *denoted by vw (and sometimes by $v \cdot w$) as the word z which consists of the symbols of v in the same order, followed by the symbols of w in the same order. A* **prefix of a word w** *over Σ is any word v such that $w = vu$ for some word u over Σ. A* **proper prefix of a word w** *over Σ is any word v such that $w = vu$ for some word $u \neq \lambda$. A* **suffix of a word w** *over Σ is any word u such that $w = xu$ for some word x over Σ. A* **proper suffix of a word w** *over Σ is any word u such that $w = xu$ for some word $x \neq \lambda$. A* **subword (proper subword) of a word w** *over Σ is any word y such that $w = zyx$ for some words z and x $(z \cdot x \neq \lambda)$. A* **quasi-subword of a word w** *over Σ is any word z such that $w = u_1 z_1 u_2 z_2 \ldots u_k z_k u_{k+1}$ for some $k \in \mathbb{N}$, where $z = z_1 z_2 \ldots z_k$ and $u_1, \ldots, u_{k+1}, z_1, \ldots, z_k$ are words over Σ.*

Now, we define some basic operations on words.

Definition 2.1.2.4 *Let w be a word over Σ. Then*

(i) $w^0 = \lambda$

(ii) $w^{n+1} = w \cdot w^n = w^n w$ *for each* $n \in \mathbb{N}$.

Definition 2.1.2.5 *For any word w over Σ the* **reversal of w**, *denoted w^R, is defined inductively according to the length of w as follows:*

(i) $\lambda^R = \lambda$

(ii) If $w = au$ *for some* $a \in \Sigma$, $u \in \Sigma^*$, *then* $w^R = (au)^R = u^R a$.

We note that the set Σ^* of all words over an alphabet Σ with the operation concatenation is a monoid with λ as its only unit. This monoid is commutative iff the alphabet Σ is a one-letter alphabet ($|\Sigma| = 1$).

Definition 2.1.2.6 *Let Σ be an alphabet. Then, for any $n \in \mathbb{N}$,*

(i) $\Sigma^n = \{x \in \Sigma^* \mid |x| = n\}$, *and*

(ii) $\Sigma^{\leq n} = \{x \in \Sigma^* \mid |x| \leq n\} = \bigcup_{i=0}^{n} \Sigma^i$.

Definition 2.1.2.7 *Given two alphabets Σ and Γ, a* **homomorphism** *is a mapping h from Σ^* to Γ^* such that*

(i) $h(\lambda) = \lambda$, *and*

(ii) $h(uv) = h(u) \cdot h(v)$ *for all* $u, v \in \Sigma^*$.

It should be observed that for defining a homomorphism from Σ^* to Γ^*, it is sufficient to define $h(a)$ for each symbol $a \in \Sigma$.

Now, we define the languages and the operations on languages.

Definition 2.1.2.8 *Let Σ be an alphabet. Each subset $L \subseteq \Sigma^*$ is called a* **language over Σ**.

Since the languages are sets of words the operations union $(A \cup B)$, intersection $(A \cap B)$, difference $(A - B)$, and symmetric difference $(A \oplus B)$ are well understood as operations on sets. Next, we give some typical operations on languages.

Definition 2.1.2.9 *Let A be a language over an alphabet Σ, and B be a language over an alphabet Γ. We define*

(i) $\boldsymbol{h(A)} = \{h(w) \mid w \in A\}$ *for any homomorphism h from Σ^* to Γ^*,*

(ii) $\boldsymbol{A^{\complement_\Sigma}} = \Sigma^* - A$ *is the* **complement of the language A** *according to the alphabet Σ; if $\Sigma = \{0,1\}$, we use the simple notation $\boldsymbol{A^\complement}$,*

(iii) $\boldsymbol{A \cdot B} = \boldsymbol{AB} = \{w \mid w = x \cdot y$ *for some $x \in A$ and some $y \in B\}$ is the* **concatenation of the languages A and B**,

(iv)

$$\begin{aligned} \boldsymbol{B^0} &= \{\lambda\}, \text{ and} \\ \boldsymbol{B^{i+1}} &= B \cdot B^i \text{ for any } i \in \mathbb{N}, \\ \boldsymbol{B^+} &= \bigcup_{i=1}^{\infty} B^i, \text{ and} \\ \boldsymbol{B^*} &= \bigcup_{i \in \mathbb{N}} B^i = B^+ \cup \{\lambda\}, \end{aligned}$$

(v) $\boldsymbol{B[n]} = B \cap \Gamma^n = \{x \in B \mid |x| = n\}$ *for any $n \in \mathbb{N}$ is called the* **n-th level of the language B**.

Definition 2.1.2.10 *Let $\mathcal{L} \subseteq \{L \mid L \subseteq \{0,1,\}^*\}$ be a class of languages. We say that $\mathcal{L}$ is* **closed under complement** *iff for each language $L \in \mathcal{L}$, $L^\complement$ also belongs to $\mathcal{L}$. Let $\odot$ be a binary operation on languages. We say that $\mathcal{L}$* **is closed under** $\odot$ *iff* $\forall L_1, L_2 \in \mathcal{L}, L_1 \odot L_2 \in \mathcal{L}$.

There are only few parts of this book where the reader needs to be familiar with such fundamental notions of formal language theory as finite automata,

pushdown automata, grammars, or language classes of the Chomsky hierarchy. For our considerations, however, formal descriptions are not needed. Nevertheless, we assume that the reader is familiar with the above basic concepts of formal language theory.

2.1.3 Boolean Functions and Boolean Matrices

Boolean functions and Boolean matrices are the most widely used representations of computing problems in this book. First we start with the terminology connected with Boolean functions. More formal and comprehensive terminology can be found in the standard monographs.

Definition 2.1.3.1 *A* **Boolean variable** *is any symbol to which we can associate either of the truth values* 0 *("false") and* 1 *("true"). A* **Boolean function** *of n variables (for some* $n \in \mathbb{N}$*) is any mapping f from* $\{0,1\}^n$ *to* $\{0,1\}$*. Any vector* $\tilde{\alpha} = (\alpha_1, \ldots, \alpha_n)$ *with* $\alpha_i \in \{0,1\}$ *for* $i = 1, \ldots, n$ *is called an* **input of** f*.* $f(\tilde{\alpha})$ *denotes the* **output value of** f **for the input** $\tilde{\alpha}$*.* $B_2^n = \{f \mid f$ *is a boolean function from* $\{0,1\}^n$ *to* $\{0,1\}\}$ *is the* **set of all Boolean functions of** n **variables**. *For any Boolean function f of n variables,* $N^1(f) = \{\tilde{\alpha} \in \{0,1\}^n \mid f(\tilde{\alpha}) = 1\}$ *and* $N^0(f) = \{\tilde{\alpha} \in \{0,1\}^n \mid f(\tilde{\alpha}) = 0\}$*. If the input* $\tilde{\alpha} \in N^1(f)$ *for some* $f \in B_2^n$*, then we say that* $\tilde{\alpha}$ **satisfies** f*.*

Note that $|B_2^n| = 2^{2^n}$, and that $N^0(f) \cup N^1(f) = \{0,1\}^n$ for every $f \in B_2^n$, $n \in \mathbb{N}$.

For any $f \in B_2^n$ for some $n \in \mathbb{N}$ we can name the Boolean variables of f. Thus, "f is a Boolean function of n variables $x_1, x_2, \ldots, x_n$"means that the names of the variables of f are fixed as $x_1, x_2, \ldots, x_n$. Moreover, we frequently use the notation $f(x_1, \ldots, x_n)$ instead of f to underline this fact.

Definition 2.1.3.2 *Let* $f(x_1, \ldots, x_n) \in B_2^n$ *be a Boolean function of n Boolean variables* $x_1, \ldots, x_n$*. We say, for some* $i \in \{1, \ldots, n\}$*, that* $f(x_1, \ldots, x_n)$ **essentially depends on the variable** x_i *iff there exist*

$$\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n \in \{0,1\}$$

such that

$$f(\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n) \neq f(\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n).$$

If $f(x_1, \ldots, x_n)$ *does not essentially depend on a variable* x_j *for some* $j \in \{1, \ldots, n\}$*, then we say that the Boolean variable* x_j **is dummy for** f*.*

For any $n, m \in \mathbb{N}$, $m \leq n$,

$B_2^n(m) = \{f \in B_2^n \mid f$ *essentially depends on at most m variables*$\}$.

Observation 2.1.3.3 *For any* $n, m \in \mathbb{N}$, $m \leq n$,

$$|B_2^n(m)| \leq \binom{n}{m} \cdot 2^{2^m}.$$

Proof. There are exactly $\binom{n}{m}$ ways to choose m variables from n variables, and $|B_2^m| = 2^{2^m}$. □

As described above, the inputs of Boolean function of n variables are considered to be vectors $(\alpha_1, \alpha_2, \ldots, \alpha_n)$. This is also the usual representation of inputs in Boolean function complexity theory. But there are cases where some other representations are more convenient. We shall use two of these representations in order to be able to effectively handle the inputs in distinct situations. We consider inputs as input words which means we use the representation (notation) $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$ instead of $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$. The most common representation of inputs, however, is given in the following definition.

Definition 2.1.3.4 *Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables $x_1, \ldots, x_n$ for some $n \in \mathbb{N}$. An* **input assignment (on X)** *is any mapping α from X to $\{0,1\}$.*

To avoid misunderstandings, we fix the notation of inputs as follows. When the notion "input" is used, then we consider the vector of Boolean values as the input representation. In case of "input word" a word over the alphabet $\{0,1\}$ is considered as the representation of the input. If the notion "input assignment" is used, then the input representation is considered to be a function from the input variables to $\{0,1\}$. Obviously, each of these three input representations is in one-to-one correspondence to the other two. So, we may alternate among these representations (an "input" α may be considered at the same time as an input, an input word, as well as an input assignment) in order to operate on the "input" in a comfortable way. The next definitions outline the possible advantages of input assignments for the input representation.

Definition 2.1.3.5 *Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables for some $n \in \mathbb{N}$, and let $Z = \{x_{i_1}, \ldots, x_{i_m}\} \subseteq X$ be a subset of X. Let α be an input assignment on X and β be an input assignment on Z. We say that β* **preserves α on Z** *iff $\beta(z) = \alpha(z)$ for every $z \in Z$.*

Definition 2.1.3.6 *Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables for some $n \in \mathbb{N}$. Let X_1, X_2 be two subsets of X such that $X_1 \cup X_2 = X$ and $X_1 \cap X_2 = \emptyset$. Let Π be a mapping from X to $\{1,2\}$ with the property $\Pi(x) = 1$ for every $x \in X_1$, and $\Pi(z) = 2$ for every $z \in X_2$. Let α be an input assignment on X_1 and β an input assignment on X_2. Then $\gamma = \Pi^{-1}(\alpha, \beta)$, the* **composition of α and β according to Π**, *denotes the input assignment on X with the property that α preserves γ on X_1 and β preserves γ on X_2.*

We illustrate the above definition by a small example. Let $X_1 = \{x_1, x_3, x_5\}$, and $X_2 = \{x_2, x_4, x_6\}$. Let $\alpha : x_1 \to \{0,1\}$ be defined as $\alpha(x_1) = \alpha(x_5) = 1$, $\alpha(x_3) = 0$ ($\alpha = 101$ in the word representation), and let $\beta : X_2 \to \{0,1\}$ be defined as $\beta(x_2) = \beta(x_6) = 0, \beta(x_4) = 1$ ($\beta = 010$ in the word representation). Then $\gamma = \Pi^{-1}(\alpha, \beta) : X_1 \cup X_2 \to \{0,1\}$ is defined by $\gamma(x_1) = \gamma(x_4) = \gamma(x_5) = 1$, $\gamma(x_2) = \gamma(x_3) = \gamma(x_6) = 0$ ($\gamma = 100110$ in the word representation).

A further problem is how to represent a Boolean function. The simplest possibility is to describe a Boolean function f of n variables as a sequence of 2^n pairs $\{(\alpha_i, f(\alpha_i))\}_{i=1,\ldots,2^n}$, where $\{\alpha_1, \alpha_2, \ldots, \alpha_{2^n}\} = \{0,1\}^n$. In what follows, we always consider α_i to be the i-th input in lexicographical order. Using this assumption f can be unambiguously described as the sequence $\{f(\alpha_i)\}_{i=1,\ldots,2^n}$. Another possibility is to use Boolean matrices.

Definition 2.1.3.7 *A* **Boolean matrix** *of size $m \times k$ ($m, k \in \mathbb{N} - \{0\}$) is any matrix $M = [a_{ij}]_{i=1,\ldots,m,j=1,\ldots,k}$, where $a_{ij} \in \{0,1\}$ for every $i = 1, \ldots, m$ and every $j = 1, \ldots, k$. M is called the* **identity matrix** *of size n, denoted by $I_{n\times n}$, if $n = m = k$, and $a_{ii} = 1$ for $i = 1, \ldots, n$, $a_{ij} = 0$ for $i \neq j$, $i, j \in \{1, \ldots, n\}$. For any $\delta \in \{0,1\}$, M is called a* **δ-monochromatic matrix** *iff $a_{ij} = \delta$ for all i, j. M of size $n \times n$ is called* **symmetric** *iff $a_{ij} = a_{ji}$ for all $i, j \in \{1, \ldots, n\}$.*

Definition 2.1.3.8 *Let f be a Boolean function of n variables in $X = \{x_1, \ldots, x_n\}$. Let $X_1, X_2 \subseteq X$, $X_1 \cup X_2 = X$, $X_1 \cap X_2 = \emptyset$, and let $\Pi(x) = b$ for every $x \in X_b$ ($b = 1, 2$). Π is called a* **partition of X** *(sometimes we write $\Pi = (X_1, X_2)$). Let $|X_1| = r$, $|X_2| = s$. Then the Boolean matrix $M(f, \Pi) = [a_{ij}]_{i=1,\ldots,2^r,j=1,\ldots,2^s}$, where $a_{ij} = f(\Pi^{-1}(\alpha_i, \beta_j))$ for the lexicographically i-th α_i in $\{0,1\}^r$ and the lexicographically j-th β_j in $\{0,1\}^s$, is the* **matrix representation of f according to Π**.

Obviously, both these Boolean function representations require 2^n bits to describe any Boolean function of n variables. It may be also very hard to observe some properties of Boolean functions given in these representations. In several cases, the representation of Boolean functions as Boolean formulas is much shorter and more convenient than the representations given above. So, let us define Boolean formulas now.

Definition 2.1.3.9 *For any two Boolean values α, β from $\{0,1\}$,*

(i) $\alpha \wedge \beta = \begin{cases} 1 & \text{if } \alpha = 1 \text{ and } \beta = 1 \\ 0 & \text{otherwise} \end{cases}$

is the **conjunction of α and β**;

(ii) $\alpha \vee \beta = \begin{cases} 1 & \text{if } \alpha = 1 \text{ or } \beta = 1 \\ 0 & \text{otherwise} \end{cases}$

is the **disjunction of α and β**;

(iii) $\alpha \oplus \beta = \begin{cases} 1 & \text{if } (\alpha = 1 \text{ and } \beta = 0) \text{ or } (\alpha = 0 \text{ and } \beta = 1) \\ 0 & \text{otherwise} \end{cases}$

is the **sum(mod2) of α and β** *(also called* **exclusive or***);*

(iv) $\alpha \equiv \beta = \begin{cases} 1 & \text{if } (\alpha = 1 \text{ and } \beta = 1) \text{ or } (\alpha = 0 \text{ and } \beta = 0) \\ 0 & \text{otherwise} \end{cases}$

is the **equivalence of α and β***;*

(v) *("***α implies β***")*

$\alpha \Rightarrow \beta = \begin{cases} 1 & \text{if } (\alpha = 0) \text{ or } (\alpha = 1 \text{ and } \beta = 1) \\ 0 & \text{otherwise;} \end{cases}$

(vi) $\alpha^{\mathtt{C}} = (\alpha)^{\mathtt{C}} = \begin{cases} 1 & \text{if } \alpha = 0 \\ 0 & \text{if } \alpha = 1 \end{cases}$

is the **negation of α***. Sometimes we also use the denotations* $\Gamma(\alpha)$ *or* $\overline{\alpha}$ *instead of* $\alpha^{\mathtt{C}}$ *or* $(\alpha)^{\mathtt{C}}$.

Observation 2.1.3.10 *For any Boolean values* α *and* β*:*

(i) $\alpha \vee \beta = (\alpha^{\mathtt{C}} \wedge \beta^{\mathtt{C}})^{\mathtt{C}} = \Gamma(\Gamma(\alpha) \wedge \Gamma(\beta))$

(ii) $\alpha \wedge \beta = (\alpha^{\mathtt{C}} \vee \beta^{\mathtt{C}})^{\mathtt{C}} = \Gamma(\Gamma(\alpha) \wedge \Gamma(\beta))$

(iii) $\alpha \oplus \beta = (\alpha \equiv \beta)^{\mathtt{C}} = \Gamma(\alpha \equiv \beta)$.

Definition 2.1.3.11 *Let* X *be a countable set of Boolean variables. The class of* **Boolean formulas over X** *(shortly,* **formulas***) is defined recursively as follows:*

(i) The Boolean values (constants) 0 *and* 1 *are Boolean formulas.*

(ii) For every Boolean variable $x \in X$*,* x *is a Boolean formula.*

(iii) If F *is a Boolean formula, then* $(F)^{\mathtt{C}}$ *is a Boolean formula.*

(iv) If F_1 *and* F_2 *are two Boolean formulas, and* Δ *is an operation in* $\{\vee, \wedge, \oplus, \equiv, \Rightarrow\}$*, then* $(F_1 \Delta F_2)$ *is a Boolean formula.*

(v) Only the expressions constructed by using (i), (ii), (iii) and (iv) are Boolean formulas over X*.*

An example of a Boolean formula is $(((x_1 \vee x_2) \wedge x_7) \Rightarrow (x_2 \equiv x_3))$. Since the operators $\vee, \wedge, \oplus$ and $\equiv$ are commutative and associative we may sometimes omit the brackets. So, for instance, the expressions $x_1 \vee x_2 \vee x_3 \vee x_4$, $((x_1 \vee x_2) \vee (x_3 \vee x_4))$, $(((x_1 \vee x_2) \vee x_3) \vee x_4)$, etc. represent the same Boolean function. We shall also use $\bigvee_{i=1}^{n} x_i$ $[\bigwedge_{i=1}^{n} x_i$, $\bigoplus_{i=1}^{n} x_i]$ instead of $x_1 \vee x_2 \vee \ldots \vee x_n$ $[x_1 \wedge x_2 \wedge \ldots \wedge x_n$, $x_1 \oplus x_2 \oplus \ldots \oplus x_n]$, and $x^{\mathtt{C}}$ instead of $(x)^{\mathtt{C}}$ for any Boolean variable x.

Now we define the Boolean function corresponding to a given Boolean formula.

Definition 2.1.3.12 *Let X be a set of Boolean variables, and let F be a formula over X. Let α be an input assignment on X. Then the* **value of F under the input assignment α**, $F(\alpha)$, *is the Boolean value defined as follows:*

(i) $F(\alpha) = \begin{cases} 0 & \text{if } F = 0 \\ 1 & \text{if } F = 1 \end{cases}$

(ii) $F(\alpha) = \alpha(x)$ *if* $F = x$ *for an* $x \in X$;

(iii) $F(\alpha) = (F_1(\alpha))^{\mathtt{C}}$ *if* $F = (F_1)^{\mathtt{C}}$ *for some formula* F_1;

(iv) $F(\alpha) = F_1(\alpha)\ \Delta\ F_2(\alpha)$ *for some* $\Delta \in \{\wedge, \vee, \equiv, \Rightarrow, \oplus\}$, *if* $F = (F_1\ \Delta\ F_2)$ *for some formulas* F_1 *and* F_2.

Let f be a Boolean function defined on the variables in X. If, for each input assignment β from X to $\{0,1\}$, $f(\beta) = F(\beta)$, then we say that **F represents f**.

Obviously, each formula represents exactly one Boolean function. But one Boolean function can be represented by (infinitely) many formulas. For instance, the formulas $x_1 \vee x_2$, $((x_1)^{\mathtt{C}} \wedge (x_2)^{\mathtt{C}})^{\mathtt{C}}$, $x_1 \vee x_2 \vee x_1 \vee x_2$, and $(x_1 \Rightarrow x_2)^{\mathtt{C}} \vee (x_2 \Rightarrow x_1)^{\mathtt{C}} \vee (x_2 \Rightarrow x_1^{\mathtt{C}})^{\mathtt{C}}$ represent the same Boolean function.

Since we frequently deal with Boolean matrices, especially with their ranks, we give some basic definitions concerning Boolean vectors and matrices in this subsection.

Definition 2.1.3.13 *Let n be a positive integer. The Boolean vector $\tilde{0}^n = (0,0,\ldots,0) \in \{0,1\}^n$ is called the* **zero-vector**, *and the Boolean vector $\tilde{1}^n = (1,1,\ldots,1) \in \{0,1\}^n$ is called the* **one-vector**. *For any two vectors $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \{0,1\}^n$ and $\tilde{\beta} = (\beta_1, \beta_2, \ldots, \beta_n) \in \{0,1\}^n$, $\tilde{\boldsymbol{\alpha}} \oplus \tilde{\boldsymbol{\beta}} = (\alpha_1 \oplus \beta_1, \alpha_2 \oplus \beta_2, \ldots, \alpha_n \oplus \beta_n)$, and $\tilde{\boldsymbol{\alpha}} \leq \tilde{\boldsymbol{\beta}}$ if $\alpha_i \leq \beta_i$ for every $i \in \{1,\ldots,n\}$. For any $d \in \{0,1\}$, and any $\tilde{\gamma} = (\gamma_1, \ldots, \gamma_n) \in \{0,1\}^n$, $\boldsymbol{d} \cdot \tilde{\boldsymbol{\gamma}} = (d \wedge \gamma_1, d \wedge \gamma_2, \ldots, d \wedge \gamma_n)$.*

Definition 2.1.3.14 *Let $\mathcal{M} = \{\tilde{\alpha_1}^n, \tilde{\alpha_2}^n, \ldots, \tilde{\alpha_k}^n\}$ be a set of Boolean vectors on $\{0,1\}^n$. We say that $\mathcal{M}$ is* **linearly independent** *iff*

$$d_1\tilde{\alpha_1}^n \oplus d_2\tilde{\alpha_2}^n \oplus \ldots \oplus d_k\tilde{\alpha_k}^n \neq \tilde{0}^n$$

for every $d_1, d_2, \ldots, d_k \in \{0,1\}, d_i \neq 0$ for some $i \in \{1,\ldots,k\}$.

Definition 2.1.3.15 *Let $M = [a_{ij}]$ be a Boolean matrix of a size $m \times n$ for some positive integers n, m. Let, for $i = 1,\ldots,m$,* $\mathbf{row_i(M)} = (a_{i1}, a_{i2}, \ldots, a_{in}) \in \{0,1\}^n$ *be the Boolean vector corresponding to the i-th row of M. Let, for $j = 1,\ldots,n$,* $\mathbf{col_j(M)} = (a_{1j}, \ldots, a_{nj}) \in \{0,1\}^m$ *be the Boolean vector corresponding to the j-th column of M.*

Let $\mathbf{Row}(M) = \{\text{row}_i(M) \mid i = 1, \ldots, m\}$ *be the set of all rows (vectors) of* M. *Then*

$$\mathbf{rank}(\mathbf{M}) = \max\{|\mathcal{M}| \mid \mathcal{M} \subseteq \mathit{Row}(M) \text{ and } \mathcal{M} \text{ is linearly independent}\}$$

is called the **rank of the matrix** M.

The matrix M *is called* **singular** *iff* $\mathit{rank}(M) < m$ *(i.e.,* $\text{Row}(M)$ *is not an independent set). The matrix* M *is called* **non-singular** *iff* $\text{rank}(M) = m$ *(i.e.,* $\text{Row}(M)$ *is an independent set).*

A trivial example of a non-singular matrix is the identity matrix $I_{n\times n}$ for any $n \in \mathbb{N} - \{0\}$. If n is even, and $I_{n\times n} = M(f, \Pi)$, where Π divides the set of input variables $\{x_1, \ldots, x_n\}$ into the sets $\{x_1, \ldots, x_{n/2}\}$ and $\{x_{n/2+1}, \ldots x_n\}$, then the Boolean function f defined by $M(f, \Pi) = I_{n\times n}$ can be described by the following formula:

$$f(x_1, \ldots, x_n) = \bigwedge_{i=1}^{n/2} (x_i \equiv x_{n/2+i}).$$

Clearly, $\text{rank}(M)$ of a Boolean matrix M defined above is rank of M over the field Z_2. But it is possible to consider ranks of Boolean matrices over any field F with the identity elements 0 and 1. Since we will need it later, we give the following definition.

Definition 2.1.3.16 *Let* M *be a Boolean matrix. For an arbitrary field* F *with identity elements 0 and 1, let* $\mathbf{rank_F(M)}$ *denote the* **rank of the matrix** M **over** F. *We define*

$$\mathbf{Rank(M)} = \max\{\text{rank}_F(M) \mid F \text{ is a field with identity elements 0 and 1}\}.$$

2.1.4 Representation of Computing Problems

In this subsection we explain how computing problems are represented in this book. In general, the representation is based on Boolean functions.

Definition 2.1.4.1 *Let* n, r *be some positive integers. A* **computing problem with** n **inputs and** r **outputs** P_n^r *(or simply, a* **computing problem of size** n*) is a set of* r *Boolean functions* $\{f_1, f_2, \ldots, f_r\}$, *where all Boolean functions in* P_n^r *are defined over the same set* $X = \{x_1, \ldots, x_n\}$ *of* n *input variables.* X *is also called the* **set of input variables of** P_n^r. *Usually, we assign an output variable* y_j *to each Boolean function* f_j *for* $j = 1, \ldots, r$. *The set* $Y = \{y_1, \ldots, y_r\}$ *is called* **the set of output variables of** P_n^r.

Obviously, typical computing problems like sorting, matrix multiplication, language recognition, etc., are computing tasks defined for all infinitely many

input sizes (numbers of input variables), and their complexity is measured as a function depending on the size of the input. We represent such computing tasks as (possibly infinite) sequences of computing problems with fixed sizes.

Definition 2.1.4.2 *Let $J = j_0, j_1, \ldots, j_i, \ldots$ be a (possibly infinite) sequence of positive integers such that $j_q < j_p$ for any $q < p$, $q, p \in \mathbb{N}$. Let φ be a function from $\{j_0, j_1, \ldots, j_i, \ldots\}$ to $\mathbb{N} - \{0\}$. Then each sequence $\mathcal{P} = \{P_s^{\varphi(s)}\}_{s \in J}$ of computing problems $P_s^{\varphi(s)}$ of size s is a* **computing problem.**

To illustrate the above definition, let us consider the sum $\tilde{\alpha} \oplus \tilde{\beta}$ of two Boolean vectors as a computing problem. The formal representation of the problem is $P = \{P_{2n}^n\}_{n=1}^{\infty}$, where $P_{2n}^n = \{f_1, \ldots, f_n\}$ is the computing problem of size $2n$ with the set of input variables $\{x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_n\}$, and $f_i(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_n) = x_i \oplus z_i$ for each $i \in \{1, \ldots, n\}$.

Note. When considering a computing problem $P = \{f\}$ for some $f \in B_2^n$, we omit the brackets. Thus, a Boolean function f (instead of $\{f\}$) will also be considered as a computing problem of size n.

Next, we fix the notation for the computing problems corresponding to the language recognition.

Definition 2.1.4.3 *Let L be a language over the alphabet $\{0,1\}$. Then, for any $n \in \mathbb{N}$, we define $h_n(L)$ to be a Boolean function of n variables such that $h_n(L)(\tilde{\alpha}) = 1$ iff $\alpha \in L[n] = L \cap \{0,1\}^n$ (or such that $N^1(h_n(L)) = L[n]$).*

To illustrate Definition 2.1.4.3 let us consider the language $L = \{ww \mid w \in \{0,1\}^*\}$. The corresponding computing problem representation of the recognition of L is $\{h_n(L)\}_{n=1}^{\infty}$, where

$$\begin{aligned} h_n(L)(x_1, \ldots, x_n) &= \bigwedge_{i=1}^{n/2} (x_i \equiv x_{n/2+i}) \text{ for } n \text{ even, and} \\ h_n(L)(x_1, \ldots, x_n) &= 0 \text{ for } n \text{ odd.} \end{aligned}$$

Among others we also consider some algorithmic problems on graphs. To begin with, we give some basic notions from graph theory.

Definition 2.1.4.4 *An* **undirected graph**, *for short* **graph, $G = (V, E)$** *consists of a finite* **set of vertices (nodes) V** *and a* **set of edges $E \subseteq \{\{u, v\} \mid u, v \in V\}$.** *An undirected* **edge** $\{u, v\}$ *will also be denoted by (u, v) or (v, u).*

Definition 2.1.4.5 *Let $G = (V, E)$ be a graph. Two vertices v, $w \in V$ are* **adjacent** *if $(v, w) \in E$. We say the edge (v, w) is* **incident** *upon the vertices v and w. A* **path** *is a sequence of vertices $v_1, v_2, \ldots, v_n$ from V such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < n$. A path $v_1, v_2, \ldots, v_n$ for $n \geq 2$ is* **simple** *if all vertices on the path are distinct ($|\{v_1, v_2, \ldots, v_n\}| = n$), with the exception that v_1 and v_n may be identical ($v_1 = v_n$ and $|\{v_1, \ldots, v_n\}| = n - 1$). The* **length of**

the path $v_1, \ldots, v_n$ *is* $n-1$ *(the number of edges along the path). A* **(simple) cycle** *in* G *is a path* $v_1, \ldots, v_n$ *of length three or more with* $v_1 = v_n$. *A simple cycle of length* 3 *is called a* **triangle**.

Definition 2.1.4.6 *Let* $G = (V, E)$ *be a graph. Two vertices* u *and* v *in* V **are connected** *iff* $u = v$ *or there exists a (simple) path between* u *and* v *in* G. *We say that* G **is connected**, *if* $\forall u, v \in V$, u *and* v *are connected.* G *is* **cyclic** *if it contains at least one cycle. If* G *does not contain any cycle, then* G *is* **acyclic**. *A connected acyclic graph is called* **tree**.

Observation 2.1.4.7 *Let* $G = (V, E)$ *be a tree. Then* $|V| = |E| + 1$.

Definition 2.1.4.8 *Let* $G = (V, E)$ *be a graph. For each* $v \in V$, *the* **degree of** v, $\mathbf{deg}(v) = |\{(v, u) \mid (v, u) \in E\}|$, *is the number of edges incident to* v. *The* **degree of the graph** G *is* $\mathbf{deg}(G) = \max\{\deg(v) \mid v \in V\}$.

Typical algorithmic problems on graphs include decision problems (or so called yes/no problems). The task is to decide (to give the answer "yes" (1) or "no" (0)) whether a graph G (as the input) has a given property (for instance, whether G is connected, cyclic, acyclic, etc.). In what follows we shall consider graph problems as language recognition problems. To do this, we need to code the graphs as words over the alphabet $\{0, 1\}$.

Definition 2.1.4.9 *Let* $G = (V, E)$ *be a graph, and let* $V = \{v_1, \ldots, v_n\}$ *for some posititive integer* n. *Then the* **adjacency matrix of** G *is*

$$M(G) = [m_{ij}]_{i,j=1,\ldots,n},$$

where

$$m_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

Note: Every adjacency matrix of an undirected graph G is symmetric, because (v_i, v_j) denotes the same edge as (v_j, v_i) in our notation. Note that the notation $\{u, v\}$ for some edge between u and v would be fully correct and unambiguous, but we prefer to denote an edge as a pair of vertices instead as a set of two vertices in what follows.

Since $M(G)$ is symmetric, the fact of whether an edge (v_i, v_j), $i \neq j$, is in G or not is included twice in $M(G)$ as m_{ij} and as m_{ji}. Since we shall not consider graphs having edges (v, v) for some vertex $v \in V$ (only edges between two distinct vertices are allowed), the diagonal elements $m_{ii} = 0$ for $i = 1, \ldots, n$ also represent superfluous information. So, to represent G, it is sufficient to take only the elements m_{ij} for $i < j$, $i, j \in \{1, \ldots, n\}$.

Definition 2.1.4.10 *Let* G *be a graph, and let* $M(G) = [m_{ij}]_{i,j=1,\ldots,n}$ *be the adjacency matrix of* G. *Then* $\mathbf{word}(G) = m_{12}m_{13} \ldots m_{1n}m_{23}m_{24} \ldots m_{2n} \ldots m_{ii+1} \ldots m_{in} \ldots m_{n-2\ n-1}m_{n-2\ n}m_{n-1\ n}$, *determined by the upper-right corner of* $M(G)$, *is called the* **word representation of** $M(G)$.

Obviously, $|\text{word}(G)| = \binom{n}{2}$ for any graph of n vertices, and word(G) does not contain any superfluous information about the existence of edges in G. So, the problem of deciding whether a given graph G contains a triangle can be represented as recognition of the language $L_\Delta = \{\text{word}(G) \mid G \text{ contains a triangle}\}$. For constructing a graph from a word we give the following definition.

Definition 2.1.4.11 *Let $w = w_1 \dots w_m \in \{0,1\}^*$ be a word of length $m = \binom{n}{2}$ for some positive integer n. Let $M(w) = [a_{ij}]_{i,j \in \{1,\dots,n\}}$ be a Boolean matrix defined by*

$$\begin{aligned} a_{ii} &= 0 \text{ for every } i = 1, \dots, n, \text{ and} \\ a_{ij} &= a_{ji} = w_{r(i,j)} \text{ for every } i, j \in \{1, \dots, n\}, i \neq j, \\ \text{where } r(i,j) &= (\sum_{k=1}^{i-1}(n-k)) + (j-i). \end{aligned}$$

The graph G corresponding to the symmetric matrix $M(w)$, denoted by $G(w)$, is called the **graph induced by the word w**.

Using the notation of Definition 2.1.4.11, we can write $L_\Delta = \{\alpha \in \{0,1\}^* \mid |\alpha| = \binom{n}{2}$ for some $n \in \mathbb{N} - \{0\}$, and $G(\alpha)$ contains a triangle$\}$.

We have already noted that we will use three types of representation of Boolean inputs — inputs (Boolean vectors), words, and input assignments. But sometimes we have also to deal with problems defined over integers. To work comfortably with their binary representations, we fix the following notation.

Definition 2.1.4.12 *Let $w = w_1 w_2 \dots w_m$, $w_i \in \{0,1\}$ for $i = 1, \dots, m$. Then the* **integer binary coded by the word w** *is*

$$\mathbf{BIN(w)} = \sum_{i=1}^{m} w_i 2^{m-i}.$$

Let k, j be nonnegative integers, $k \geq \lceil \log_2 j \rceil$. Then $\mathbf{BIN_k^{-1}(j)}$ is a word from $\{0,1\}^k$ such that $\text{BIN}(\text{BIN}_k^{-1}(j)) = j$.

For instance, $\text{BIN}(110) = 6$ and $\text{BIN}_6^1(7) = 000111$.

While undirected graphs are mostly used to describe computing problems here, directed graphs will be used to describe computing devices in Chapters 3 and 4. Next, we give the basic definitions concerning directed graphs.

Definition 2.1.4.13 *A* **directed graph** *$G = (V, E)$ consists of a finite* **set of vertices (nodes) V** *and a* **set of directed edges $E \subseteq V \times V = \{(u,v) \mid u \in V, v \in V\}$**.

Definition 2.1.4.14 *Let $G = (V, E)$ be a directed graph. If $e = (u,v) \in E$, then we say that* **e leads from u to v**. *We also say that* **(u,v) outcomes**

from u *and* **incomes to v**. *A* **directed path** *(from v_1 to v_n) in G is a sequence of vertices $v_1, v_2, \ldots, v_n$ of V such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq n$. A directed path $v_1, v_2, \ldots, v_n$ for $n \geq 2$ is* **simple** *if all vertices of the path are distinct ($|\{v_1, v_2, \ldots, v_n\}| = n$), with the exception that v_1 and v_n may be identical ($v_1 = v_n$ and $|\{v_1, \ldots, v_n\}| = n - 1$). The* **length** *of the directed path $v_1, \ldots, v_n$ is $n-1$. A* **(simple) directed cycle** *in G is a (simple) directed path $v_1, \ldots, v_n$ of length two or more with $v_1 = v_n$.*

Definition 2.1.4.15 *Let $G = (V, E)$ be a directed graph. We say that G is* **cyclic** *if it contains at least one cycle. If G does not contain any cycle, then G is* **acyclic**.

Definition 2.1.4.16 *Let $G = (V, E)$ be a directed graph. For each $v \in V$, the* **outdegree of v, outdeg(v)** $= |\{(v, u) \mid u \in V, (v, u) \in E\}|$, *is the number of edges outcoming from v. The* **indegree of v, indeg(v)** $= |\{(w, v) \mid w \in V, (w, v) \in E\}|$, *is the number of edges incoming to v. For every $v \in V$, the* **degree** *of v is* **deg(v)** $= \text{outdeg}(v) + \text{indeg}(v)$.

We conclude the definition part by fixing the usual notation used in complexity theory to measure the complexity of computing tasks (problems).

Definition 2.1.4.17 *Let f, g, h be arbitrary functions from $\mathbb{N}$ to $\mathbb{N}$. Then* **$f(n) = o(g(n))$** *denotes the fact $\lim_{n\to\infty}(f(n)/g(n)) = 0$.* **$O(f) = \{h \mid h$** *is a function from $\mathbb{N}$ to $\mathbb{N}$, and there exist positive integers c_h, d_h such that, for every $n \geq c_h$, $h(n) \leq d_h \cdot f(n)\}$.* **$\Omega(g) = \{r \mid r$** *is a function from $\mathbb{N}$ to $\mathbb{N}$, and there exist positive integers c_r, d_r such that, for every $n \geq c_r$, $d_r \cdot r(n) \geq g(n)\}$.* **$\Theta(h) = O(h) \cap \Omega(h)$**. *If somebody shows that the complexity of a computing problem P is in $\Theta(h)$ (in $O(h)$ as well as in $\Omega(h)$), then we say that the* **asymptotic complexity of P** *is h.*

In what follows we also write **$f(n) = O(g(n))$ $[f(n) = \Omega(g(n))]$** to denote the fact $f \in O(g)$ $[f \in \Omega(g)]$.

2.1.5 Exercises

The exercises formulated here are either simple, fundamental observations assumed to be known to the reader, or real exercises offering training in the use of the presented formalism.

Exercise 2.1.5.1 *Let $w \in \{0,1\}^n$ for some even $n \in \mathbb{N}$. How many different quasi-subwords of w exist if*

(i) $w = 1^n$

(ii) $w = (01)^{n/2}$

(iii) $w = 0^{n/2}1^{n/2}$ *?*

How many different ways exist to choose a quasi-subword from a word w of length n ?

Exercise 2.1.5.2 *Let $\mathcal{L} = \{L \mid L \subseteq \{0,1\}^*\}$. Prove that the cardinality of $\mathcal{L}$ is equal to the cardinality of the set of all real numbers.*

Exercise 2.1.5.3 * *Let $\overline{B}_2^n(m) = \{f \in B_2^n(m) \mid f$ essentially depends exactly on m variables$\}$. Estimate the cardinality of $\overline{B}_2^n(m)$.*

Exercise 2.1.5.4 *In Definition 2.1.3.5 we have defined "β preserves α" for two input assignments α and β. Consider α and β to be two words over the alphabet $\{0,1\}$, and define equivalently "β preserves α" without using the notion of input assignments.*

Exercise 2.1.5.5 *Let $M = [a_{ij}]_{i,j=1,\ldots,n}$ be the "upper-triangle" Boolean matrix ($a_{ij} = 1$ iff $i < j$). Prove, that* Row(M) *is an independent set.*

Exercise 2.1.5.6 *Prove, that any Boolean function can be expressed as a formula including only the following operations:*

(i) $\vee$, $\wedge$, $\mathbf{C}$

(ii) $\vee$, $\mathbf{C}$

(iii) $\wedge$, $\mathbf{C}$.

Exercise 2.1.5.7 *Find a Boolean function (operation) φ of two variables such that every Boolean function can be expressed as a formula only over the one operation φ.*

Exercise 2.1.5.8 * *Prove that, for sufficiently large n, at least one quarter of all $n \times n$ Boolean matrices are non-singular over Z_2.*

[Hint: Choose n Boolean vectors of size n randomly, and ask for the probability that they form an independent set.]

Exercise 2.1.5.9 *Describe the Boolean functions $h_n(L)$ as formulas for the following language L:*

(i) $L = \{www \mid w \in \{0,1\}^+\}$,

(ii) $L = \{w \in \{0,1\}^* \mid \#_1(w) = 2\}$,

(iii) $L = \{w \in \{0,1\}^* \mid G(w)$ *is a graph containing a triangle*$\}$.

Exercise 2.1.5.10 *Prove that* $G = (V, E)$ *is a tree iff* G *is connected and* $|V| = |E| + 1$.

Exercise 2.1.5.11 *Prove the following facts:*

(i) $g(n) = n^3 - 2n^2 + 7n - 4 \in \Theta(n^3)$.

(ii) $g(n) = n^{\log_2 n} \in O(2^{\sqrt{n}})$ *and* $g(n) \notin O(n^k)$ *for any* $k \in \mathbb{N}$.

(iii) Let t *be a function from* $\mathbb{N}$ *to* $\mathbb{N}$ *defined recursively as follows:*

$t(0) = t(1) = 1$ *and* $t(n) = k \cdot t(n/k) + n$ *for some* $k \in \mathbb{N}$.

Then $t(n) \in O(n \log_2 n)$.

(iv) Let $F(0) = F(1)$ *and* $F(n) = F(n-1) + F(n-2)$ *be a function from* $\mathbb{N}$ *to* $\mathbb{N}$ *defining the Fibonacci sequence. Find an explicitly defined function* f *such that* $F(n) \in \Theta(f)$.

Exercise 2.1.5.12 *Prove or disprove: "For any two functions* f, g, $f \in O(g)$ *iff* $g \in \Omega(f)$*".*

Exercise 2.1.5.13 *Prove the following assertion: Let* A, B, C *be squared Boolean matrices such that* $A = B + C$. *Then* $\text{rank}(A) \leq \text{rank}(B) + \text{rank}(C)$.

Exercise 2.1.5.14 *Give a formal definition of* $\text{rank}_F(M)$ *for any field* F *with identity elements* 0 *and* 1. *(Note that we did it for* $F = Z_2$ *in Definitions 2.1.3.14 and 2.1.3.15.)*

Exercise 2.1.5.15 * *Find a field* F *with identity elements 0 and 1 such that* $\text{rank}_F(M) = \text{Rank}(M)$ *for any Boolean matrix* M.

Exercise 2.1.5.16 *Find a squared Boolean matrix* M *of size* $n \times n$ *such that* M *is singular (i.e.,* $\text{rank}(M) < n$*) over* Z_2*, but non-singular over* Q.

Exercise 2.1.5.17 *Prove the following claim: If a Boolean matrix* M *has* $\text{rank}(M) = d$*, then* M *consists of at most* 2^d *different rows.*

Exercise 2.1.5.18 *Prove, for the field of reals* R *and for the field of rationals* Q*, that*

$$\text{rank}_R(M) = \text{rank}_Q(M) = \text{Rank}(M)$$

for any Boolean matrix M.

Exercise 2.1.5.19 *Let* f *be a Boolean function over a set of input variables* X*, and let* Π *be a partition of* X. *Prove that, for any field with identity elements* 0 *and* 1, $\text{rank}_F(M(f, \Pi))$ *differs from* $\text{rank}_F(M(\overline{f}, \Pi))$ *at most by* 1.

Exercise 2.1.5.20 * *Find a Boolean function f such that* $\text{rank}(M(f,\Pi)) = O(\log_2(\text{rank}_Q(M(f,\Pi))))$ *for some* Π.

Exercise 2.1.5.21 * *Let, for every* $n \in \mathbb{N}$,

$$f_{2n}^{\text{mod}}(x_1,\ldots,x_n,z_1,\ldots,z_n) = \sum_{i=1}^{n}(x_i \wedge z_i) \bmod 2$$

be the inner product function. Let, for every $X_n = \{x_1,\ldots,x_n,z_1,\ldots,z_n\}$, $\Pi_n = (\{x_1,\ldots,x_n\},\{z_1,\ldots,z_n\})$ *be a partition of* X_n. *Prove, for every* $n \in \mathbb{N}$,

(i) $\text{rank}(M(f_{2n}^{\text{mod}},\Pi_n)) = n$, *and*

(ii) $\text{rank}_Q(M(f_{2n}^{\text{mod}},\Pi_n)) = 2^n - 1$.

[Hint: To prove (ii) consider the matrix $M' = 2M(f_{2n}^{\text{mod}},\Pi_n) - J_n$, *where* J_n *denotes the* $2^n \times 2^n$ *matrix with* $J_n[i,j] = 1$ *for all* $i,j \in \{1,\ldots,2^n\}$. M' *is so called Hadamard matrix and it can be showed that* $\text{rank}_Q(M') = 2^n$. *On the other hand, the transformation* $M \to 2 \cdot M - J_n$ *can increase the rank by at most* 1.*]*

Exercise 2.1.5.22 * *Prove that, for almost all Boolean matrices* M *of the size* $m \times m$, $\text{rank}_Q(M) = m$.

2.2 Communication Complexity According to a Fixed Partition

2.2.1 Definitions

The communication complexity in a specific case (the number of bits exchanged between two abstract computers) depends on the given partition of the input variables between the two abstract computers. Different applications of communication complexity require to consider distinct sets of partitions. We start by defining the partitions considered here and the communication complexity according to a given fixed partition. Communication complexity according to a fixed partition is a basic stone which is sufficient for building any kind of communication complexity used here.

Definition 2.2.1.1 *Let* $X = \{x_1,x_2,\ldots,x_n\}$ *be a set of input variables. Any function* $\Pi : X \to \{1,2\}$ *is called a* **partition of** X. $\Pi_{L,X}$ *(for short,* Π_L *if the connection to* X *is clear) denotes the set* $\{x \in X \mid \Pi(x) = 1\}$, *analogously* $\Pi_{R,X}(\Pi_R) = \{x \in X \mid \Pi(x) = 2\}$. *(Obviously* $\Pi_{R,X} \cup \Pi_{L,X} = X$ *and* $\Pi_{R,X} \cap \Pi_{L,X} = \emptyset$.*)*

So, following our abstract model of two communicating computers, the set $\Pi_{L,X}$ corresponds to the set of input variables assigned to the first (left) com-

puter C_I, and $\Pi_{R,X}$ corresponds to the set of input variables assigned to the second (right) computer C_{II}.

Definition 2.2.1.2 *Let Π be a partition of $X = \{x_1, \ldots, x_n\}$. We say that Π is* **balanced** *if $| \, |\Pi_{L,X}| - |\Pi_{R,X}| \, | \leq 1$. We say that Π is* **almost balanced** *if $n/3 \leq |\Pi_{L,X}| \leq 2n/3$ (Obviously, this implies that $n/3 \leq |\Pi_{R,X}| \leq 2n/3$). Let* $\mathbf{Bal(X)} = \{\Pi \mid \Pi$ *is a balanced partition of* $X\}$, *and* $\mathbf{Abal(X)} = \{\Pi \mid \Pi$ *is an almost balanced partition of* $X\}$.

Observation 2.2.1.3 *The number of all partitions of $X = \{x_1, \ldots, x_n\}$ is 2^n. If n is even, then the number of balanced partitions of X is $\binom{n}{n/2}$, and*

$$|\text{Abal}(X) \, | = \sum_{i=\lceil n/3 \rceil}^{\lfloor 2n/3 \rfloor} \binom{n}{i}.$$

The above defined partitions are suitable for the study of one-output problems (Boolean functions), because one can assume without loss of generality that the right computer always produces the output. Even allowing both computers to compute the output (for some inputs the first computer computes the result, for other inputs the second one does so) has no effects on our applications. For many-output problems (the sets of Boolean functions), however, the partition of output variables between the two computers may be essential. So, we extend our definition to partitions of output variables.

Definition 2.2.1.4 *Let $X = \{x_1, \ldots, x_n\}$ be a set of input variables, and let $Y = \{y_1, \ldots, y_r\}$ be a set of output variables. Any function $\Pi : X \cup Y \to \{1, 2\}$ is called a* **partition of X and Y**. $\boldsymbol{\Pi_{L,X}} = \{x \in X \mid \Pi(x) = 1\}$, $\boldsymbol{\Pi_{R,X}} = \{x \in X \mid \Pi(x) = 2\}$, $\boldsymbol{\Pi_{L,Y}} = \{y \in Y \mid \Pi(y) = 1\}$, *and* $\boldsymbol{\Pi_{R,Y}} = \{y \in Y \mid \Pi(y) = 2\}$. *A partition Π of X and Y is* **balanced** *if $||\Pi_{L,X}| - |\Pi_{R,X}|| \leq 1$. Π is called* **almost balanced** *if $n/3 \leq |\Pi_{L,X}| \leq 2n/3$.* $\mathbf{Bal(X, Y)} = \{\Pi \mid \Pi$ *is a balanced partition of X and $Y\}$, and* $\mathbf{Abal(X, Y)} = \{\Pi \mid \Pi$ *is an almost balanced partition of X and $Y\}$.*

Note, that we do not have any requirement on the way in which the set of output variables is partitioned in Definition 2.2.1.4. This means we allow any (unbalanced) partition of the set of output variables.

To allow efficient handling of input assignments partitioned between the two computers, we give the following definition.

Definition 2.2.1.5 *Let Π be a partition of X and Y, $X = \{x_1, \ldots, x_n\}, Y = \{y_1, \ldots, y_r\}$. Let $\alpha : X \to \{0,1\} (\beta : Y \to \{0,1\})$ be an input (output) assignment. We denote by $\boldsymbol{\alpha_{\Pi,L}}$ $(\boldsymbol{\beta_{\Pi,L}})$ an assignment $\alpha_{\Pi,L} : \Pi_{L,X} \to \{0,1\}$ $(\beta_{\Pi,L} : \Pi_{L,Y} \to \{0,1\})$ such that $\alpha_{\Pi,L}$ $(\beta_{\Pi,L})$ preserves α (β). Analogously, $\alpha_{\Pi,R}$ $(\beta_{\Pi,R})$ is the assignment from $\Pi_{R,X}$ $(\Pi_{R,Y})$ to $\{0,1\}$, preserving α (β). We denote (as*

defined in Definition 2.1.3.6) by $\Pi^{-1}(\alpha_{\Pi,L}, \alpha_{\Pi,R})$ *the original assignment* α, *and similarly* $\Pi^{-1}(\beta_{\Pi,L}, \beta_{\Pi,R}) = \beta$.

Now we are ready to define the communication complexity of a computing problem according to a fixed partition.

We start with an informal description of a communication protocol. A protocol for a computing problem P_n^r with a set of input variables X and a set of output variables Y is a pair $D_n = \langle \Pi, \Phi \rangle$, where Π is a partition of X and Y, and Φ is a so-called communication function. The communication function Φ describes the communication between the two abstract computers. The submitted messages are words over the alphabet {0,1}. The computation starts with the submission of a message $\Phi(\alpha_{L,\Pi}, \lambda)$ from the first (left) computer C_I to the second computer C_{II}, where $\alpha_{\Pi,L}$ is the part of the input assignment corresponding to the variables in $\Pi_{L,X}$ and λ denotes the present, empty communication. In the second step, the second computer sends the message $\Phi(\alpha_{\Pi,R}, \Phi(\alpha_{\Pi,L}, \lambda)\$)$ to the first computer, etc. Thus, the arguments of the function Φ are always the corresponding part of the input ($\alpha_{\Pi,L}$ for the first computer, and $\alpha_{\Pi,R}$ for the second computer) and all communication messages exchanged between the two computers (visually separated by the endmarker \$) so far. The computation of the protocol ends when one of the computers produces a word $\Phi(\alpha_{\Pi,A}, c) \in \{\bar{0}, \bar{1}\}^{|\Pi_{A,Y}|}\$$ for an $A \in \{L, R\}$ and some $c \in \{0, 1, \$\}^*$ denoting the present communication. This word is interpreted as the output values for the variables in $\Pi_{A,Y}$ (The symbols $\bar{0}, \bar{1}$ are only used to distinguish them from the communication bits 0,1, but the meaning is the same.), and it is assumed that the other computer gets an empty message $\lambda\$$, and immediately computes the output assignment $\Phi(\alpha_{\Pi,B}, c\$) \in \{\bar{0}, \bar{1}\}^{|\Pi_{Y,B}|}\$$ for the output variables in $\Pi_{Y,B}$, where $B \in \{L, R\}, B \neq A$. In fact this means that after the execution of the communication C_I and C_{II} must know together the whole output.

Now we give a formal definition of a protocol.

Definition 2.2.1.6 *Let* P_n^r *be a computing problem of size* n *with the set* $X = \{x_1, \ldots, x_n\}$ *of input variables and the set* $Y = \{y_1, \ldots, y_r\}$ *of output variables. A* **protocol** *for* X *and* Y *is a pair* $D_n = \langle \Pi, \Phi \rangle$, *where*

(a) Π *is a partition of* X *and* Y,

(b) Φ *is a communication function from*

$$\{0,1\}^m \times \{0,1,\$\}^* \cup \{0,1\}^k \times \{0,1,\$\}^* \text{ to } \{0,1\}^+ \cup \{\bar{0},\bar{1}\}^u\$ \cup \{\bar{0},\bar{1}\}^v\$,$$

where

(i) $m = |\Pi_{L,X}|, k = |\Pi_{R,X}| = n - m, u = |\Pi_{L,Y}|, v = |\Pi_{R,Y}| = r - u$,

(ii) Φ *has the* **prefix-freeness property** *(assuring that the messages exchanged between the two computers are self-delimiting, and no extra "end of transmission" symbol is required), i.e., for each* $c \in \{0,1,\$\}^*$,

and any two different $y, y' \in \{0,1\}^m$ *(*$\{0,1\}^k$*),* $\Phi(y,c)$ *is not a proper prefix of* $\Phi(y',c)$,

(*iii*) Φ *has the property that if one of the computers computes the output values for its output variables, then the other computer gives its output in the next step, i.e.,*
if $\Phi(x,c) \in \{\bar{0},\bar{1}\}^u\$$ *for any* $x \in \{0,1\}^m$ *and some* $c \in \{0,1,\$\}^*$
(if $\Phi(x,c) \in \{\bar{0},\bar{1}\}^v\$$ *for any* $x \in \{0,1\}^k$ *and some* $c \in \{0,1,\$\}^*$*),*
then $\Phi(z,c\$) \in \{\bar{0},\bar{1}\}^v\$$ *for any* $z \in \{0,1\}^k$
$(\Phi(z,c\$) \in \{\bar{0},\bar{1}\}^u\$$ *for any* $z \in \{0,1\}^m$*).*

A **computation of D_n on an input assignment** $\alpha : X \to \{0,1\}$ *is a string* $c = c_1\$c_2\$\ldots\$c_k\$\$c_{k+1}$, *where* $k \geq 0, c_1, \ldots c_{k-1} \in \{0,1\}^+, c_k, c_{k+1} \in \{\bar{0},\bar{1}\}^*$, *and*

(1) if $c_k \in \{\bar{0},\bar{1}\}^u$ *then* $c_{k+1} \in \{\bar{0},\bar{1}\}^v$, *and*
if $c_k \in \{\bar{0},\bar{1}\}^v$ *then* $c_{k+1} \in \{\bar{0},\bar{1}\}^u$, *and*

(2) for each integer l, $0 \leq l \leq k$, *we have*

(2.1) if l *is even, then* $c_{l+1} = \Phi(\alpha_{\Pi,L}, c_1\$c_2\$\ldots\$c_l\$)$

(2.2) if l *is odd, then* $c_{l+1} = \Phi(\alpha_{\Pi,R}, c_1\$c_2\$\ldots\$c_l\$)$.

The **communication of D_n** *corresponding to the computation* $c = c_1\$c_2\$\ldots\$c_k\$\$c_{k+1}$ *is* $\bar{c} = c_1c_2\ldots c_{k-1} \in \{0,1\}^*$.

D_n computes, *for an input* $\beta \in \{0,1\}^n$ *and a number* $j \in \{1,\ldots,r\}$, **the j-th output** $\gamma_j \in \{0,1\}$ *if* y_j *is the* i*-th variable in* $Y_{\Pi,L}$ ($Y_{\Pi,R}$) *and the* i*-th symbol of* $d \in \{\bar{0},\bar{1}\}^u (d \in \{\bar{0},\bar{1}\}^v)$ *is* $\bar{\gamma}_j \in \{\bar{0},\bar{1}\}$ *for a* $d \in \{c_k, c_{k+1}\}$.

We say that **D_n computes**, *for an input* $\beta \in \{0,1\}^n$, **the output $D_n(\beta)$** $= \gamma = \gamma_1\gamma_2\ldots\gamma_r \in \{0,1\}^r$ *if, for each* $j \in \{1,\ldots,r\}$, D_n *computes the output* γ_j *for the input* β *and the number* j. *We say that* **D_n computes P_n^r** *if* $D_n(\beta) = P_n^r(\beta)$ *for any input assignment* β.

The **length of a computation c** *is* $\#_0(c) + \#_1(c)$, *i.e., the total length of all exchanged messages. The* **communication complexity of the protocol $D_n = \langle \Pi, \Phi \rangle$** *is* **$\mathrm{cc}(D_n)$** *– the maximum of the lengths of all computations of* D_n.

The **communication complexity of P_n^r according to a partition Π** *is* $\mathbf{cc(P_n^r, \Pi)} = \min\{\mathrm{cc}(D_n) \mid D_n = \langle \Pi, \Phi \rangle$ *for a* Φ, *and* D_n *computes* $P_n^r\}$, *(i.e., the minimum over all communication functions* Φ*).*

We give an example to illustrate the work of a protocol.

Let $P_n^3 = \{f_1, f_2, f_3\}$ be a computing problem for the set of input variables $X = \{x_1,\ldots,x_m,u_1,\ldots,u_m,v_1,\ldots,v_m\}, m \geq 4$, where

$$f_1(x_1,\ldots,x_m,u_1,\ldots,u_m,v_1,\ldots,v_m) = \bigwedge_{i=1}^{m}(x_i \equiv u_i),$$

$$f_2(x_1,\ldots,x_m,u_1,\ldots,u_m,v_1,\ldots,v_m) = \bigvee_{i=1}^{m}(x_i \vee u_i \vee v_i),$$

$$f_3(x_1, \ldots, x_m, u_1, \ldots, u_m, v_1, \ldots, v_m) = \bigvee_{i=1}^{m}(u_i).$$

Let $Y = \{y_1, y_2, y_3\}$ be the set of output variables, y_i corresponding to the function f_i. Let us consider the following protocol $D_n = \langle \Pi, \Phi \rangle$, $n = 3m$, computing P_n^3:

(a) Π is defined so that $\Pi_{L,X} = \{x_1, \ldots, x_m, v_1, v_2, v_3\}$, $\Pi_{R,X} = \{u_1, \ldots, u_m, v_4, v_5, \ldots, v_m\}$, $\Pi_{L,Y} = \{y_1\}$, and $\Pi_{R,Y} = \{y_2, y_3\}$.

(b) Φ is defined for every input $\alpha_1\alpha_2 \ldots \alpha_m\beta_1\beta_2 \ldots \beta_m\gamma_1\gamma_2 \ldots \gamma_m$ as follows:

$\Phi(\alpha_1\alpha_2 \ldots \alpha_m\gamma_1\gamma_2\gamma_3, \lambda) = c_1 \in \{0,1\}$, where $c_1 = (\bigvee_{i=1}^{m} \alpha_i) \vee \gamma_1 \vee \gamma_2 \vee \gamma_3$

$\Phi(\beta_1 \ldots \beta_m\gamma_4 \ldots \gamma_m, c_1\$) = c_2 \in \{0,1\}^m$, where $c_2 = \beta_1 \ldots \beta_m$;

$\Phi(\alpha_1 \ldots \alpha_m\gamma_1\gamma_2\gamma_3, c_1\$\beta_1 \ldots \beta_m\$) = \bar{1}\$$ if $\bigwedge_{i=1}^{m}(\alpha_i \equiv \beta_i) = 1$ and

$\Phi(\alpha_1 \ldots \alpha_m\gamma_1\gamma_2\gamma_3, c_1\$\beta_1 \ldots \beta_m\$) = \bar{0}\$$ if $\bigwedge_{i=1}^{m}(\alpha_i \equiv \beta_i) = 0$;

$\Phi(\beta_1 \ldots \beta_m\gamma_4 \ldots \gamma_m, c_1\$c_2\$\$) = \bar{a}\bar{b}\$$, where

$a = c_1 \vee (\bigvee_{i=1}^{m} \beta_i) \vee (\bigvee_{j=4}^{m} \gamma_j)$, and $b = \bigvee_{i=1}^{m} \beta_i$.

Obviously, D_n computes P_n^3. The communication complexity of any computation of D_n is exactly $m + 1 = n/3 + 1$. Thus, $\mathrm{cc}(D_n) = n/3 + 1$. Later, we shall show that $n/3$ communication bits are also necessary to compute P_n^3, i.e., that $\mathrm{cc}(P_n^3, \Pi) \geq n/3$. In the previous example one can see different ways of computing the functions f_1, f_2 and f_3. To compute f_3, the second computer does not need any communication, because all variables on which f_3 depends are in $\Pi_{R,X}$. To compute f_2 (the disjunction of all input variables), the first computer computes the disjunction $\bigvee_{i=1}^{m} x_i \vee v_1 \vee v_2 \vee v_3$ over all input variables in $\Pi_{L,X}$, and sends the result c_1 to the second computer. Then, the second computer can finish the computation of f_2 by computing $c_1 \vee (\bigvee_{z \in \Pi_{R,X}} z)$. To compute f_1, one has to check whether $x_i \equiv u_i$ for every $i = 1 \ldots, m$. Since $\{x_1, \ldots, x_m\} \subseteq \Pi_{L,X}$ and $\{u_1, \ldots, u_m\} \subseteq \Pi_{R,X}$, the protocol D_n computes f_1 by sending all actual values of $u_1, \ldots, u_m$ from the second computer to the first computer. After that, the first computer knows the whole input assignment of variables in $\{x_1, \ldots, x_m, u_1, \ldots, u_m\}$ and can compute f_1. Obviously, it is the computation of f_1 causing the high communication complexity of D_n (the submission of n/3 bits). Later we shall show that there exists no strategy to compute f_1 with $\mathrm{cc}(\{f_1\}, \Pi) < m$ if $\{x_1, \ldots, x_m\} \subseteq \Pi_{L,X}$ and $\{u_1, \ldots, u_m\} \subseteq \Pi_{R,X}$.

We observe that to give a formal description of a protocol computing a concrete problem is similar to the situation when one gives a formal description of a Turing machine by defining its transition function. Obviously, this is close to programming in the machine code. As it is usual in the case of Turing machines we shall often prefere to give an informal description of protocols

computing concrete problems rather than to write the exact formal description. This will be more convinient for capturing the idea how to communicate in order to compute the given problem. We need the formal definition of protocols in order to be able to prove lower bounds on the communication complexity of concrete problems (i.e., to prove the non-existence of protocols with bounded communication complexity for the given problems).

We now make some simple observations, showing that the communication complexity of a problem of size n according to a given partition Π cannot be greater than n (Note that one can construct protocols using many more communication bits than n).

Observation 2.2.1.7 *Let P_n^r be a problem of size n with a set of input variables X and a set of output variables Y, and let Π be a partition of X and Y. Then* $cc(P_n^r, \Pi) \leq n$.

Proof. The idea is very simple. Each computer sends all values of its input variables to the other one. When both have the values of all input variables, then each can immediately compute any function defined on these variables. (Formally, we take $D_n = \langle \Pi, \Phi \rangle$, where $\Phi(\alpha_{\Pi,L}, \lambda) = \alpha_{\Pi,L}$, $\Phi(\alpha_{\Pi,R}, \alpha_{\Pi,L}\$) = \alpha_{\Pi,R}$, $\Phi(\alpha_{\Pi,L}, \alpha_{\Pi,L}\$\alpha_{\Pi,R}\$) =$ the values of the output variables in $\Pi_{L,Y}$ for the input α, and $\Phi(\alpha_{\Pi,R}, \alpha_{\Pi,L}\$\alpha_{\Pi,R}\$\$) =$ the values of the output variables in $\Pi_{R,Y}$ for the input α.) □

Generally, the length of the communication can be still decreased if we assume that the whole output is computed by one computer. Obviously, in this case, it is sufficient to provide all input values to this computer.

Observation 2.2.1.8 *Let P_n^r be a problem of size n with a set of input variables X and a set of output variables Y. Let Π be an almost balanced partition with the property $\Pi_{L,Y} = \emptyset$ ($\Pi_{R,Y} = Y$). Then* $cc(P_n^r, \Pi) \leq |\Pi_{L,X}| \leq 2n/3$.

The most investigated communication complexity in the literature is the communication complexity of a Boolean function (i.e., of one-output problems). Furthermore, the theoretical properties of communication complexity are only studied as theoretical properties of a complexity measure of Boolean functions. Therefore we give the precise definition of communication protocols computing a Boolean function.

Definition 2.2.1.9 *Let f_n be a Boolean function of n variables in $X = \{x_1, \ldots, x_n\}$. A* **protocol** *over X is a pair $D_n = \langle \Pi, \Phi \rangle$, where*

(a) Π is a partition of X,

(b) Φ is a communication function from $\{0,1\}^m \times \{0,1,\$\}^* \cup \{0,1\}^k \times \{0,1,\$\}^*$ to $\{0,1\}^+ \cup \{\bar{0}, \bar{1}\}$, where

(i) $m = |\Pi_{L,X}|, k = |\Pi_{R,X}| = n - m$;

(ii) Φ *has the prefix freeness property:*
For each $c \in \{0,1,\$\}^*$ *and any two different* $\alpha, \beta \in \{0,1\}^m$ $(\{0,1\}^k)$, $\Phi(\alpha, c)$ *is not a proper prefix of* $\Phi(\beta, c)$;

(iii) If $\Phi(\alpha, c) \in \{\bar{0}, \bar{1}\}$ *for an* $\alpha \in \{0,1\}^m, c \in (\{0,1\}^+\$)^{2p}$ *for some* $p \in \mathbb{N}$ *(for an* $\alpha \in \{0,1\}^k, c \in (\{0,1\}^+\$)^{2p+1}$*), then for all* $q \in \mathbb{N}, \gamma \in \{0,1\}^k, d \in (\{0,1\}^+\$)^{2q+1}$ *(for all* $q \in \mathbb{N}, \gamma \in \{0,1\}^m, d \in (\{0,1\}^+\$)^{2q}$*),* $\Phi(\gamma, d) \notin \{\bar{0}, \bar{1}\}$ *[this property secures that the output value is always computed by the same computer independently of the input assignment];*

(iv) If $\Phi(\alpha, c) \in \{\bar{0}, \bar{1}\}$ *for an* $\alpha \in \{0,1\}^m$*, then* $\Phi(\beta, c) \notin \{0,1\}^+$ *for any* $\beta \in \{0,1\}^m$ *[this property assures that if computer A computes the output for an input assignment, then computer B knows that A already knows the result (so B does not wait for further communication)].*

A **computation** *of* D_n *on an input assignment* $\alpha \in \{0,1\}^n$ *is a string* $c = c_1\$c_2\$\ldots\$c_k\$c_{k+1}$*, where*

(1) $k \geq 0, c_1, \ldots, c_k \in \{0,1\}^+, c_{k+1} \in \{\bar{0}, \bar{1}\}$;

(2) for each integer l, $0 \leq l \leq k$, *we have*

(2.1) if l *is even, then* $c_{l+1} = \Phi(\alpha_{\Pi,L}, c_1\$c_2\$\ldots\$c_l\$)$

(2.2) if l *is odd, then* $c_{l+1} = \Phi(\alpha_{\Pi,R}, c_1\$c_2\$\ldots\$c_l\$)$.

We say that D_n **computes** $f_n : \{0,1\}^n \to \{0,1\}$ *if, for each* $\alpha \in \{0,1\}^n$*, the computation of* D_n *on the input assignment* α *is finite and ends with* $\bar{1}$ *iff* $f_n(\alpha) = 1$. *In the following, we also say that a computation is* **accepting** *(***unaccepting***) if it ends with* $\bar{1}(\bar{0})$.

The **length of a computation** c *is the total length of all messages in* c *(ignoring \$'s and the final* $\bar{0}, \bar{1}$*). The* **communication complexity of the protocol** D_n**,** $\mathbf{cc}(D_n)$*, is the maximum of all computation lengths of* D_n.

The **communication complexity of** f_n **according to a partition** Π *is* $\mathbf{cc}(f_n, \Pi) = \min\{\mathrm{cc}(D_n) \mid D_n = \langle \Pi, \Phi \rangle$ *for a* Φ*, and* D_n *computing* $f_n\}$.

Now we illustrate this definition by the recognition of the language $\tilde{L} = \{x \in \{0,1\}^+ \mid \#_0(x) = \#_1(x)\}$. We give the protocol $D_{2m} = \langle \overline{\Pi}, \overline{\Phi} \rangle$ computing $h_{2m}(\tilde{L})$ for any $m \in \mathbb{N}$. Let the set of input variables be denoted by $X = \{x_1, \ldots, x_{2m}\}$, and let $k = \lceil \log_2(m+1) \rceil$.

Informally D_{2m} woks as follows. It divides the input into the first halve and second halve. The communication is organized as follows. C_I sends the number of 1's in the first halve of the input to C_{II}. Then C_{II} accepts if the submitted number plus the number of 1's in the second halve of the input is equal to m. The formal description of D_{2m} follows:

(1) $\overline{\Pi} : \overline{\Pi}_{L,X} = \{x_1, \ldots, x_m\}$

(2) $\Phi : \Phi(\alpha_1\alpha_2 \dots \alpha_m, \lambda) = c_1$, where $c_1 = \mathrm{BIN}_k^{-1}(\#_1(\alpha_1 \dots \alpha_m))$;

$$\begin{aligned} \Phi(\alpha_{m+1} \dots \alpha_{2m}, \mathrm{BIN}_k^{-1}(\#_1(\alpha_1 \dots \alpha_m))\$) &= \bar{1} \\ \text{if } \#_1(\alpha_1 \dots \alpha_m) &+ \#_1(\alpha_{m+1} \dots \alpha_{2m}) = m \\ \Phi(\alpha_{m+1} \dots \alpha_{2m}, \mathrm{BIN}_k^{-1}(\#_1(\alpha_1 \dots \alpha_m))\$) &= \bar{0} \\ \text{if } \#_1(\alpha_1 \dots \alpha_m) &+ \#_1(\alpha_{m+1} \dots \alpha_{2m}) \neq m \end{aligned}$$

Obviously, $\mathrm{cc}(D_n) = k = \lceil \log_2(m+1) \rceil \leq \log_2 n$. For all presented examples the communication complexity of each computation (for each input) is the same.

Now we give a further example, where the lengths of computations essentially differ (the "average" computation length is smaller than the maximal length).

Let us consider the language $Sm = \{xy \mid |x| = |y|, x, y \in \{0,1\}^+, \mathrm{BIN}(x) < \mathrm{BIN}(y)\}$. We define $D'_{2m} = \langle \overline{\Pi}, \Phi' \rangle$ computing $h_{2m}(Sm)$ for any $m \in \mathbb{N}$. Let $X = \{x_1, \dots, x_m, y_1, \dots, y_m\}$ be the set of input variables of $h_{2m}(Sm)$.

Informally, for an input $\alpha_1 \dots \alpha_m \beta_1 \dots \beta_m$, D'_{2m} consecutively searches for the smallest i such that $\alpha_i \neq \beta_i$. Obviously, the relation between α_i and β_i determines the relation between $\mathrm{BIN}(\alpha_1 \dots \alpha_m)$ and $\mathrm{BIN}(\beta_1 \dots \beta_m)$. The formal description of D'_{2m} follows.

- $\overline{\Pi} : \overline{\Pi}_{L,X} = \{x_1, \dots, x_m\}, \overline{\Pi}_{R,X} = \{y_1, \dots, y_m\}$.
- Φ':

$$\begin{aligned} \Phi'(\alpha_1 \dots \alpha_m, \lambda) &= \alpha_1; \\ \Phi'(\beta_1 \dots \beta_m, \alpha_1\$) &= 01 \text{ if } \alpha_1 < \beta_1 \\ &= 00 \text{ if } \alpha_1 > \beta_1 \\ &= 1 \text{ if } \alpha_1 = \beta_1; \\ &\vdots \\ \Phi'(\alpha_1 \dots \alpha_m, \alpha_1\$1\$ \dots \alpha_i\$1\$) &= \alpha_{i+1} \quad (i < m); \\ \Phi'(\beta_1 \dots \beta_m, \alpha_1\$1\$ \dots \alpha_i\$1\$\alpha_{i+1}\$) &= 01 \quad \text{if } \alpha_{i+1} < \beta_{i+1} \\ &= 00 \quad \text{if } \alpha_{i+1} > \beta_{i+1} \\ &= 1 \quad \text{if } \alpha_1 = \beta_1; \\ &\vdots \\ \Phi'(\alpha_1 \dots \alpha_m, \alpha_1\$1\$ \dots \alpha_{m-1}\$1\$) &= \alpha_m; \\ \Phi'(\beta_1 \dots \beta_m, \alpha_1\$1\$ \dots \$1\$\alpha_m\$) &= 01 \quad \text{if } \alpha_m < \beta_m \\ &= 00 \quad \text{if } \alpha_m \geq \beta_m \\ \Phi'(\alpha_1 \dots \alpha_m, \alpha_1\$1\$ \dots \alpha_i\$0\gamma\$) &= \overline{\gamma} \in \{\bar{0}, \bar{1}\} \text{ for any } i \in \{1, \dots, m\}. \end{aligned}$$

2.2.2 Methods for Proving Lower Bounds

To apply the communication complexity in real computing models for real computing problems, we need to get lower bounds on communication complexity of

concrete problems for distinct partitions. Hence, the development of methods for proving lower bounds on communication complexity is the crucial point.

The first lower bound method presented here is based on the so-called "crossing sequence argument". The idea is to find a set of inputs $\mathcal{A}$ such that for any two $x, y \in \mathcal{A}$ the communication (the communication between the two computers of the communication protocol) must be different. If one succeeds in finding such a set $\mathcal{A}$ for a given partition Π, any protocol with the partition Π must have at least $|\mathcal{A}|$ different communications, i.e., there must be a communication of length at least $\log_2 |\mathcal{A}|$. We first give an example how to find such a set $\mathcal{A}$ for the recognition of the language $\tilde{L}$, and then formalize this idea.

Example 2.2.2.1 Let $X = \{x_1, \ldots, x_{2m}\}$, $n = 2m$, be the set of input variables of $h_{2m}(\tilde{L})$ for the language $\tilde{L} = \{\alpha \in \{0,1\}^+ \mid \#_0(\alpha) = \#_1(\alpha)\}$. Let $\overline{\Pi}$ be the partition with $\overline{\Pi}_{L,X} = \{x_1, \ldots, x_m\}$. Let $\mathcal{A}(h_{2m}(\tilde{L}), \overline{\Pi}) = \{1^i 0^j 1^j 0^i \mid i, j \in \{0, \ldots, m\}, i+j = m\}$. Obviously, $|\mathcal{A}(h_{2m}(\tilde{L}), \overline{\Pi})| = m+1$. Now, we shall show that every protocol $D_n = \langle \overline{\Pi}, \overline{\Phi} \rangle$, computing $h_{2m}(\tilde{L})$, must have at least m+1 different communications. We prove this by contradiction. Let there be two distinct words $\alpha = 1^a 0^b 1^b 0^a, \beta = 1^e 0^d 1^d 0^e$, $a \neq e$, in $\mathcal{A}(h_{2m}(\tilde{L}), \overline{\Pi})$, having the same communication $\bar{c} \in \{0,1\}^*$. Because of the prefix-freeness property of protocols we get $\Phi(1^a 0^b, \lambda) = \Phi(1^e 0^d, \lambda) = c_1$, where c_1 is a prefix of $\bar{c}$. For the same reason we get $\Phi(1^b 0^a, c_1\$) = \Phi(1^d 0^e, c_1\$) = c_2, \Phi(1^a 0^b, c_1\$c_2\$) = \Phi(1^e 0^d, c_1\$c_2\$) = c_3$, etc. Thus any two inputs from $\tilde{L}$ with the same communication of D_n must also have the same computation of D_n (Note that this generally holds for any computing problem P and any protocol computing P).

Let $c = c_1\$c_2\$ \ldots \$c_k\$\bar{1}$ be the computation of D_n for both inputs α and β. Let us show that c is also the computation for $\gamma = \overline{\Pi}^{-1}(\alpha_{\overline{\Pi},L}, \beta_{\overline{\Pi},R}) = 1^a 0^b 1^d 0^e$. Obviously, this completes the proof, because $\gamma \notin \tilde{L}$ $(h_{2m}(\tilde{L})(\gamma) = 0$ since $a \neq e)$ which contradicts the fact that D_n computes $\bar{1}$ for the input γ.

Since $\gamma_{\overline{\Pi},L} = \alpha_{\overline{\Pi},L}$ and $\gamma_{\overline{\Pi},R} = \beta_{\overline{\Pi},R}$ we get

$$\Phi(\gamma_{\overline{\Pi},L}, \lambda) = \Phi(1^a 0^b, \lambda) = \Phi(\alpha_{\overline{\Pi},L}, \lambda) = c_1,$$

$$\Phi(\gamma_{\overline{\Pi},R}, c_1\$) = \Phi(1^d 0^e, c_1\$) = \Phi(\beta_{\overline{\Pi},R}, c_1\$) = c_2, \ldots$$

$$\Phi(\gamma_{\overline{\Pi},L}, c_1\$ \ldots \$c_k\$) = \Phi(\alpha_{\overline{\Pi},L}, c_1\$ \ldots \$c_k\$) = \bar{1} \text{ if } k \text{ is even, or}$$

$$\Phi(\gamma_{\overline{\Pi},R}, c_1\$ \ldots \$c_k\$) = \Phi(\beta_{\overline{\Pi},R}, c_1\$ \ldots \$c_k\$) = \bar{1} \text{ if } k \text{ is odd.}$$

Thus, $D_n(\gamma) = 1$, i.e., D_n does not compute $h_{2m}(\tilde{L})$.

We have proved that any protocol D_n computing $h_{2m}(\tilde{L})$ for the partition $\overline{\Pi}$ must have at least $m+1$ different communications. Since the communications are words over the alphabet $\{0,1\}$, there is a communication of a length of at least $\lceil \log_2(m+1) \rceil$. So, $\mathrm{cc}(D_n) \geq \lceil \log_2(n/2+1) \rceil$ for any D_n with the partition $\overline{\Pi}$ and any $n \in \mathbb{N}$. We note that this shows that the protocol for computing $h_{2m}(\tilde{L})$ which we gave in Section 2.2.1 is optimal. □

Now we formalize the above idea in order to provide a general method for proving lower bounds.

Definition 2.2.2.2 *Let P_n^r be a computing problem of size n with a set of input variables $X = \{x_1, \ldots, x_n\}$ and a set of output variables $Y = \{y_1, \ldots, y_r\}$. Let Π be a partition of X and Y. Then a* **fooling set $\mathcal{A}(P_n^r, \Pi)$ for P_n^r and Π** *is any set of input assignments from X to $\{0,1\}$ such that for any distinct α and β in $\mathcal{A}(P_n^r, \Pi)$ one of the following four conditions holds:*

(1) $P_n^r(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R}))$ *differs from* $P_n^r(\alpha)$ *on some variable in* $\Pi_{L,Y}$.

(2) $P_n^r(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R}))$ *differs from* $P_n^r(\beta)$ *on some variable in* $\Pi_{R,Y}$.

(3) $P_n^r(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R}))$ *differs from* $P_n^r(\beta)$ *on some variable in* $\Pi_{L,Y}$.

(4) $P_n^r(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R}))$ *differs from* $P_n^r(\alpha)$ *on some variable in* $\Pi_{R,Y}$.

Theorem 2.2.2.3 *Let P_n^r be a computing problem with a set of input variables X, and a set of output variables Y. Let Π be a partition of X and Y. If $\mathcal{A}(P_n^r, \Pi)$ is a fooling set for P_n^r and Π, then*

$$\mathrm{cc}(P_n^r, \Pi) \geq \lceil \log_2 | \mathcal{A}(P_n^r, \Pi) | \rceil.$$

Proof. The idea is to show that any protocol computing P_n^r with the partition Π must have different communications for different input assignments from $\mathcal{A}(P_n^r, \Pi)$. Let there be a protocol $D_n = \langle \Pi, \Phi \rangle$ having the same communication $\bar{c}$ for two distinct inputs $\alpha, \beta \in \mathcal{A}(P_n^r, \Pi)$. If $c_1\$ \ldots \$c_{k-1}\$c_k\$\$c_{k+1}$ is the computation of D_n on α, and $d_1\$ \ldots \$d_{r-1}\$d_r\$\$d_{r+1}$ is the computation of D_n on β, then $\bar{c} = c_1c_2 \ldots c_{k-1} = d_1d_2 \ldots d_{r-1}$ implies that $r = k$, and $c_i = d_i$ for any $i \in \{1, \ldots, k-1\}$. The same communication $\bar{c}$ for α and β implies:

(1′) $P_n^r(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R}))$ agrees with $P_n^r(\alpha) = P_n^r(\Pi^{-1}(\alpha_{\Pi,L}, \alpha_{\Pi,R}))$ on all variables in $\Pi_{L,Y}$, because for both inputs $\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})$ and α the argument of the left computer C_I (Φ), computing the output assignment for the variables in $\Pi_{L,Y}$ is the same (namely, $\alpha_{\Pi,L}$ and $\bar{c}$).

(2′) $P_n^r(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R}))$ agrees with $P_n^r(\beta) = P_n^r(\Pi^{-1}(\beta_{\Pi,L}, \beta_{\Pi,R}))$ on all variables in $\Pi_{R,Y}$, because for both input assignments the right computer C_{II} computing its outputs has the same arguments, namely $\beta_{\Pi,R}$ and $\bar{c}$.

(3′) $P_n^r(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R}))$ agrees with $P_n^r(\beta) = P_n^r(\Pi^{-1}(\beta_{\Pi,L}, \beta_{\Pi,R}))$ on all variables in $\Pi_{L,Y}$, because for both input assignments the left computer C_I computing its outputs has the same arguments $\beta_{\Pi,L}$ and $\bar{c}$.

(4′) $P_n^r(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R}))$ agrees with $P_n^r(\alpha) = P_n^r(\Pi^{-1}(\alpha_{\Pi,L}, \alpha_{\Pi,R}))$ on all variables in $\Pi_{R,Y}$, because for both input assignments the right computer C_{II} computing its outputs has the same arguments $\alpha_{\Pi,R}$ and $\bar{c}$.

Obviously, (1'), (2'), (3'), and (4') together contradict the fact that $\mathcal{A}(P_n^r, \Pi)$ is a fooling set (more precisely, the condition (1) or (2) or (3) or (4) given in the definition of the fooling set). □

So, using Theorem 2.2.2.3, we get the following method for proving lower bounds.

Method foolfix

Input: – A problem P_r^n of size n with a set of input variables X and a set of output variables Y.

– a partition Π of X and Y.

Step 1: Find a fooling set $\mathcal{A}(P_r^n, \Pi)$ for P_r^n and Π.

Step 2: Compute $\alpha = \lceil \log_2 |\mathcal{A}(P_r^n, \Pi)| \rceil$.

Output: "$\mathrm{cc}(P_r^n, \Pi) \geq \alpha$"

Obviously, the method foolfix can be considered as an algorithm, but Step 1 seems to be computationally very hard if one wants to find the largest fooling set. So Step 1 should be examined by researchers investigating the communication complexity of a computing problem. We give one more example to illustrate the use of this method.

Example 2.2.2.4 Let $P_n^2 = \{h_1, h_2\}$ be a computing problem with a set of input variables $X = \{x_1, \ldots, x_m, z_1, \ldots, z_m\}$, $n = 2m$ for an even positive integer m. Let

$$h_1(x_1, \ldots, x_m, z_1, \ldots, z_m) = \bigoplus_{i=1}^{m} (x_i \wedge z_i), \quad \text{and}$$

$$h_2(x_1, \ldots, x_m, z_1, \ldots, z_m) = \bigwedge_{j=1}^{m/2} (x_j \equiv x_{j+m/2}).$$

Let $Y = \{y_1, y_2\}$, and let Π be the partition of X and Y defined by $\Pi_{L,X} = \{x_1, \ldots, x_{m/2}, z_{m/2+1}, \ldots, z_m\}$, $\Pi_{R,X} = \{x_{m/2+1}, \ldots, x_m, z_1, \ldots, z_{m/2}\}$, $\Pi_{L,Y} = \{y_1\}$ and $\Pi_{R,Y} = \{y_2\}$. Now we show that for $\mathcal{A}_1 = \{\alpha_1\alpha_2 \ldots \alpha_{m/2}\alpha_1\alpha_2 \ldots \alpha_{m/2}0^m \mid \alpha_i \in \{0,1\} \text{ for } i = 1, \ldots, m/2\}$ and $\mathcal{A}_2 = \{1^{m+1}0^{m-1}\}$ the set $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ is a fooling set for P_n^2 and Π. Obviously, the one-element set $\mathcal{A}_2$ is a fooling set. First, we show that $\mathcal{A}_1$ is a fooling set. Let $\beta = \beta_1 \ldots \beta_{m/2}\ \beta_1 \ldots \beta_{m/2}0^m$ and $\gamma = \gamma_1 \ldots \gamma_{m/2}\gamma_1 \ldots \gamma_{m/2}0^m$ be two different elements of $\mathcal{A}_1$. This means that there exists an $i \in \{1, \ldots, m/2\}$ such that $\beta_i \neq \gamma_i$.

This implies $h_2(\Pi^{-1}(\gamma_{\Pi,L}, \beta_{\Pi,R})) = h_2(\gamma_1\gamma_2 \ldots \gamma_{m/2}\beta_1\beta_2 \ldots \beta_{m/2}0^m) = 0$. Since $h_2(\beta) = h_2(\Pi^{-1}(\beta_{\Pi,L}, \beta_{\Pi,R})) = 1$, $\Pi^{-1}(\gamma_{\Pi,L}, \beta_{\Pi,R})$ differs from β on the variable y_2 in $\Pi_{R,Y}$, i.e., the condition (2) of Definition 2.2.2.2 is fulfilled.

To complete the proof (of the fact that $\mathcal{A}$ is a fooling set), it remains to show that one of the conditions (1),(2),(3) or (4) of Definition 2.2.2.2 holds for $w = 1^{m+1}0^{m-1}$ and any word $\delta = \delta_1 \dots \delta_{m/2}\delta_1 \dots \delta_{m/2}0^m \in \mathcal{A}_1$. Since $h_1(w) = 1$, and $h_1(\Pi^{-1}(w_{\Pi,L}, \delta_{\Pi,R})) = h_1(1^{m/2}\delta_1 \dots \delta_{m/2}0^{m/2}0^{m/2}) = 0$ for any $\delta_1 \dots \delta_{m/2} \in \{0,1\}^{m/2}$, w differs from $\Pi^{-1}(w_{\Pi,L}, \delta_{\Pi,R})$ on the variable y_1 in $\Pi_{L,Y}$ (condition (1) of Definition 2.2.2.2).

Obviously, $|\mathcal{A}| = |\mathcal{A}_1| + |\mathcal{A}_2| = 2^{m/2} + 1$. Since $\lceil \log_2 |\mathcal{A}| \rceil = m/2 + 1, \mathrm{cc}(P_2^n, \Pi) \geq m/2 + 1$. Note that one can easily show that $\mathrm{cc}(h_2, \Pi) \leq m/2$ which, combined with the fact that $\mathcal{A}_1$ is a fooling set for h_2 and Π, gives $\mathrm{cc}(h_2, \Pi) = m/2$. □

The lower bound proof technique introduced above is sometimes not very transparent or elegant. For this reason we present two further methods for proving lower bounds on communication complexity of one-output problems (Boolean functions). One of these methods is computationally efficient. To introduce them, we need the following model of the representation of Boolean functions. Note that the matrix $M(f, \Pi)$, defined in the Definition 2.2.2.5 below, is the same object as $M(f, \Pi)$ defined in Definition 2.1.3.8; only the formal description of the matrix elements is given differently.

Definition 2.2.2.5 *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function with a set of input variables $X = \{x_1, \dots, x_n\}$. Let Π be an almost balanced partition of X, with $|\Pi_{L,X}| = m$. Then we define the $2^m \times 2^{n-m}$* **Boolean matrix $M(f, \Pi)$** $= [\alpha_{ij}]_{i=1,\dots,2^m; j=1,\dots,2^{n-m}}$, *where $a_{ij} = f(\Pi^{-1}(\mathrm{BIN}_m^{-1}(i-1), \mathrm{BIN}_{n-m}^{-1}(j-1)))$.*

Obviously, the matrix $M(f, \Pi)$ precisely defines the function f (containing the output values for all 2^n input assignments). As an illustration, let us consider the matrix $M(f, \Pi)$ in Figure 2.1 for $f(x_1, x_2, x_3, x_4) = (x_1 \oplus x_3) \vee (x_2 \oplus x_4)$ and $\Pi_{L,X} = \{x_1, x_3\}$.

			0	1	2	3	
			0	0	1	1	x_2
			0	1	0	1	x_4
0	0	0	0	1	1	0	
1	0	1	1	1	1	1	
2	1	0	1	1	1	1	
3	1	1	0	1	1	0	
	x_1	x_3					

Fig. 2.1.

Definition 2.2.2.6 *Let M be a $k \times l$ Boolean matrix whose rows $1, \dots, k$ are labelled by $i_1, \dots, i_k$ for some positive integers $i_1 < i_2 < \dots < i_k$, and*

whose columns $1, \ldots, l$ *are labelled by* $j_1, \ldots, j_l$ *for some positive integers* $j_1 < j_2 < \ldots < j_l$. *Let* $S_1 \subseteq \{i_1, \ldots, i_k\}$ *and* $S_2 \subseteq \{j_1, \ldots, j_l\}$. *A* **row-split of M according to S_1** *is the partition of* M *into two matrices* $M(S_1)$ *of size* $|S_1| \times l$ *and* $M(S_1^C)$ *of size* $(k - |S_1|) \times l$, *where* $M(S_1)$ *contains all* $|S_1|$ *rows of* M *labelled by numbers of* S_1, *and* $M(S_1^C)$ *contains all* $k - |S_1|$ *rows (the remaining rows) of* M *labelled by numbers of* $\{i_1, \ldots, i_k\} - S_1$. *A* **column-split of M according to S_2** *is the partition of* M *into two matrices* $M(S_2)$ *of size* $k \times |S_2|$ *and* $M(S_2^C)$ *of size* $k \times (l - |S_2|)$, *where* $M(S_2)$ *contains all* $|S_2|$ *columns of* M *labelled by numbers of* S_2, *and* $M(S_2^C)$ *contains all* $l - |S_2|$ *columns of* M *labelled by numbers of* $\{j_1, \ldots, j_l\} - S_2$.

Figure 2.2 depicts two matrices obtained by split operations from the matrix $M = M(f, \Pi)$ of Figure 2.1. The left matrix is $M(\{0, 2, 3\})$ for the row-split and the right one is $M(\{0, 3\})$ for the column-split.

	0	1	2	3
0	0	1	1	0
2	1	1	1	1
3	0	1	1	0

	0	3
0	0	0
1	1	1
2	1	1
3	0	0

Fig. 2.2.

Now we show that, for each protocol $D_n = \langle \Pi, \Phi \rangle$ computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$, one can assign a sequence of m splits to each computation c of length m. This can be done by the following observation. Because both abstract processors of D_n have unbounded power, one can assume that each of them knows the whole matrix $M(f, \Pi) = [a_{rs}]$ and the complete definition of Φ. Obviously, the left computer C_I knows the number i of the row corresponding to its part of the input, and the right computer C_{II} knows the number j of the column corresponding to its part of the input. (Thus, if the left computer learns the number j, then the left computer knows the output a_{ij}. Similarly, if the right computer learns the number i, then the right computer knows the output a_{ij}.)

Let $\bar{c} = c_1 c_2 \ldots c_k$ for $c_i \in \{0,1\}, i = 1, \ldots, k$, be the communication of D_n for a given input. Let us consider the communication part of the computation of D_n as a sequence of k simple steps, where one bit is sent in one step from one computer to the other. First, the left computer sends the bit c_1 to the right computer. This step corresponds to the row-split according to a set $S(c_1)$, where $S(c_1) = \{r \in \{0, \ldots, 2^m - 1\} \mid \Phi(\mathrm{BIN}_m^{-1}(r), \lambda) = c_1 z \text{ for some } z \in \{0,1\}^*\}$, for $m = |\Pi_{L,X}|$. Set $M(c_1) = M(S(c_1))$. Next, the submision of the bit c_2 determines a split of $M(c_1)$. This split is the row–split if the bit c_2 is submitted from the left computer to the right one, and it is the column-split if c_2 is submitted from the left computer to the right one.

The matrix $M(c_1)$ describes the situation in which the left computer knows nothing about the input of the right computer, and the right computer knows that the input of the left computer is in $\{\mathrm{BIN}_m^{-1}(r) \mid r \in S(c_1)\}$.

Let us describe the general situation after the exchange of i bits $c_1c_2 \dots c_i$ for $1 \leq i < k$. Let $M(c_1c_2 \dots c_i)$ be the matrix obtained after the corresponding i splits. Let $S_L(c_1c_2 \dots c_i)$ be the set of all labels of rows in $M(c_1c_2 \dots c_i)$ and $S_R(c_1c_2 \dots c_i)$ be the set of all labels of columns of $M(c_1c_2 \dots c_i)$. This describes the situation when the left computer knows that the input of the right computer is in $\{\mathrm{BIN}_{n-m}^{-1}(s) \mid s \in S_R(c_1 \dots c_i)\}$ and the right computer knows that the input of the left computer is in $\{\mathrm{BIN}_m^{-1}(r) \mid r \in S_L(c_1 \dots c_i)\}$.

Next, the bit c_{i+1} is submitted. Which computer sends (and which one receives) is unambiguously given for the input assignments in $I(c_1 \dots c_i) = \{\Pi^{-1}(\mathrm{BIN}_m^{-1}(r), \mathrm{BIN}_{n-m}^{-1}(s)) \mid r \in S_L(c_1 \dots c_i), s \in S_R(c_1 \dots c_i)\}$ (corresponding to the matrix $M(c_1c_2 \dots c_i)$) because of the prefix-freeness property of Φ. [If this is not clear, then we can prove it by contradiction. Let $\alpha = \Pi^{-1}(\mathrm{BIN}_m^{-1}(r_1), \mathrm{BIN}_{n-m}^{-1}(s_1))$ and $\beta = \Pi^{-1}(\mathrm{BIN}_m^{-1}(r_2), \mathrm{BIN}_{n-m}^{-1}(s_2))$ be two different inputs in $I(c_1 \dots c_i)$, where the $(i+1)$-th bit flows from the left computer to the right computer in the computation on α, and the $(i+1)$-th bit flows from the right computer to the left computer in the computation on β. Without loss of generality we may assume that c_i was submitted by the left computer. For β, this means that the initial part of the computation is $c_1 \dots c_{j_1}\$ \dots \$c_{j_d} \dots c_i\$c_{i+1} \dots$ for some $j_1, \dots, j_d$ and, for α, this means that the initial part of the computation is $c_1 \dots c_{j_1}\$ \dots \$c_{jd} \dots c_ic_{i+1} \dots$. Since $c_{j_d} \dots c_i$ is a proper prefix of $c_{j_d} \dots c_ic_{i+1}$, the prefix-freeness property is violated.]

Let the bit c_{i+1} be submitted by the left computer. Then, we make the row-split of $M(c_1 \dots c_i)$ according to the set $S_L(c_1c_2 \dots c_{i+1}) = \{r \in S_L(c_1c_2 \dots c_i) \mid \Phi$ produces c_{i+1} for the arguments $\mathrm{BIN}_m^{-1}(r)$ and the recent communication $c_1 \dots c_i\}$, and we set $M(c_1 \dots c_{i+1}) = M(S_L(c_1c_2 \dots c_{i+1}))$. If the bit c_{i+1} is submitted from the right computer, then we make the column-split of $M(c_1 \dots c_i)$ according to the set $S_R(c_1c_2 \dots c_{i+1}) = \{s \in S_R(c_1c_2 \dots c_i) \mid \Phi$ produces c_{i+1} for the arguments $\mathrm{BIN}_{n-m}^{-1}(r)$ and the recent communication $c_1 \dots c_i\}$, and we set $M(c_1 \dots c_{i+1}) = M(S_R(c_1 \dots c_{i+1}))$.

Now let us have a closer look at the matrix $M(c_1 \dots c_k)$ which must describe a situation where at least one computer is able to know the output value. First, let us assume the output is computed by the left computer. Since C_I knows the result, the row in $M(c_1 \dots c_k)$, corresponding to the actual input α of C_I, must either contain only 0's or only 1's. Since $\Phi(\alpha, c_1 \dots \$ \dots c_k\$) \in \{\bar{0}, \bar{1}\}$, we have $\Phi(\beta, c_1 \dots \$ \dots c_k\$) \in \{\bar{0}, \bar{1}\}$ for all $\beta \in \{\mathrm{BIN}_m^{-1}(r) \mid r \in S_L(c_1 \dots c_k)\}$ (see the property (iv) of Definition 2.2.1.9). This implies that all other rows of $M(c_1 \dots c_k)$ also consist either only of 0's or only of 1's (are monochromatic).

Similarly, if the output is computed by the second computer, then each column of $M(c_1 \dots c_k)$ consists either only of 0's or only of 1's. Thus, in both cases, the rank of $M(c_1 \dots c_k)$ is one. Now we are ready to formulate the next result providing a lower bound proof technique for communication complexity.

Theorem 2.2.2.7 *For any Boolean function f with a set X of input variables, and for any partition Π of X*

$$cc(f, \Pi) \geq \lceil \log_2(\text{Rank}(f, \Pi)) \rceil.$$

Proof. Let $D_n = \langle \Pi, \Phi \rangle$ be any protocol computing f. Let F be an arbitrary field with identity elements 0 and 1. At the beginning of the computation, the first bit submitted by the left computer to the right one is either 0 or 1, depending on the input assignment for $\Pi_{L,X}$. Let $M(0)$ and $M(1)$ be the matrices obtained from $M(f, \Pi)$ by the row-split according to $S(0)$. Obviously, at least one of the matrices $M(0)$ and $M(1)$ must have a rank of at least $\lceil \text{rank}_F(M(f, \Pi)/2) \rceil$. Let $M(d_1)$ for some $d_1 \in \{0, 1\}$ have this property. Continuing with the split of $M(d_1)$, at least one of the matrices $M(d_1 0)$ and $M(d_1 1)$ must have a rank of at least $\lceil \text{rank}_F(M(d_1))/2 \rceil \geq \lceil \text{rank}_F(M(f, \Pi)/4) \rceil$. Thus, after k splits, we always find a matrix $M(d_1 \ldots d_k)$ such that $\text{rank}_F(M(d_1 \ldots d_k)) \geq \lceil \text{rank}_F(M(f, \Pi))/2^k \rceil$. Since any computation of D_n can halt and compute the output value only after a communication corresponding to a matrix with rank 1, $\log_2(\text{rank}_F(M(f, \Pi)))$ communication bits (splits) are necessary to get a matrix of rank 1. Since we have proved this for every field F, the assertion of Theorem 2.2.2.7 holds. □

Theorem 2.2.2.7 provides the following method for proving lower bounds. Note, that we consider two versions of this method. One computing the rank over Z_2 and one searching for a field F such that the rank over F is as large as possible.

Method rankfix

Input: A matrix $M(f, \Pi)$ for a Boolean function f with a set of input variables X, and a partition Π of X.

Procedure: Compute $d = \text{rank}(M(f, \Pi))$
(for instance, by using the elimination method of Gauss)
[or choose a field F with identity elements 0 and 1 and compute $d = \text{rank}_F(M(f, \Pi))$]

Output: "$cc(f, \Pi) \geq \lceil \log_2 d \rceil$"

Obviously, from the viewpoint of computational complexity, the rankfix method running in $O(n^3)$ is much more efficient that the foolfix method requiring exponential time if the maximal fooling set is selected from among all subsets of the set of input assignments. We show in the following examples that lower bound proofs using the rankfix method can be easier (or more transparent) than those using fooling sets.

Example 2.2.2.8 Equality problem
Let $Eq = \{w \in \{0, 1\}^* \mid w = uu\}$. Let, for $n = 2m, X = \{x_1, \ldots, x_m, z_1, \ldots, z_m\}$

be the set of input variables of $h_n(Eq) = \bigwedge_{i=1}^{m}(x_i \equiv z_i)$. Let us consider the balanced partition $\overline{\Pi}$ with the property $\overline{\Pi}_{L,X} = \{x_1, \ldots, x_m\}$. Since the matrix $M(h_n(Eq), \overline{\Pi})$ is the identity matrix of size $2^m \times 2^m$, $\text{rank}(M(h_n(Eq), \overline{\Pi})) = 2^m$. So, $\text{cc}(h_n(Eq), \overline{\Pi}) = n/2$ for any even n. □

Example 2.2.2.9 Inequality problem
Let $Un = \{xy \in \{0,1\}^* \mid |x| = |y| \text{ and } x \neq y\}$. Let, for $n = 2m, X = \{x_1, \ldots, x_m, z_1, \ldots, z_m\}$ be the set of input variables of $h_n(Un) = \bigvee_{i=1}^{m}(x_i \oplus z_i)$, and let $\overline{\Pi}$ be the balanced partition of Example 2.2.2.8. Since the only zero elements of $M(h_n(Un), \overline{\Pi})$ are the diagonal elements and the sizes of $M(h_n(Un), \overline{\Pi})$ are even, $M(h_n(Un, \overline{\Pi})$ is non-singular. So, $\text{cc}(h_n(Un), \overline{\Pi}) = \log_2(2^m) = n/2$. □

Example 2.2.2.10 Comparison problem
Let Com=$\{uv \in \{0,1\}^* \mid |u| = |v| \text{ and } \text{BIN}^{-1}(u) \geq \text{BIN}^{-1}(v)\}$, and let X and $\overline{\Pi}$ be defined as in the previous example. If the input assignments to $\{x_1, \ldots, x_m\}$ (and also to $\{z_1, \ldots, z_m\}$) are sorted according to the positive integers coded by them, then $M(h_n(Com), \overline{\Pi})$ is the upper triangular matrix which is clearly non-singular. So, $\text{cc}(h_n(Com), \overline{\Pi}) \geq n/2$ for any even positive integer n. □

Thus, one can construct numerous examples for which the rankfix method directly provides high lower bounds.

The third and last lower bound proof technique presented here is also based on the investigation of the matrices $M(f, \Pi)$. Let f be a Boolean function defined on a set X of Boolean variables, and let $\Pi \in \text{Abal}(X)$. Let $D = \langle \Pi, \Phi \rangle$ be a protocol computing f. We have already observed that if c is the same accepting (unaccepting) computation of D on two different inputs α and β, then c is also the accepting (unaccepting) computation of D on inputs $\Pi^{-1}(\alpha_L, \beta_R)$ and $\Pi^{-1}(\beta_L, \alpha_R)$, i.e., $f(\alpha) = f(\beta) = f(\Pi^{-1}(\alpha_L, \beta_R)) = f(\Pi^{-1}(\beta_L, \alpha_R))$. For the matrix $M(f, \Pi) = [a_{ij}]$, this means that $f(\alpha) = a_{rs}$ and $f(\beta) = a_{k\ell}$ imply $a_{rs} = a_{kl} = a_{r\ell} = a_{sk}$. Thus, all inputs having the computation c form a submatrix $M(c)$ whose elements are the same (note that a **submatrix** of $M(f, \Pi)$ is any intersection of a subset of rows of $M(f, \Pi)$ with a subset of columns of $M(f, \Pi)$). Let us call such submatrices **monochromatic submatrices**. Then each computation c of D corresponds exactly to one monochromatic submatrix $M(c)$, and since D is deterministic (we have exactly one computation of D for every input), $M(c_1)$ and $M(c_2)$ are disjoint for any two different computations c_1 and c_2. Since D computes f, the monochromatic submatrices determined by computations of D cover the whole matrix $M(f, \Pi)$. Thus, D having m distinct computations determines a covering of $M(f, \Pi)$ by m disjoint monochromatic submatrices. This means that the cardinality of a minimal cover of $M(f, \Pi)$ by disjoint monochromatic submatrices of $M(f, \Pi)$ directly provides a lower

bound on the number of different computations of every protocol D computing f.

We now formally describe the lower bound method introduced above.

Definition 2.2.2.11 *Let M be a Boolean matrix and let $S = \{M_1, \ldots, M_k\}$ be a set of monochromatic submatrices of M. We say that* $\boldsymbol{S}$ **is a cover of** $\boldsymbol{M}$ *if for each element a_{ij} of M there exists an $m \in \{1, \ldots, k\}$ such that a_{ij} is an element of M_m. We say that* $\boldsymbol{S}$ **is an exact cover of** $\boldsymbol{M}$ *if S is a cover of M and $M_r \cap M_s = \emptyset$ for every $r \neq s$, $r, s, \in \{1, \ldots, k\}$. The* **tiling complexity of** $\boldsymbol{M}$, **Til(M)**, *is*

$$\min\{|S| \mid S \text{ is an exact cover of } M\}.$$

Thus, using the formalism of Definition 2.2.2.11, we have established the following result.

Theorem 2.2.2.12 *For every Boolean function f defined over a set of variables X, and every partition Π of X,*

$$\mathrm{cc}(f, \Pi) \geq \lceil \log_2 \mathrm{Til}(M(f, \Pi)) \rceil - 1.$$

Proof. We have shown above that every protocol $D = \langle \Pi, \Phi \rangle$ computing f must have at least $\mathrm{Til}(M(f, \Pi))$ different computations, i.e., at least $\mathrm{Til}(M(f, \Pi))/2$ different communications. □

Method tilingfix

Input: A matrix $M(f, \Pi)$ for a Boolean function f with a set of input variables X, and a partition Π of X

Procedure: Compute $d = \mathrm{Til}(M(f, \Pi))$

Output: "$\mathrm{cc}(f, \Pi) \geq \lceil \log_2 d \rceil - 1$ "

Note that, as in the case of the foolfix method, we do not know any efficient algorithm computing the procedure of tilingfix, and so estimating $\mathrm{Til}(M(f, \Pi))$ is a research problem for anyone using this method to obtain a lower bound on $\mathrm{cc}(f, \Pi)$.

We illustrate the use of this method by an example. Note that Examples 2.2.2.8, 2.2.2.9, and 2.2.2.10 are also examples applying the tilingfix method, because it can easily be observed that $\mathrm{Til}(h_n(Eq), \overline{\Pi}) = \mathrm{Til}(h_n(Un), \overline{\Pi}) \geq 2^m + 1$, and $\mathrm{Til}(h_n(Com), \overline{\Pi}) \geq 2^{m+1}$.

Example 2.2.2.13 Let $f_{2m}^{\mathrm{mod}} : \{0,1\}^{2m} \to \{0,1\}$ be a Boolean function defined on variables $x_1, \ldots, x_m, z_1, \ldots, z_m$ as follows:

$$f_{2m}^{\mathrm{mod}}(x_1, \ldots, x_m, z_1, \ldots, z_m) = (x_1 \wedge z_1) \oplus (x_2 \wedge z_2) \oplus \ldots \oplus (x_m \wedge z_m).$$

Let $\overline{\Pi} \in \mathrm{Bal}(X)$ be defined by $\overline{\Pi}_L = \{x_1, \ldots, x_m\}$. It is not hard to observe (we leave this to the reader as a combinatorial exercise) that the largest monochromatic submatrix of $M(f_{2m}^{\mathrm{mod}}, \overline{\Pi})$ has 2^m elements. Since $M(f_{2m}^{\mathrm{mod}}, \overline{\Pi})$ has 2^{2m} elements, $\mathrm{Til}(M(f_{2m}^{\mathrm{mod}}, \overline{\Pi})) \geq 2^m$. Thus $\mathrm{cc}(f_{2m}^{\mathrm{mod}}, \overline{\Pi}) \geq m - 1$. □

Now an obvious question appears. We have introduced three different lower bound proof methods. Which relationships exist between them (which provides the highest lower bounds)? How large may be the difference between $\mathrm{cc}(f, \Pi)$ and the lower bounds provided by our lower bound methods?

Since we have considered fooling sets for general computing problems and the methods rankfix and tilingfix have been introduced only for one-output problems we define a simplified version of fooling sets for one-output problems (Boolean functions) before comparing the methods. Note that the main simplification lies in the fact that we do not need to consider the partition of the output variables (for all inputs the same computer computes the output bit).

Definition 2.2.2.14 *Let f_n be a Boolean function over a set of input variables $X = \{x_1, \ldots, x_n\}$. Let Π be an almost balanced partition of X and let $\sigma \in \{0,1\}$. Then, a* **σ-fooling set $\mathcal{A}$ (f_n, Π) for f_n and Π** *is any set of input assignments from X to $\{0,1\}$ such that*

(i) $\forall \alpha \in \mathcal{A}(f_n, \Pi) : f_n(\alpha) = \sigma$

(ii) $\forall \beta, \gamma \in \mathcal{A}(f_n, \Pi), \beta \neq \gamma :$
$f_n(\Pi^{-1}(\beta_{\Pi,L}, \gamma_{\Pi,R})) \neq \sigma$ *or* $f_n(\Pi^{-1}(\gamma_{\Pi,L}, \beta_{\Pi,R})) \neq \sigma$.

(We observe that if $\mathcal{A}$ is a δ-fooling set for some f and Π, then $\mathcal{A}$ is a $\overline{\delta}$-fooling set for $\Gamma(f)$ and Π.)

We define

$$\mathbf{Fool}(f_n, \Pi) = \max\{|\mathcal{A}| \mid \mathcal{A} \text{ is a } \sigma\text{-fooling set for } f_n \text{ and } \Pi, \sigma \in \{0,1\}\}.$$

Obviously, Definition 2.2.2.14 corresponds to the crossing sequence argument of the fooling set method (i.e., each σ-fooling set is a fooling set). If two input assignments $\beta, \gamma \in \mathcal{A}(f_n, \Pi)$ have the same communication (computation), then the protocol has the same communication (computation) for the input assignments $\Pi^{-1}(\beta_{\Pi,L}, \gamma_{\Pi,R})$ and $\Pi^{-1}(\gamma_{\Pi,L}, \beta_{\Pi,R})$. Thus, if $f_n(\Pi^{-1}(\beta_{\Pi,L}, \gamma_{\Pi,R})) \neq \sigma$ or $f_n(\Pi^{-1}(\gamma_{\Pi,L}, \beta_{\Pi,R})) \neq \sigma$, then the communication on the input β must differ from that on the input γ.

In what follows, we set $\mathbf{Rank}(f, \Pi)$=$\mathrm{Rank}(M(f, \Pi))$, $\mathbf{rank}(f, \Pi) = \mathrm{rank}(M(f, \Pi))$, and $\mathbf{Til}(f, \Pi)$=$\mathrm{Til}(M(f, \Pi))$ for any f, Π.

Now, we start to compare $\mathrm{Fool}(f, \Pi)$, $\mathrm{Rank}(f, \Pi)$, $\mathrm{rank}(f, \Pi)$ and $\mathrm{Til}(f, \Pi)$ each to each other, and to compare their logarithms to $\mathrm{cc}(f, \Pi)$. We shall show the following main results:

(i) $\max\{\mathrm{Rank}(f, \Pi), \mathrm{Fool}(f, \Pi)\} \leq \mathrm{Til}(f, \Pi)$ for any f and Π, and $\log_2(\mathrm{Til}(f, \Pi))$ is polynomially related to $\mathrm{cc}(f, \Pi)$ for any f and Π.

(ii) $\text{Til}(f, \Pi) \leq 2^{\text{rank}(f,\Pi)+1}$ for any f and Π and there exists a Boolean function g with an exponential gap between $\text{Til}(g, \Pi)$ and $\text{rank}(g, \Pi)$ (i.e., for some Boolean functions the method tilingfix can provide essentially higher lower bounds than the weaker version of the rankfix method).

(iii) There exists a Boolean function h such that the gap between $\text{Til}(h, \Pi)$ and $\text{Fool}(h, \Pi)$ is exponential (i.e., for some problems the foolfix method is very weak).

(iv) $\text{Fool}(f, \Pi) \leq (\text{rank}(f, \Pi)+2)^2$ for any f and Π, and there exists a Boolean function g with an exponential gap between $\text{rank}(g, \Pi)$ and $\text{fool}(g, \Pi)$ (i.e, the weaker version of the rankfix method may be much better than the foolfix method, but the foolfix method may be only a little bit better than the rankfix method).

(v) There exists a Boolean function h such that there is an exponential gap between $\text{rank}(h, \Pi)$ and $\text{Rank}(h, \Pi)$ (i.e. the stronger version of the rankfix method may be essentially better that the weaker one).

Above we see that the tilingfix method is the best one according to the relation to communication complexity. But this fact is not sufficient for using only this method for proving lower bounds on communication complexity because it may happen, for some f and Π, that $\text{Til}(f, \Pi)$ is hard to estimate while $\text{Rank}(f, \Pi)$ or $\text{Fool}(f, \Pi)$ can be easily computed. We discuss this problem still at the end of Section 2.2.2.

Theorem 2.2.2.15 *For every Boolean function f defined on a set X of input variables, and for every partition Π of X*

(i) $\text{Fool}(f, \Pi) \leq \text{Til}(f, \Pi)$, *and*

(ii) $\text{Rank}(f, \Pi) \leq \text{Til}(f, \Pi)$.

Proof.

(i) Let, for a $k \in \mathbb{N}$, $S = \{M_1, \ldots, M_k\}$ be an exact cover of $M(f, \Pi) = [a_{ij}]$, and let $\mathcal{A}$ be a fooling set for f and Π. If α, $\beta \in \mathcal{A}$ and $\alpha \neq \beta$, then the corresponding elements $a_{\text{BIN}(\alpha_{\Pi,L}),\text{BIN}(\alpha_{\Pi,R})}$ and $a_{\text{BIN}(\beta_{\Pi,L}),\text{BIN}(\beta_{\Pi,R})}$ of $M(f, \Pi)$ cannot lie in the same submatrix M_i for some $i \in \{1, \ldots, k\}$ (if they were in the same submatrix, then $f(\alpha) = f(\beta) = f(\Pi^{-1}(\alpha_L, \beta_R)) = f(\Pi^{-1}(\beta_L, \alpha_R))$ which would contradict the fact that $\mathcal{A}$ is a fooling set). Thus $|\mathcal{A}| \leq k$, which implies (i).

(ii) Let $M(f, \Pi)$ have tiling complexity $k = \text{Til}(f, \Pi)$. Let F be any field with identity elements 0 and 1. Then $M(f, \Pi) = M_1 + M_2 + \ldots, +M_d$ for some $d \leq k$, where for every $i \in \{1, \ldots, d\}$ all 1's of M_i can be covered by one monochromatic submatrix of M_i (i.e., $\text{rank}_F(M_i) = 1$ for

every $i \in \{1, \ldots, d\}$). Since $\text{rank}_F(A) \leq \text{rank}_F(B) + \text{rank}_F(C)$ for every matrix $A = B + C$ (Exercise 2.1.5.13), we directly get $\text{rank}_F(M(f, \Pi)) \leq \sum_{i=1}^{d} \text{rank}_F(M_i) = d \leq k = \text{Til}(f, \Pi)$ for every F.

□

We see that $\text{Til}(f, \Pi)$ provides the closest lower bound on $\text{cc}(f, \Pi)$ of the three lower bound methods. The next result shows that $\log_2(\text{Til}(f, \Pi))$ cannot be too far from $\text{cc}(f, \Pi)$.

Theorem 2.2.2.16 *For every Boolean function f defined on a set X of input variables, and for every partition Π of X*

$$\lceil \log_2(\text{Til}(f, \Pi)) \rceil - 1 \leq \text{cc}(f, \Pi) \leq (\lceil \log_2(\text{Til}(f, \Pi)) \rceil + 1)^2.$$

Proof. The fact $\lceil \log_2(\text{Til}(f, \Pi)) \rceil - 1 \leq \text{cc}(f, \Pi)$ has been proved in Theorem 2.2.2.12. The inequality $\text{cc}(f, \Pi) \leq (\lceil \log_2 \text{Til}(f, \Pi)) \rceil + 1)^2$ will be a direct consequence of the investigation of nondeterministic communication complexity in Section 2.5. We omit the nontrivial proof here, because it is only a special case of the more general proof of Theorem 2.5.4.6. □

Next we show that $\log_2(\text{Fool}(f, \Pi))$ and $\log_2(\text{rank}(f, \Pi))$ are not so close to $\text{cc}(f, \Pi)$ as $\log_2(\text{Til}(f, \Pi))$ by showing that the difference between $\text{Til}(f, \Pi)$ on one hand and $\text{Fool}(f, \Pi)$, and $\text{rank}(f, \Pi)$ on the other hand can be very large. Let f_{2m}^{mod} and $\overline{\Pi}$ respectively be the Boolean function and the balanced partition of Example 2.2.2.13.

Theorem 2.2.2.17 *For every Boolean function f defined on a set X of input variables, and for every partition Π of X*

(*i*) $\text{Til}(f, \Pi) \leq 2^{\text{rank}(f,\Pi)+1}$, *and*

(*ii*) $\text{Til}(f_{2m}^{\text{mod}}, \overline{\Pi}) \geq 2^m$ *and* $\text{rank}(f_{2m}^{\text{mod}}, \overline{\Pi}) \leq m$.

Proof. If a Boolean matrix M has $\text{rank}(M) = d$, then M consists of at most 2^d different rows (Exercise 2.1.5.17). Each group of equal rows can be covered by two monochromatic submatrices (one for 1's, and the other for 0's). This implies (*i*).

The fact $\text{Til}(f_{2m}^{\text{mod}}, \Pi) \geq 2^m$ has already been claimed in Example 2.2.2.13. To see that $\text{rank}(f_{2m}^{\text{mod}}, \Pi) \leq m$, consider the m rows of $M(f_{2m}^{\text{mod}}, \Pi)$ corresponding to the following input assignments from $\overline{\Pi_L}$ to $\{0,1\}$: 10^{m-1}, 010^{m-2}, $\ldots$, 0^i10^{m-i-1}, $\ldots$, $0^{m-1}1$. It can be easily observed that all other rows are linear combination of these m rows (more precisely, if a row corresponds to an input assignments with 1's on the positions $i_1, i_2, \ldots, i_r$, then this row is the

sum of rows corresponding to the input assignments $0^{i_1-1}10^{m-i_1}$, $0^{i_2-1}10^{m-i_2}$, $\ldots$, $0^{i_r-1}10^{m-i_r}$). □

We shall show that there is a large difference between $\mathrm{Fool}(f, \Pi)$ and $\mathrm{Til}(f, \Pi)$, but in an existential way. This existential proof is based on the investigation of random (almost all) squared Boolean matrices. To do this we have to observe some interesting properties of fooling sets $\mathcal{A}(f, \Pi)$ connected with the matrix representation $M(f, \Pi)$ of f according to Π.

Let $\mathcal{A} = \mathcal{A}(f, \Pi)$ be a fooling set for f and Π, and let $|\mathcal{A}| = m$. Without loss of generality we may assume that, for $\forall \alpha \in \mathcal{A}$, $f(\alpha) = 1$. Let $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, and let $a_{i_1 j_1}, a_{i_2 j_2}, \ldots, a_{i_m j_m}$ be the elements of $M(f, \Pi)$ corresponding to $\alpha_1, \alpha_2, \ldots, \alpha_m$ respectively. (Note that for each $\alpha_k = \Pi^{-1}(\alpha_{k\Pi,L}, \alpha_{k\Pi,R})$, $i_k = \mathrm{BIN}(\alpha_{k\Pi,L}) + 1$ and $j_k = \mathrm{BIN}(\alpha_{k\Pi,R}) + 1$.)

Now we show that, for all $r, s \in \{1, \ldots, m\}$, $r \neq s$ implies $i_r \neq i_s$ and $j_r \neq j_s$ (i.e., no column (or row) may contain values for two different input assignments from $\mathcal{A}$). We prove this fact by contradiction. Let there exist $u, v \in \{1, \ldots, m\}$ such that $u \neq v$ and $i_u = i_v [j_u = j_s]$. This means $\alpha_{u\Pi,L} = \alpha_{v\Pi,L} [\alpha_{u\Pi,R} = \alpha_{v\Pi,R}]$ which implies $\Pi^{-1}(\alpha_{u\Pi,L}, \alpha_{v\Pi,R}) = \Pi^{-1}(\alpha_{v\Pi,L}, \alpha_{v\Pi,R}) = \alpha_v$ and $\Pi^{-1}(\alpha_{v\Pi,L}, \alpha_{u\Pi,R}) = \Pi^{-1}(\alpha_{u\Pi,L}, \alpha_{u\Pi,R}) = \alpha_u$ $[\Pi^{-1}(\alpha_{u\Pi,L}, \alpha_{v\Pi,R}) = \Pi^{-1}(\alpha_{u\Pi,L}, \alpha_{u\Pi,R}) = \alpha_u$ and $\Pi^{-1}(\alpha_{v\Pi,L}, \alpha_{u\Pi,R}) = \Pi^{-1}(\alpha_{v\Pi,L}, \alpha_{v\Pi,R}) = \alpha_v]$.

Thus, $f(\Pi^{-1}(\alpha_{u\Pi,L}, \alpha_{v\Pi,R})) = f(\alpha_v) = 1$ and $f(\Pi^{-1}(\alpha_{v\Pi,L}, \alpha_{u\Pi R})) = f(\alpha_u) = 1 [f(\Pi^{-1}(\alpha_{u\Pi,L}, \alpha_{v\Pi,R}) = f(\alpha_u) = 1$ and $f(\Pi^{-1}(\alpha_{v\Pi,L}, \alpha_{u\Pi,R})) = f(\alpha_v) = 1]$, which contradicts the fact that $\mathcal{A}$ is a fooling set for f and Π.

So, we see that one can assign a squared submatrix of $M(f, \Pi)$ of the size $|\mathcal{A}(f, \Pi)| \times |\mathcal{A}(f, \Pi)|$ to each fooling set $\mathcal{A}(f, \Pi)$.

On the other hand, each submatrix with the above described properties defines a fooling set.

Definition 2.2.2.18 *Let $\delta \in \{0, 1\}$. A squared Boolean matrix $M = [m_{ij}]_{i,j=1,\ldots,d}$ for some $d \in \mathbb{N} - \{0\}$ is called a* **δ-fooling matrix** *iff*

(i) $m_{ii} = \delta$ *for* $i = 1, \ldots, d$, *and*

(ii) for all $r, s \in \{1, \ldots, d\}$, $r \neq s$ *implies* $m_{rs} = \overline{\delta}$ *or* $m_{sr} = \overline{\delta}$.

Any matrix M' obtained from a δ-fooling matrix M by any permutation of rows and columns of M is called a **δ-quasifooling matrix**.

Now our idea is to show that, for sufficiently large m, most of the $m \times m$ Boolean matrixes do not have any large quasifooling matrix, but they have a large tiling number. Obviously, each such matrix may be considered as $M(f, \Pi)$ for some balanced Π and some Boolean function f. Let us start with some technical results.

Lemma 2.2.2.19 *Let* $\mathrm{Mf}(m, k)$ *be the number of all $m \times m$ Boolean matrices having a δ-quasifooling submatrix of size at least $k \times k$ for some $\delta \in \{0, 1\}$. Then*

$$\mathrm{Mf}(m,k) \leq 2 \cdot \binom{m}{k}^2 \cdot k! \cdot 3^{\binom{k}{2}} \cdot 2^{m^2-k^2}.$$

Proof. We have two possibilities to choose $\delta \in \{0,1\}$, and $\binom{m}{k}^2$ possibilities to choose a placement of the $k \times k$ δ-quasifooling submatrix M' ($\binom{m}{k}$ possibilities to choose k rows [columns] from m rows [columns]). k elements of M' have fixed values δ and their positions in M' can be chosen in $k!$ different ways. If we permute the rows of M' to get a δ-fooling matrix M, then we see that we have only three possibilities for assiging the values to any pair of symmetrical elements of M ($(\overline{\delta},\overline{\delta})$, $(\overline{\delta},\delta)$, and $(\delta,\overline{\delta})$). Thus, there are $3^{\binom{k}{2}}$ possibilities for choosing the values for the elements in M'. All other elements lying outside M' may be chosen arbitrarily, providing $2^{m^2-k^2}$ possibilities. □

Lemma 2.2.2.20 *Let* $\mathrm{Mt}(m,a,b)$ *be the number of all* $m \times m$ *Boolean matrices having a monochromatic submatrix of size* $a \times b$. *Then*

$$\mathrm{Mt}(m,a,b) \leq 2 \cdot \binom{m}{a} \cdot \binom{m}{b} \cdot 2^{m^2-a\cdot b}.$$

Proof. There are $\binom{m}{a} \cdot \binom{m}{b}$ possibilities for choosing the position of an $a \times b$ submatrix. We have two possibilities for the value of elements in this monochromatic submatrix, and $2^{m^2-a\cdot b}$ possibilities for the values of elements lying outside this monochromatic submatrix. □

Lemma 2.2.2.21 *For every* $k \geq 20\log_2 m$,

$$\lim_{m\to\infty} \mathrm{Mf}(m,k)/2^{m^2} = 0.$$

Proof. To show this it is sufficient to show $\lim_{m\to\infty} 2 \cdot \binom{m}{k}^2 \cdot (k!) \cdot 3^{\binom{k}{2}} \cdot 2^{-k^2} = 0$ for $k \geq 20\log_2 m$ (see Lemma 2.2.2.19). We asympthotically bound this expression in the following way:

$$\begin{aligned} 2 \cdot \binom{m}{k}^2 \cdot (k!) \cdot 3^{\binom{k}{2}} \cdot 2^{-k^2} &\leq 2 \cdot \frac{(m-\frac{k}{2})^{2k}}{k!} \cdot 3^{k^2/2} \cdot 2^{-k^2} \\ &\leq 2^{1-k^2+k^2(\frac{1}{2}\cdot\log_2 3)+2k\cdot\log_2(m-\frac{k}{2})} \\ &\leq 2^{k^2(-1+(\log_2 3)/2+(2\log_2 m)/k)}. \end{aligned}$$

It is easy to see that $(\log_2 3)/2 + 2\log_2 m/k < 1$ for any $k \geq 20\log_2 m$, and so

$$\lim_{m\to\infty} k^2(-1+(\log_2 3)/2+(2\log_2 m)/k) = -\infty$$

for $k \geq 20\log_2 m$. □

Lemma 2.2.2.22 *Let* $\mathrm{Mt}(m,r) = \sum_{a\cdot b=r} \mathrm{Mt}(m,a,b)$ *be the number of all* $m \times m$ *Boolean matrices having a monochromatic submatrix of* r *elements. Then*

$$\lim_{m\to\infty} \mathrm{Mt}(m,r)/2^{m^2} = 0$$

for any $r \geq 3 \cdot m \log_2 m$.

Proof. It is sufficient to show that $\mathrm{Mt}(m,a,b)/2^{m^2} \leq 2^{-m}$ for every a, $b \leq m$ with $r = a \cdot b \geq 3m \log_2 m$.

$$\begin{aligned} \mathrm{Mt}(m,a,b)/2^{m^2} &\leq \binom{m}{a} \cdot \binom{m}{b} \cdot 2^{1-ab} \\ &\leq m^a \cdot m^b \cdot 2^{-r} \leq 2^{-r+(a+b)\cdot \log_2 m} \\ &\leq 2^{-r+2m\log_2 m}. \end{aligned}$$

Obviously, for $r \geq 3m \log_2 m$ we have $\mathrm{Mt}(m,a,b)/2^{m^2} \leq 2^{-m\log_2 m}$. □

Now, we can formulate our comparison results.

Theorem 2.2.2.23 *There exists a Boolean function* f *and a balanced partition* Π *such that*

(i) $\mathrm{Til}(f,\Pi) \geq m/3\log_2 m$, *and*

(ii) $\mathrm{Fool}(f,\Pi) \leq 20\log_2 m$.

Proof. Following the Lemmas 2.2.2.21 and 2.2.2.22 we obtain that there exists a sufficiently large m such that

(*iii*) more than half of the $m \times m$ Boolean matrices do not have any δ-quasifooling submatrix of size $20\log_2 m \times 20\log_2 m$, and

(*iv*) more than half of the $m \times m$ Boolean matrices do not have any monochromatic submatrices of $3m\log_2 m$ elements, i.e., they have tiling number at least $m/3\log_2 m$.

Thus, there exists an $m \times m$ Boolean matrix M with both properties (*iii*) and (*iv*), and clearly $M = M(f,\Pi)$ for some f and Π. □

Next we compare $\mathrm{rank}(f,\Pi)$ and $\mathrm{Fool}(f,\Pi)$. For this purpose, we need the following definition.

Definition 2.2.2.24 *Let* $\mathcal{A} = \mathcal{A}(f,\Pi) = \{\alpha_1,\alpha_2,\ldots,\alpha_m\}$ *be a fooling set for a Boolean function* f *and an almost balanced partition* Π. *Let* $a_{i_1j_1},\ldots,a_{i_mj_m}$ *be the* m *elements of* $M(f,\Pi)$ *corresponding to* $\alpha_1,\ldots,\alpha_m$. *We define* $M(\mathcal{A},f,\Pi)$ *to be the* $m \times m$ *submatrix of* $M(f,\Pi)$ *obtained as the intersection of the rows*

$i_1, i_2, \ldots, i_m$ and of the columns $j_1, j_2, \ldots, j_m$. We define $M'(\mathcal{A}, f, \Pi)$ to be an $m \times m$ matrix obtained from $M(\mathcal{A}, f, \Pi)$ by some row and column permutations such that the m elements of the diagonal of $M'(\mathcal{A}, f, \Pi)$ correspond to the m elements of the fooling set $\mathcal{A}$.

Note that Definition 2.2.2.24 is consistent (i.e., $M'(\mathcal{A}, f, \Pi)$ always exists) due to the property of fooling sets proved above.

Now, we introduce a special type of fooling sets with a direct connection to the rank of matrices $M(f, \Pi)$.

Definition 2.2.2.25 *Let f_n be a Boolean function with the set of input variables $X = \{x_1, \ldots, x_m\}$, and let Π be a partition in* Abal(X). *For a σ-fooling set $\mathcal{A}(f_n, \Pi) = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ for f_n and Π, and $\sigma \in \{0, 1\}$, we say that $\mathcal{A}(f_n, \Pi)$ is a* **σ-regular fooling set** *if*

(i) there exists a permutation $\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_m}$ of $\alpha_1, \alpha_2, \ldots, \alpha_m$ such that $f_n(\Pi^{-1}(\alpha_{i_r \Pi, L}, \alpha_{i_s \Pi, R})) = \overline{\sigma}$ for every $r < s, r, s \in \{1, \ldots, m\}$.

We observe that condition (i) of Definition 2.2.2.25 is stronger than the conditions given in the definition of fooling sets, because Definition 2.2.2.25 determines which one of $f_n(\Pi^{-1}(\alpha_{i_r \Pi, L}, \alpha_{i_s \Pi, R}))$ and $f_n(\Pi^{-1}(\alpha_{i_s \Pi, L}, \alpha_{i_r \Pi, R}))$ must be different from σ.

Now we can already see why the σ-regular fooling sets have been introduced. If $\mathcal{A} = \mathcal{A}(f, \Pi) = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ is a σ-regular fooling set and $\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_m}$ is the permutation satisfying (i) of Definition 2.2.2.25, then $M'(\mathcal{A}, f, \Pi)$ is a $m \times m$ matrix such that, just by permuting the rows and columns of $M'(\mathcal{A}, f, \Pi)$, we can get a matrix $M''(\mathcal{A}, f, \Pi) = [b_{ij}]_{i,j=1,\ldots,m}$, where

$$b_{jj} = f(\alpha_{i_j}) = \sigma \text{ for } j = 1, \ldots, m$$

$$b_{rs} = f(\Pi^{-1}(\alpha_{i_r \Pi, L}, \alpha_{i_s \Pi, R})) = \overline{\sigma} \text{ for every } r < s, r, s = 1, \ldots, m.$$

Doubtlessly,
$\text{rank}(M''(\mathcal{A}, f, \Pi)) = m$ if $\delta = 1$, and $m - 1 \leq \text{rank}(M''(\mathcal{A}, f, \Pi)) \leq m$ if $\delta = 0$ (see Exercises 2.1.5.19, and 2.2.4.5 if it is not clear). Since no permutation of rows or columns of a matrix can change the rank of this matrix we have

$$m - 1 \leq \text{rank}(M'(\mathcal{A}, f, \Pi)) = \text{rank}(M(\mathcal{A}, f, \Pi)) \leq m.$$

Since $M(\mathcal{A}, f, \Pi))$ is a submatrix of $M(f, \Pi)$ we finally get $\text{rank}(f, \Pi) \geq m - 1$. So we have proved the following result.

Theorem 2.2.2.26 *For any Boolean function f with a set of input variables X, and any almost balanced partition Π of X,*

$\text{rank}(f, \Pi) \geq \max\{|\mathcal{A}(f, \Pi)| \mid \mathcal{A}$ *is a σ-regular fooling set for f and Π, $\sigma \in \{0, 1\}\} - 1$.*

Thus, Theorem 2.2.2.26 claims that if a σ-regular fooling set is among the largest fooling sets for f and Π, then the lower bound method rankfix provides lower bounds at least as high as those given by the method based on fooling sets. However, this is not always the case (i.e., there are Boolean functions with large fooling sets but without equally large σ-regular fooling sets). Before proving this fact, let us look at the structure of the matrix $M'(\mathcal{A}, f, \Pi)$ for an arbitrary fooling set $\mathcal{A}$. Let $f(\alpha) = \sigma$ for every $\alpha \in \mathcal{A}$. Then, $M'(\mathcal{A}, f, \Pi)$ contains only σ on the diagonal. If $M'(\mathcal{A}, f, \Pi) = [b_{ij}]_{i,j=1,\ldots,m}$ for some $m \in \mathbb{N}$, then we note that, for any two (symmetric) elements b_{rs} and $b_{sr}, r \neq s$ implies that $b_{rs} = \overline{\sigma}$ or $b_{sr} = \overline{\sigma}$. To realize this, note that:

(1) $b_{rs} = f(\Pi^{-1}(\beta_{\Pi,L}, \gamma_{\Pi,R}))$ and $b_{sr} = f(\Pi^{-1}(\gamma_{\Pi,L}, \beta_{\Pi,R}))$ for some $\beta, \gamma \in \mathcal{A}$, and

(2) $\mathcal{A}$ is a fooling set, and so for all $\beta, \gamma \in \mathcal{A}$, $f(\Pi^{-1}(\beta_{\Pi,L}, \gamma_{\Pi,R})) \neq \sigma$ or $f(\Pi^{-1}(\gamma_{\Pi,L}, \beta_{\Pi,R})) \neq \sigma$; i.e., $M'(\mathcal{A}, f, \Pi)$ is a δ-fooling matrix.

Obviously, if a σ-fooling matrix M is a submatrix of a matrix $M(f, \Pi)$ for some f and Π, then M unambiguously defines a σ-fooling set for f and Π (the inputs corresponding to the positions of the diagonal of M are the elements of the fooling set). To find f and Π such that there exists a large fooling set $\mathcal{A}(f, \Pi)$ with the rank of $M(f, \Pi)$ smaller than $|\mathcal{A}(f, \Pi)|$, it is sufficient to build a σ-fooling matrix M with rank(M) smaller than the size of M (Note that each Boolean matrix of size $2^d \times 2^d$ together with an arbitrary partition of $2d$ variables unambiguously define a Boolean function of $2d$ variables. Moreover, if this matrix is a σ-fooling matrix, then the 2^d inputs corresponding to the diagonal build the fooling set for f and Π).

We start by presenting a 1-fooling matrix M_1 of size 4×4 with $\text{Rank}(M_1) = \text{rank}(M_1) = 3$ in Fig. 2.3.

$$M_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 2.3.

The singularity of M_1 over every field F follows directly from the fact that $\text{row}_1(M_1) + \text{row}_3(M_1) = \text{row}_2(M_1) + \text{row}_4(M_1)$.

Now we extend this idea to matrices of size $4^d \times 4^d$ for any positive integer d.

Lemma 2.2.2.27 *For any positive integer d there exists a 1-fooling matrix M_d of size $4^d \times 4^d$ with* $\text{Rank}(M_d) = \text{rank}(M_d) = 3^d$.

Proof. We prove this by induction on d. For $d = 1$ the matrix M_1 depicted in Fig. 2.3 has the required properties. Now let there be a 1-fooling matrix $M_d = [a_{ij}]_{i,j=1,\ldots,2^d}$ with $\text{Rank}(M_d) = 3^d$ for some $d \geq 1$. We construct M_{d+1} from M_d by substituting each element $a_{ij} = 1$ of M_d by the matrix M_1, and each element $a_{ij} = 0$ of M_d by the zero-matrix of size 4×4. Obviously, M_{d+1} has size $4^{d+1} \times 4^{d+1}$. Let us denote $M_{d+1} = [b_{ij(rs)}]_{i,j=1,\ldots,4^d}$, where $b_{ij(rs)}$ is the element of the intersection of the r-th row and the s-th column of the 4×4 matrix substituted at the position (i, j) of the matrix M_d. For $m = 1, \ldots, 4^d, k = 1, \ldots 4$, let $\text{row}_{m,k}(M_{d+1})$ denote $\text{row}_{4(m-1)+k}(M_{d+1})$ obtained by the substition of $\text{row}_m(M_d) = (a_{m1}, a_{m2}, \ldots, a_{m4^d})$ by

$$(b_{m1(k1)}, b_{m1(k2)}, b_{m1(k3)}, b_{m1(k4)}, b_{m2(k1)}, \ldots, b_{m4(k4)}, \ldots, a_{m4^d(k1)}, \ldots, a_{m2^d(k4)}).$$

First, let us prove that M_{d+1} is a 1-fooling matrix. Since all diagonal elements $a_{ii} (i = 1, \ldots, 4^d)$ of M_d are 1's, and also all diagonal elements of M_1 are 1's, all diagonal elements $b_{ii(rr)} (i = 1, \ldots, 4^d, r = 1, \ldots, 4)$ are 1's too. Let $b_{ii(rr)}$ and $b_{jj(ss)}$ be two different diagonal elements of M_{d+1}. If $i = j$, then $b_{ii(rs)} = 0$ or $b_{ii(sr)} = 0$, because M_1 is a 1-fooling matrix. If $i \neq j$, then $a_{ij} = 0$ or $a_{ji} = 0$. Without loss of generality, we assume $a_{ij} = 0$, which means that a_{ij} was substituted by the 4×4 zero-matrix. This implies $b_{ij(rs)} = 0$, because $b_{ij(uv)} = 0$ for all $u, v \in \{1, \ldots, 4\}$. So we have proved that M_{d+1} is a 1-fooling matrix.

Now we shall show that $\text{Rank}(M_{d+1}) = \text{rank}(M_{d+1}) = 3^{d+1}$. According to the induction hypothesis we have $\text{Rank}(M_d) = \text{rank}(M_d) = 3^d$. We also observe that the first three rows of M_1 build an independent set of vectors over R. Let $S_d = \{p_1, p_2, \ldots, p_{3^d}\}$ be a set of numbers from $B_d = \{1, 2, \ldots, 4^d\}$ such that $R(S_d) = \{\text{row}_k(M_d) \mid k \in S_d\}$ is an independent set, and let, for each $z \in B - S$, $\text{row}_z(M_d)$ be a linear combination of vectors in $R(S_d)$. We claim that

(1) $R(S_{d+1}) = \{\text{row}_{m,k}(M_{d+1}) \mid m \in S_d \text{ and } k \in \{1, 2, 3\}\}$ is an independent set over R and Z_2, and

(2) each $\text{row}_{u,v}(M_{d+1}) \notin R(S_{d+1})$ is a linear combination (over Z_2) of rows in $R(S_{d+1})$.

(1) follows immediately from the facts that $R(S_d)$ is an independent set, and that the first three rows of M_1 build an independent set.

To show (2) we distinguish two possibilities. First, let $\text{row}_{u,v}(M_{d+1}) \notin R(S_{d+1})$, and $\text{row}_u(M_d) \notin R(S_d)$. The fact, that $\text{row}_u(M_d)$ is a linear combination (sum) of some vectors $\text{row}_{i_1}(M_d), \ldots, \text{row}_{i_s}(M_d)$ for some $s \in \mathbb{N}$, implies immediately that $\text{row}_{u,v}(M_{d+1})$ is the same linear combination of the vectors $\text{row}_{i_1,v}(M_{d+1}), \ldots, \text{row}_{i_s,v}(M_{d+1})$. The second (remaining) possibility for a row not in $R(S_{d+1})$ is that the row is $\text{row}_{u,4}(M_{d+1})$ for some $\text{row}_u(M_d) \in R(S_d), u \in \{1, \ldots 4^d\}$. In this case, $\text{row}_{u,4}(M_{d+1})$ is a linear combination of the vectors $\text{row}_{u,1}(M_{d+1})$, $\text{row}_{u,2}(M_{d+1})$, and $\text{row}_{u,3}(M_{d+1})$. [A detailed, formal proof of facts (1) and (2) is left as a simple exercise for the reader]. □

An immediate consequence of Lemma 2.2.2.27 is the following theorem.

Theorem 2.2.2.28 *For every positive integer $n, n = 4m, m \in \mathbb{N}$, and every partition $\Pi \in \mathrm{Bal}(\{x_1, \ldots, x_n\})$, there is a Boolean function f_n depending on the variables $x_1, \ldots, x_n$ such that:*

(i) there exists a fooling set $\mathcal{A}(f_n, \Pi)$ for f_n and Π with $|\mathcal{A}(f_n, \Pi)| = 2^{n/2}$ (i.e., $\mathrm{cc}(f_n, \Pi) = n/2$), and

(ii) $\mathrm{Rank}(f_n, \Pi) = 3^{n/4}$ (i.e., the rank lower bound method provides only $\mathrm{cc}(f_n, \Pi) \geq ((\log_2 3)/4) \cdot n$).

Theorem 2.2.2.28 shows that there are computing problems for which the fooling set method can provide a litle bit higher lower bounds than the method based on the matrix rank. But the difference between these two lower bounds $n/2$ and $(log_2 3) \cdot n/4$ is only linear, and the question whether this difference may be greater for some computing problems will be answered in what follows.

First, we need the notion of Kronecker product of two matrices generalizing the construction of M_{d+1} from M_d and M_1 in Lemma 2.2.2.27.

Definition 2.2.2.29 *Let F be a field. For arbitrary finite index sets $I, J, K, L \neq \emptyset$ and matrices $A = (\alpha_{ij})_{i \in I, j \in J} \in F^{I \times J}$, $B = (\beta_{kl})_{k \in K, l \in L} \in F^{K \times L}$ over F, the* **Kronecker product** $A \otimes B$ *is defined as the matrix*

$$C = (\gamma_{(i,k),(j,l)})_{(i,k,j,l) \in I \times K \times J \times L}$$

where $\gamma_{(i,k),(j,l)} = \alpha_{i,j} \cdot \beta_{k,l}$.

Observation 2.2.2.30 *For arbitrary A and B over some field F*

$$\mathrm{rank}_F(A \otimes B) = \mathrm{rank}_F(A) \cdot \mathrm{rank}_F(B).$$

Proof. Since the proof of this fact is a simple generalization of the idea of the proof of the fact $\mathrm{rank}_F(M_{d+1}) = \mathrm{rank}_F(M_d) \cdot \mathrm{rank}_F(M_1)$ in the proof of Lemma 2.2.2.27, it is left as an exercise to the reader. □

Now, we are ready to show that $\mathrm{Fool}(f, \Pi)$ cannot be essentially larger than $\mathrm{rank}_F(f, \Pi)$ for any f, Π.

Theorem 2.2.2.31 *Let X be a set of Boolean variables, $|X| = 2n$. For all fields F with identity elements 0,1, all Boolean functions f over X, and all partitions $\Pi \in \mathrm{Bal}(X)$,*

$$\mathrm{Fool}(f, \Pi) \leq (\mathrm{rank}_F(f, \Pi) + 2)^2.$$

Proof. Let F be a field with the identity elements 0 and 1. Let $X = \{x_1, \ldots, x_{2n}\}$ be a set of Boolean variables, and let $\Pi \in \mathrm{Bal}(X)$. The idea of the proof is to construct, for every Boolean function f over X, a Boolean function f^* such

that $\text{rank}_F(M(f^*, \Pi')) \leq (\text{rank}_F(M(f, \Pi)) + 1)^2$ and $\text{rank}(M(f^*, \Pi')) + 1 \geq \text{Fool}(f, \Pi)$ for some partition Π' of the input variables of f^*.

For every function $f \in B_2^{2n}$, we define $f^* : \{0,1\}^{4n} \to \{0,1\}$ over $X' = X \cup \{y_1, y_2, \ldots, y_{2n}\}$ as follows. For every $\alpha, \beta \in \{0,1\}^{2n}$,

$$f^*(\alpha, \beta) = f(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,L})) \cdot f(\Pi^{-1}(\beta_{\Pi,R}, \alpha_{\Pi,R})).$$

We define $\Pi' \in \text{Bal}(X')$ by setting $\Pi'_L = X$ and $\Pi'_R = \{y_1, y_2, \ldots, y_{2n}\}$.

First of all we prove $(\text{rank}_F(M(f, \Pi)) + 1)^2 \geq \text{rank}_F(M(f^*, \Pi'))$. To do it we define the Boolean function $f^R \in B_2^{2n}$ by setting $f^R(\gamma) = f(\Pi^{-1}(\gamma_{\Pi,R}, \gamma_{\Pi,L}))$ for every $\gamma \in \{0,1\}^{2n}$. We observe $M(f^*, \Pi') = M(f, \Pi) \otimes M(f^R, \Pi)$. Because of Observation 2.2.2.30 we obtain

$$\text{rank}_F(M(f^*, \Pi')) = \text{rank}_F(M(f, \Pi)) \cdot \text{rank}_F(M(f^R, \Pi))$$

$$\leq \text{rank}_F(M(f, \Pi)) \cdot (\text{rank}_F(M(f, \Pi)) + 1).$$

It remains to show $\text{rank}_F(M(f^*, \Pi')) + 1 \geq \text{Fool}(f, \Pi)$. We prove it by showing $\text{rank}_F(M(f^*, \Pi')) + 1 \geq |\mathcal{A}|$ for every δ-fooling set $\mathcal{A}$ for f and Π. We distinquish two possibilities according to δ.

Let $\delta = 1$. Assume $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\} \subseteq \{0,1\}^{2n}$ be a 1-fooling set for f and Π. We claim that the submatrix of $M(f^*, \Pi')$ obtained by the intersection of the rows corresponding to the input assignments $\alpha_1, \alpha_2, \ldots, \alpha_r$ from Π'_L to $\{0,1\}$ and the columns corresponding to the input assignments $\Pi^{-1}(\alpha_{1\Pi,R}, \alpha_{1\Pi,L}), \Pi^{-1}(\alpha_{2\Pi,R}, \alpha_{2\Pi,L}), \ldots, \Pi^{-1}(\alpha_{r\Pi,R}, \alpha_{r\Pi,L})$ from Π'_R to $\{0,1\}$ is a diagonal matrix. For this, observe that

$$a_{ij} = M(f^*, \Pi')[\alpha_i, \Pi^{-1}(\alpha_{j\Pi,R}, \alpha_{j\Pi,L})] = f^*(\alpha_i, \Pi^{-1}(\alpha_{j\Pi,R}, \alpha_{j\Pi,L}))$$

$$= f(\Pi^{-1}(\alpha_{i\Pi,L}, \alpha_{j\Pi,R})) \cdot f(\Pi^{-1}(\alpha_{j\Pi,L}, \alpha_{i\Pi,R})).$$

If $i = j$ then $a_{ij} = a_{ii} = f(\alpha_i) \cdot f(\alpha_i) = 1$. If $i \neq j$, then the fact that $\mathcal{A}$ is a 1-fooling set implies $f(\Pi^{-1}(\alpha_{i\Pi,L}, \alpha_{j\Pi,R})) = 0$ or $f(\Pi^{-1}(\alpha_{j\Pi,L}, \alpha_{i\Pi,R})) = 0$. Thus $a_{ij} = 0$ for $i \neq j$. So, we have obtained

$$|\mathcal{A}| \leq \text{rank}_F(M(f^*, \Pi')) \leq (\text{rank}_F(M(f, \Pi)) + 1)^2.$$

Let $\delta = 0$. If $|\mathcal{A}|$ is a 0-fooling set for f and Π, then $|\mathcal{A}|$ is a 1-fooling set for $g(x_1, \ldots, x_{2n}) = f(x_1, \ldots, x_{2n}) \oplus 1$ and Π. Following the case $\delta = 1$ above we obtain

$$|\mathcal{A}| \leq \text{rank}_F(M(g^*, \Pi')) \leq (\text{rank}_F(M(g, \Pi)) + 1)^2 \leq (\text{rank}_F(M(f, \Pi)) + 2)^2.$$

□

Note that Theorem 2.2.2.31 implies that the rankfix method provides lower bounds which are in the worst case the half of the lower bounds provided by the foolfix method, but never smaller.

In the opposite direction we show in an existential way that $\text{rank}(f, \Pi)$ can be exponentially larger than $\text{Fool}(f, \Pi)$.

Theorem 2.2.2.32 *There exists a Boolean function f of $2m$ variables and a balanced partition Π such that*

(i) $\text{Fool}(f, \Pi) \leq 20 \cdot m$,

(ii) $\text{rank}(f, \Pi) = 2^m$ *(i.e.,* $\text{cc}(f, \Pi) = m$*).*

Proof. Lemma 2.2.2.21 provides that for sufficiently large m more than three quarters of the $2^m \times 2^m$ matrices do not have any δ-quasifooling submatrix of size $20 \cdot m$. On the other hand, following the assertion of Exercise 2.1.5.8 for sufficiently large m, at least one quarter of the $2^m \times 2^m$ matrices has rank equal to 2^m. Thus, for sufficiently large m, there exists a $2^m \times 2^m$ matrix m with $\text{rank}(M) = 2^m$ and no δ-quasifooling submatrix of size at least $20m$. □

Now, we show that $\log_2(\text{rank}(M(f, \Pi)))$ can differ from $\text{cc}(f, \Pi)$ at most exponentially.

Theorem 2.2.2.33 *Let f be a Boolean function with a set of input variables $X = \{x_1, x_2, \ldots, x_n\}$, and let $\Pi \in \text{Bal}(X)$. Then*

$$\log_2(\text{Rank}(f, \Pi)) \leq \text{cc}(f, \Pi) \leq \text{rank}(f, \Pi) \leq \text{Rank}(f, \Pi).$$

Proof. The fact $\log_2(\text{Rank}(M(f, \Pi))) \leq \text{cc}(f, \Pi)$ is established in Theorem 2.2.2.7. The upper bound $\text{cc}(f, \Pi) \leq \text{rank}(M(f, \Pi))$ follows from the following consideration. Each Boolean matrix with the rank m over Z_2 has at most 2^m distinct rows (Exercise 2.1.5.17). So a protocol $D = \langle \Pi, \Phi \rangle$ computing f can work as follows. The inputs of the first computer are partitioned into at most $z = 2^{\text{rank}(M(f,\Pi))}$ classes, each class containing inputs whose correspondig rows of $M(f, \Pi)$ are equal to each other. The first computer sends a message out of the set of z different messages of the length $\text{rank}(M(f, \Pi))$ (we have one message of the set for each class of inputs). After receiving a message of the length $\text{rank}(M(f, \Pi))$ the second computer knows that the part of the input assigned to the first computer is in the class of inputs corresponding to the message submitted. Since all rows corresponding to inputs in one class are equal, the output depends only on the input part assigned to the second computer. Thus, the second computer immediately computes the output. □

Finally, we show that the stronger rankfix method based on the rank over Q can be exponentially better than foolfix method and the weaker rankfix method based on the rank over Z_2. To show it we take the same sequence of Boolean functions as in Theorem 2.2.2.17. For any $n \in \mathbb{N}$, we consider the inner product function $f_{2n}^{\text{mod}} \in B_2^n$ defined as follows:

$$f_{2n}^{\text{mod}}(x_1, \ldots, x_n, y_1, \ldots, y_n) = \left(\sum_{i=1}^{n}(x_i \wedge y_i)\right) \bmod 2.$$

Let $X_n = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ and let $\Pi^n = (\Pi_L^n, \Pi_R^n)$ be a balanced partition of X_n with $\Pi_L^n = \{x_1, \ldots, x_n\}$.

Theorem 2.2.2.34 *For every $n \in \mathbb{N}$,*

(i) $\text{rank}(M(f_{2n}^{\text{mod}}, \Pi^n)) = n$,

(ii) $\text{Fool}(f_{2n}^{\text{mod}}, \Pi^n) \leq (n+2)^2$, *and*

(iii) $\text{Rank}(f_{2n}^{\text{mod}}, \Pi^n) = 2^n - 1$.

Proof. To see $\text{rank}(M(f_{2n}^{\text{mod}}, \Pi^n)) = n$, consider the n rows of $M(f_{2n}^{\text{mod}}, \Pi^n)$ corresponding to the rows for input assignments

$$10^{n-1}, 010^{n-2}, \ldots, 0^i10^{n-i-1}, \ldots, 0^{n-1}1$$

from Π_L^n to $\{0,1\}$. Obviously, these rows are linearly independent. It can easily be observed that all other rows are linear combinations of these n rows (more precisely,if a row corresponds to a "left" part of the input with 1's on the positions $i_1, i_2, \ldots, i_r$, then this row is the sum of the rows corresponding to the input assignments

$$0^{i_1-1}10^{n-i_1}, 0^{i_2-1}10^{n-i_2}, \ldots, 0^{i_r-1}10^{n-i_r}$$

from Π_L^n to $\{0,1\}$).

The claim *(ii)* follows directly from *(i)* and the assertion of Theorem 2.2.2.31.

To show *(iii)* it is sufficient to show $\text{rank}_Q(M(f_{2n}^{\text{mod}}, \Pi^n)) = 2^n - 1$. But this is the claim of Exercise 2.1.5.21. □

We conclude this subsection by discussing the comparison results established. We have shown that the tilingfix method always provides better results than the other two methods, and that the methods rankfix and foolfix are incomparable. Rankfix may be exponentially better than foolfix, and foolfix can be only linearly better than rankfix.

The tilingfix method is polynomially close to communication complexity. Whether rankfix method is polynomially close to communication complexity is an open problem. The weaker version of rankfix method based on the rank over Z_2 is only exponentially close to communication complexity and we have given an example revealing this exponential gap. The lower bound provided by foolfix may differ exponentially from communication complexity, too.

Whether $\log_2(\text{Rank}(f, \Pi))$ is polynomially close to $\text{cc}(f, \Pi)$ for all f and Π remains as the only one open comparison problem. Observe that the claim of

Exercise 2.1.5.22 provides some intuition supporting the positive answer to this question.

But the "quality" of the techniques compared above cannot be measured only by their relation to communication complexity. A completely different issue is how hard it is to use them to get lower bounds for concrete problems. It seems that the most (successfully) used method is foolfix, followed by rankfix. The reason for this is that using foolfix means constructing a fooling set. So you prove a lower bound by a construction. If you want to use the tilingfix method you have to solve a minimization problem which is usually very hard. If you want to use the rankfix method, you can compute the rank in simple cases, but you often have to prove a nontrivial statement if you want to use this method for proving a lower bound on $cc(h_n(L), \Pi_n)$ for all infinitely many n's.

2.2.3 Theoretical Properties of Communication Complexity According to a Fixed Partition

This subsection is devoted to the study of some theoretical properties of the communication complexity introduced in Section 2.2.1. Most of the results of this subsection are not so important for applications (the relation to real computing models) and they may be skipped by readers primarily interested in the practical use of communication complexity. On the other hand, these theoretical results help to understand the nature of communication complexity and they are not only of theoretical importance.

The theoretical properties of the communication complexity measure are mainly studied only for one-output computing problems and for balanced partitions. So we only consider Boolean functions and balanced partitions here.

Definition 2.2.3.1 *We define, for any* $m \leq n, m, n \in \mathbb{N}$ *and any balanced partition* Π *of* n *variables* $\mathrm{COMM}_{n,\Pi}(m) = \{f \in B_2^n \mid cc(f, \Pi) \leq m\}$ *as the set of Boolean functions of* n *variables computable within communication complexity* m *by the partition* Π.

We now study the properties of the classes $\mathrm{COMM}_{n,\Pi}(m)$.

Observation 2.2.3.2 *For any balanced partition* Π *of* $2n$ *variables*

$$\mathrm{COMM}_{2n,\Pi}(n) = B_2^n.$$

The next result shows that almost all Boolean functions have the highest communication complexity. In what follows, let Π_i denote a balanced partition of the set $X_i = \{x_1, \ldots, x_i\}$.

Theorem 2.2.3.3 *For any sequence of balanced partitions* $\{\Pi_{2n}\}_{n=1}^{\infty}$

$$\lim_{n\to\infty} \frac{|\mathrm{COMM}_{2n,\Pi_{2n}}(n-1)|}{|B_2^{2n}|} = 0.$$

Proof. The number of all Boolean functions of $2n$ variables is $|B_2^{2n}| = 2^{2^{2n}}$. Now let us give an upper bound on $e(n-1) = |\mathrm{COMM}_{2n,\Pi_{2n}}(n-1)|$. Each $f \in \mathrm{COMM}_{2n,\Pi_{2n}}(n-1)$ can be computed by a protocol using at most 2^{n-1} communications. Thus, there is a communication c corresponding to at least $2^{2n}/2^{n-1} = 2^{n+1}$ input assignments. It means that the submatrix $M(c)$ of the $2^n \times 2^n$ matrix $M(f, \Pi_{2n})$ has some size $a \times b$, where $a \cdot b \geq 2^{n+1}$, and so $a, b \geq 2$. We know that $M(c)$ must have rank 1, and without loss of generality we may assume that the rows of $M(c)$ are constant (i.e., each row consists either of 0's or 1's exclusively).

So each matrix $M(f, \Pi_{2n})$ for a function $f \in \mathrm{COMM}_{2n,\Pi_{2n}}(n-1)$ must contain an $a \times b$ submatrix $M(f)$ of rank 1 with $a \geq 2, b \geq 2$, and $ab \geq 2^{n+1}$, $a(b-1) < 2^{n+1}$, $(a-1)b < 2^{n+1}$ [if $M(c)$ has a size $a' \times b'$ with $a' \times b'$ "essentially" larger than 2^{n+1}, we delete some rows and columns to get a submatrix $M(f)$ of the size $a \times b$ with the above stated properties]. Let us bound the number of such matrices $M(f, \Pi_{2n})$.

The number of ways to select $M(f)$ in $M(f, \Pi_{2n})$ is

$$\sum_{j=2}^{2^n} Q_j, \quad \text{where} \quad Q_j = \binom{2^n}{j} \cdot \binom{2^n}{\lceil 2^{n+1}/j \rceil}$$

[the number of rows may be $j = 2, 3, \ldots, 2^n$ and there are $\binom{2^n}{j}$ ways to choose j rows from 2^n rows, etc.].

The number of different $j \times \lceil 2^{n+1}/j \rceil$ matrices with constant rows is

$$R_j = 2^j.$$

All elements in $M(f, \Pi_{2n})$ outside $M(f)$ may be chosen arbitrarily, yielding at most

$$2^{2^{2n} - 2^{n+1}} \text{ possibilities.}$$

So we obtain

$$e(n-1) = |\mathrm{COMM}_{2n,\Pi_{2n}}(n-1)| \leq 2^{2^{2n}-2^{n+1}} \sum_{j=2}^{2^n} Q_j \cdot R_j.$$

This implies

$$e(n-1)/|B_{2n}^2| \leq 2^{-2^{n+1}} \cdot \sum_{j=2}^{2^n} Q_j \cdot R_j.$$

By some combinatorial analysis it can be shown

$$2^{-2^{n+1}} \cdot Q_j \cdot R_j \in o(2^{-2^{n/2}})$$

for any $j = 2, \ldots, 2^n$ which implies

$$e(n-1)/|B_2^{2n}| \in o(2^n \cdot 2^{-2^{n/2}}).$$

This completes the proof. □

Next we study the closure properties of Boolean function classes determined by communication complexity.

Theorem 2.2.3.4 *Let h_1 and h_2 be two Boolean functions of $2n$ variables for some $n \in \mathbb{N}$. Let Π be a balanced partition of $2n$ variables. If $h_1 \in \mathrm{COMM}_{2n,\Pi}(r)$ and $h_2 \in \mathrm{COMM}_{2n,\Pi}(s)$ for some $r, s \in \mathbb{N}$, then*

(i) $h_1^C \in \mathrm{COMM}_{2n,\Pi}(r)$.

(ii) $h_1 \odot h_2 \in \mathrm{COMM}_{2n,\Pi}(r+s+1)$,
where $\odot$ is any Boolean operation of two variables.

Proof.

(i) If $D_{2n} = \langle \Pi, \Phi \rangle$ is a protocol computing h_1, then $D'_{2n} = \langle \Pi, \Phi' \rangle$ computing h_1^C is constructed as follows:

$$\begin{aligned}
\Phi'(\alpha, c) &= \Phi(\alpha, c) \quad \text{for any } \alpha \in \{0,1\}^n, c \in \{0,1,\$\}^* \\
&\qquad\qquad \text{such that } \Phi(\alpha, c) \in \{0,1\}^*, \\
\Phi'(\alpha, c) &= \bar{1} \quad \text{if } \Phi(\alpha, c) = \bar{0} \quad \text{for some } \alpha, c, \\
\Phi'(\alpha, c) &= \bar{0} \quad \text{if } \Phi(\alpha, c) = \bar{1} \text{ for some } \alpha, c.
\end{aligned}$$

Obviously, D'_{2n} uses the same communications as D_{2n}, i.e. $\mathrm{cc}(D'_{2n}) = \mathrm{cc}(D_{2n})$.

(ii) Without loss of generality, we may assume that there exist protocols $D^1_{2n} = \langle \Pi, \Phi^1 \rangle$ and $D^2_{2n} = \langle \Pi, \Phi^2 \rangle$ such that

(1) D^i_{2n} computes h_i for $i = 1, 2$,

(2) $\mathrm{cc}(D^1_{2n}) = r$ and $\mathrm{cc}(D^2_{2n}) = s + 1$,

(3) for both protocols D^1_{2n} and D^2_{2n} the left computer computes the output.

Then $D_{2n} = \langle \Pi, \Phi \rangle$, computing $h_1 \odot h_2$, simulates at once the communications of D^1_{2n} and D^2_{2n} (i.e., $\Phi(\alpha, \lambda) = \Phi^1(\alpha, \lambda) \cdot \Phi^2(\alpha, \lambda)$ if $\Phi^1(\alpha, \lambda) \in \{0,1\}^+$ or $\Phi^2(\alpha, \lambda) \in \{0,1\}^+$, etc.). The communication phase of D_{2n} finishes when all messages exchanged in D^1_{2n} and D^2_{2n} have been also exchanged in the computation of D_{2n}. After this, the left computer of D_{2n} knows both values $h_1(\gamma)$ and $h_2(\gamma)$ for the considered input γ. So the left computer computes $b = h_1(\gamma) \odot h_2(\gamma) \in \{0,1\}$ and returns the corresponding output $\bar{b} \in \{\bar{0}, \bar{1}\}$. □

Theorem 2.2.3.5 *For any $n \in \mathbb{N}$, any $k \in \{2, \ldots, n-1\}$, and any balanced partition Π of $2n$ variables, $\mathrm{COMM}_{2n,\Pi}(k)$ is not closed under $\vee$, i.e., $\exists f_1, f_2$ with $\mathrm{cc}(f_1, \Pi) \leq k, \mathrm{cc}(f_2, \Pi) \leq k$, and $\mathrm{cc}(f_1 \vee f_2, \Pi) > k$.*

Proof. Let h be a function in $B_2^n - \mathrm{COMM}_{2n,\Pi}(k)$ such that $|N^1(h)| = |\{\alpha \in \{0,1\}^{2n} \mid h(\alpha) = 1\}|$ is minimal for all functions in $B_n^2 - \mathrm{COMM}_{2n,\Pi}(k)$. Obviously, such an h must exist, because $B_2^n - \mathrm{COMM}_{2n,\Pi}(n-1) \neq \emptyset$. Since $\mathrm{cc}(h,\Pi) > k \geq 2, |N^1(h)| \geq 2$. Thus, we can write $N^1(h) = A \cup B$, for some A, B such that $A \cap B \neq \emptyset$ and $A \neq \emptyset$, $B \neq \emptyset$. Let h_1, h_2 be functions such that $N^1(h_1) = A$ and $N^1(h_2) = B$.

Obviously, $h = h_1 \vee h_2$, and $\mathrm{cc}(h_1,\Pi) \leq k, \mathrm{cc}(h_2,\Pi) \leq k$ because $|N^1(h_1)| < |N^1(h)|$, and $|N^1(h_2)| < |N^1(h)|$ (Note that h is a function with minimal $|N^1(h)|$ in $B_2^n - \mathrm{COMM}_{2n,\Pi}(k)$). □

Corollary 2.2.3.6 *For any $n \in \mathbb{N}$, any $k \in \{2,\ldots,n-1\}$, and any balanced partition Π of $2n$ variables, $\mathrm{COMM}_{2n,\Pi}(k)$ is not closed under any Boolean operation $\odot$ in $\{\wedge, \rightarrow\}$.*

Proof. We prove this fact by contradiction. Let $\mathrm{COMM}_{2n,\Pi}(k)$ be closed under some $\odot$ in $\{\wedge, \rightarrow\}$. Since $\mathrm{COMM}_{2n,\Pi}(k)$ is closed under complement (Theorem 2.2.3.4) and the disjunction $h_1 \vee h_2$ can be expressed by the operations $\odot$ and the complement (for instance, $h_1 \vee h_2 = (h_1^C \wedge h_2^C)^C$), the class $\mathrm{COMM}_{2n,\Pi}(k)$ is closed under disjunction, contradicting Theorem 2.2.3.5. □

Next we show that the upper bounds of (ii) of Theorem 2.2.3.4 can be tight. In what follows, let $\overline{\Pi}_{2n}$ denote the balanced partition of $2n$ variables $x_1,\ldots,x_{2n}$ into $\overline{\Pi}_{L,X} = \{x_1,\ldots,x_n\}$ and $\overline{\Pi}_{R,X} = \{x_{n+1},\ldots,x_{2n}\}$.

Lemma 2.2.3.7 *For any $n \in \mathbb{N}$, any $i,j \in \mathbb{N}$, $i+j \leq n$, there exist two Boolean functions h_1 and h_2 such that*

(i) $h_1 \in \mathrm{COMM}_{2n,\overline{\Pi}_{2n}}(i), h_2 \in \mathrm{COMM}_{2n,\overline{\Pi}_{2n}}(j)$, *and*

(ii) $h_1 \wedge h_2 \notin \mathrm{COMM}_{2n,\overline{\Pi}_{2n}}(i+j-1)$.

Proof. Let $X = \{x_1,\ldots,x_{2n}\}$, and let

$$h_1(x_1,\ldots,x_{2n}) = \bigwedge_{k=1}^{i} (x_k \equiv x_{n+k}),$$

$$h_2(x_1,\ldots,x_{2n}) = \bigwedge_{q=i+1}^{i+j} (x_q \equiv x_{n+q}).$$

Obviously $\mathrm{cc}(h_1,\overline{\Pi}_{2n}) = i$ and $\mathrm{cc}(h_2,\overline{\Pi}_{2n}) = j$ (see Example 2.2.2.4).
$h(x_1,\ldots,x_{2n}) = h_1(x_1,\ldots,x_{2n}) \wedge h_2(x_1,\ldots,x_{2n}) = \bigwedge_{r=1}^{i+j} (x_r \equiv x_{n+r})$. Again, it is clear that $\mathrm{cc}(h,\overline{\Pi}_{2n}) = i+j$. □

We close Section 2.2 by translating the above results about Boolean functions into results about computing problems (language recognition).

Definition 2.2.3.8 *Let $g : \mathbb{N} \to \mathbb{N}$ be a function with the property $g(m) \leq m/2$ for any $m \in \mathbb{N}$. Let $PI = \{\Pi_n\}_{n=1}^{\infty}$ be a sequence of balanced partitions, each Π_n a balanced partition of n variables. Then we define*

$$\mathrm{COMM}_{PI}(g) = \{L \subseteq \{0,1\}^* \mid \mathrm{cc}(h_n(L), \Pi_n) \leq g(n) \text{ for any } n \in \mathbb{N}\}.$$

Theorem 2.2.3.9 *Let $g_1, g_2 : \mathbb{N} \to \mathbb{N}$ be two functions, and let L_1 and L_2 be two languages over the alphabet $\{0,1\}$. Let PI be a sequence of balanced partitions. If $L_1 \in \mathrm{COMM}_{PI}(g_1)$ and $L_2 \in \mathrm{COMM}_{PI}(g_2)$, then*

(i) $(L_1)^C \in \mathrm{COMM}_{PI}(g_1)$, *and*

(ii) $L_1 \vee L_2, L_1 \wedge L_2, L_1 - L_2, (L_1 \vee L_2) - (L_1 \wedge L_2) \in \mathrm{COMM}_{PI}(g_1(n) + g_2(n) + 1)$.

Proof. The proof follows directly from Theorem 2.2.3.4. □

Theorem 2.2.3.10 *For any function $g : \mathbb{N} \to \mathbb{N}$ such that $2 \leq g(n) \leq n/2 - 1$ and any sequence $PI = \{\Pi_n\}_{n=1}^{\infty}$ of balanced partitions, $\mathrm{COMM}_{PI}(g)$ is not closed under union, intersection and symmetrical difference.*

Proof. The proof is a direct consequence of Theorem 2.2.3.5 and Corollary 2.2.3.6. □

2.2.4 Exercises

Exercise 2.2.4.1 *Let X and Y be the sets of input variables and output variables respectively. Estimate the cardinality of* $\mathrm{Bal}(X, Y)$.

Exercise 2.2.4.2 *Let $f(x_1, \ldots, x_n, z_1, \ldots, z_n) = \bigwedge_{i=1}^{n} (x_i \Rightarrow z_i)$, and let $X = \{x_1, \ldots, x_n, z_1, \ldots, z_n\}$. Find some Π_1 and $\Pi_2 \in \mathrm{Bal}(X)$ such that*

(i) $\mathrm{cc}(f, \Pi_1) = \min\{\mathrm{cc}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\}$,

(ii) $\mathrm{cc}(f, \Pi_2) = \max\{\mathrm{cc}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\}$.

Exercise 2.2.4.3 *Consider the language $Sm = \{xy \mid |x| = |y|, x, y \in \{0,1\}^+, \mathrm{BIN}(x) < \mathrm{BIN}(y)\}$. Prove that for any $n \in \mathbb{N}$, there exist a balanced partition Π_n such that $\mathrm{cc}(h_n(Sm), \Pi_n) \leq 1$.*

Exercise 2.2.4.4 *Prove that the largest monochromatic submatrix of the matrix $M(f_{2n}^{\mathrm{mod}}, \overline{\Pi})$ from Example 2.2.2.13 has at most 2^m elements.*

Exercise 2.2.4.5 *Let $\mathcal{A}$ be a δ-fooling set for some f and Π, $\delta \in \{0,1\}$. Prove that $\mathcal{A}$ is a $\overline{\delta}$-fooling set for f^C and Π.*

Exercise 2.2.4.6 *Give the formal proofs of the claims (1) and (2) presented in the proof of Lemma 2.2.2.27.*

Exercise 2.2.4.7 *Prove the assertion of Observation 2.2.2.30.*

Exercise 2.2.4.8 * *There also exists another communication protocol model for computing Boolean functions as well as that introduced in Definition 2.2.1.9. This protocol model is informally defined as follows:*
"We have again two computers and a partition partitioning the input bits to these two computers. The protocol $\langle \Pi, \phi, \Psi \rangle$ has two functions ϕ and Ψ. Ψ prescribes, depending on the present communication, which of the two computers is to send the next message. Each message consists of exactly one bit, and what this bit should be depends on ϕ, the input of the sender, and the present communication. The protocol terminates when one processor knows the output bit and the other one knows this about the first one. The complexity of a protocol is the number of bits communicated in the worst case".

Formalize this alternative definition of a protocol and prove that, for every f and Π, $\mathrm{cc}(f, \Pi)$ may differ at most by 1 when using this definition of a protocol instead of Definition 2.2.1.9.

Exercise 2.2.4.9 *Let $X = \{x_1, \ldots, x_n\}$ be a set of input variables. Let $\Pi \in \mathrm{Bal}(X)$ and $f \in B_2^n$. Let*

$$p = |\{f \in B_2^n \mid \mathrm{cc}(f, \Pi) = \log_2(\mathrm{rank}(M(f, \Pi)))\}| / 2^{2^n}$$

be the probability that the rankfix method provides an optimal lower bound for a randomly chosen Boolean function. Give a lower bound on p ($p \geq 1/4$ at least for sufficiently large n).

Exercise 2.2.4.10 * *Find a Boolean function f and a partition Π such that*

(i) there exists a large 1-fooling set for f and Π ,

(ii) all 0-fooling sets for f and Π are small.

How large can the difference between the cardinalities of the largest 1- and 0-fooling sets be?

Exercise 2.2.4.11 * *Theorem 2.2.2.28 shows that there exist an $f_n \in B_2^n$ and a balanced partition Π such that $\mathrm{cc}(f_n, \Pi) = n/2$ and $\log_2(\mathrm{Rank}(M(f_n, \Pi))) \leq ((\log_2 3)/4) \cdot n$. Use the construction idea of Theorem 2.2.2.28 in order to get a larger difference than the difference between $1/2$ and $(\log_2 3)/4$.*
[Hint: Start from a different basic matrix than M_1 of Figure 2.3. For instance, there exists a fooling matrix of size 16×16 and rank equal to 9]

Exercise 2.2.4.12 * *Prove $2^{-2^{n+1}} \cdot Q_j \cdot R_j = o(2^{-2^{n/2}})$ for every $j \in \{2, \ldots, 2^n\}$, where Q_j and R_j are defined as in Theorem 2.2.3.3.*
[Hint: Distinguish different cases according to j.]

Exercise 2.2.4.13 *Generalize the rankfix method to work also for many-output problems (not only for Boolean functions) and study analogously the properties of the generalized method.*

Exercise 2.2.4.14 *Prove that the largest monochromatic submatrix of the matrix* $M(f_{2n}^{\text{mod}}, \overline{\Pi})$ *from Example 2.2.2.13 has at most* 2^m *elements.*

Exercise 2.2.4.15 * *Let* $k, n \in \mathbb{N}$, $\delta \in \{0,1\}$. *For every* $f \in B_2^{2n}$ *and all partitions* Π *of the set of* $2n$ *variables we define a* **δ-fooling set of order k according to f and Π** *as a set* $\mathcal{A} = \{\alpha_1, \ldots, \alpha_m\} \subseteq \{0,1\}^{2n}$ *such that*

(i) $f(\alpha) = \delta$ *for all* $\alpha \in \mathcal{A}$, *and*

(ii) for any selection of $m = k+1$ *elements* $\alpha_1, \ldots, \alpha_m$ *from* $\mathcal{A}$ *the submatrix of* $M(f, \Pi)$ *corresponding to the intersection of the rows* $\alpha_{1\Pi,L}, \ldots, \alpha_{m\Pi,L}$ *and the columns* $\alpha_{1\Pi,R}, \ldots, \alpha_{m\Pi,R}$ *is not monochromatic.*

We set $\mathbf{Fool}_k^\delta(f, \Pi) = \max\{|\mathcal{A}| \mid \mathcal{A}$ *is a* δ*-fooling set of order* k *according to* f *and* $\Pi\}$, $\mathbf{Fool}^\delta(f, \Pi) = \max\{\text{Fool}_k^\delta(f, \Pi)/k \mid k \in \mathbb{N}\}$ *and* $\mathbf{Fooling}(f, \Pi) = \max\{\text{Fool}^0(f, \Pi), \text{Fool}^1(f, \Pi)\}$.
Prove that

$$\log_2(\text{Fooling}(f, \Pi)) \leq \text{cc}(f, \Pi) \leq (\log_2(\text{Fooling}(f, \Pi)) + \log_2(4n) + 1)^2,$$

i.e. that this generalized fooling set methods is able to provide lower bounds polynomially close to the communication complexity of concrete problems.

Exercise 2.2.4.16 *Prove that* $\text{COMM}_{2n,\Pi}(k)$ *is not closed under* $\oplus$ *and* $\equiv$ *for every* $k \in \{2, \ldots, n-1\}$, $n \in \mathbb{N}$, *and every balanced partition* Π *of* $2n$ *variables.*

2.2.5 Research Problems

Problem 2.2.5.1 ** *Prove or disprove that* $\text{cc}(f, \Pi)$ *is polynomially close to* $\log_2(\text{Rank}(f, \Pi))$ *for all* f *and* Π.

Problem 2.2.5.2 * *Find a infinite sequence of Boolean functions* $\{f_n\}_{n=1}^\infty$ *with a suitable sequence of partitions* $\{\Pi_n\}$ *such that*

$$\text{Fool}(f_n, \Pi_n) = \Omega((\text{Rank}(f_n, \Pi_n))^2)$$

or improve the assertions of Theorem 2.2.2.31.

Problem 2.2.5.3 *Find a concrete sequence of Boolean functions* $\{f_n\}_{n=1}^\infty$ *with* $\log_2(\text{Til}(f_n, \Pi)) = \Omega(n)$ *(or at least* $\Omega(n/log_2)$*) and* $\log_2(\text{Fool}(f_n, \Pi_n)) = O(\log_2 n)$ *for suitable* Π_n*'s. Note that Theorem 2.2.2.23 shows the existence of such a sequence of Boolean functions.*

Problem 2.2.5.4 *Find a concrete sequence of Boolean functions* $\{f_n\}_{n=1}^{\infty}$ *with* $\text{rank}(f_n, \Pi_n) = 2^{\Omega(n)}$ *and* $\text{Fool}(f_n, \Pi_n) = O(n)$ *for suitable* Π_n*'s. Note that Theorem 2.2.2.32 shows the existence of such a sequence of Boolean functions.*

Problem 2.2.5.5 * *Theorem 2.2.2.16 and Exercise 2.2.4.15 provide two lower bound proof methods which secure (for any problem) lower bounds on the communication complexity which are at least the root of the communication complexity. Find another method securing still closer lower bounds than these two methods.*

2.3 Communication Complexity

2.3.1 Introduction

In this section, the basic model for measuring communication complexity is presented. The general communication complexity of a computing problem is not considered as a communication complexity according to a fixed partition, but as the minimum of communication complexities over a class of partitions. The reason for considering such a model is that exactly this model has the most important applications in parallel and distributed computing. The idea is to prove lower bounds on the amount of resources (the number of processors, the time of the execution of a parallel algorithm, etc. ...) of a parallel computing model (computing a given problem), by proving that a lot of information exchange must be performed between some two parts of the parallel computing model, where the number of communication links between these two parts is restricted in a reasonable way. Mostly, we are able to find a partition of the parallel computing device into two parts in such a way that:

(i) the number of communication links between these parts is small in some reasonable sense,

(ii) the number of input variables entering one part "does not essentially differ" from the number of input variables entering the other part, and

(iii) no input variable enters both parts.

Neither do we know nor can we plan which variables enter the first part of the computing device and which the second part. So we know that the physical partition of the computing model corresponds to some balanced (or almost balanced) partition of input variables, but we do not know which one. Thus, if we want to claim that an amount of information bits must be exchanged between the two parts of the computing device, then we must know that at least this amount of bits must be used by each protocol $D_n = \langle \Pi, \Phi \rangle$ computing the problem considered for any Φ and any balanced (or almost balanced) partition Π.

In this way, we see that we need the communication complexity of a computing problem as the minimum over all Π and Φ such that $\langle \Pi, \Phi \rangle$ computes the problem.

In the next subsection we give the formal definition of communication complexity. In Section 2.3.3 we present the lower bound methods for this general communication complexity measure. Section 2.3.4 is devoted to some basic theoretical properties of communication complexity and Section 2.3.5 is devoted to the relation between communication complexity and the complexity of language recognition in the Chomsky hierarchy.

2.3.2 Definitions

We define communication complexity as the minimum over all communication complexities according to the partitions in $\mathrm{Bal}(X)$ or $\mathrm{Abal}(X)$.

Definition 2.3.2.1 *Let P_n^r be a computing problem of size n with the set of input variables X and the set of output variables Y. The* **communication complexity of P_n^r** *is*

$$\mathrm{cc}(P_n^r) = \min\{\mathrm{cc}(P_n^r, \Pi) \mid \Pi \in \mathrm{Bal}(X, Y)\} \ .$$

The **a-communication complexity of P_n^r** *is*

$$\mathrm{acc}(P_n^r) = \min\{\mathrm{cc}(P_n^r, \Pi) \mid \Pi \in \mathrm{Abal}(X, Y)\} \ .$$

Again, we give a special, separate definition of communication complexity of Boolean functions (one-output problems).

Definition 2.3.2.2 *Let f_n be a Boolean function of n variables $x_1, \ldots, x_n$, $X = \{x_1, \ldots, x_n\}$. The* **communication complexity of f_n** *is*

$$\mathrm{cc}(f_n) = \min\{\mathrm{cc}(f_n, \Pi) \mid \Pi \in \mathrm{Bal}(X)\} \ .$$

The **a-communication complexity of f_n** *is*

$$\mathrm{acc}(f_n) = \min\{\mathrm{cc}(f_n, \Pi) \mid \Pi \in \mathrm{Abal}(X)\} \ .$$

Now we give an example illustrating the possible difference between the communication complexity according to some fixed partition and the communication complexity defined above.

Example 2.3.2.3 We consider the "Equality problem", i.e., the language $Eq = \{w \in \{0,1\}^* \mid w = uu\}$. In Example 2.2.2.8 we have seen $\mathrm{cc}(h_{2m}(Eq), \overline{\Pi}) = m$ for any $m \in \mathbb{N}$ (due to the method rankfix). But, $\mathrm{cc}(h_{2m}(Eq)) = 1$ for any even $m \in \mathbb{N}$. To see this, let $X = \{x_1, \ldots, x_m, x_{m+1}, \ldots, x_{2m}\}$ be the set of the input variables of $h_{2m}(Eq)$. We set $D_{2m} = \langle \Pi, \Phi \rangle$, where

(i) Π : $\Pi_{L,X} = \{x_1, x_{m+1}, x_2, x_{m+2}, \ldots, x_{m/2}, x_{m+m/2}\}$,

(ii) Φ : $\Phi(\alpha_1 \ldots \alpha_{m/2}\alpha_{m+1} \ldots \alpha_{m+m/2}, \lambda) = 1$ if $\alpha_i = \alpha_{m+i}$ for all $i \in \{1, \ldots, m/2\}$;

$\Phi(\alpha_1 \ldots \alpha_{m/2}\alpha_{m+1} \ldots \alpha_{m+m/2}, \lambda) = 0$ if $\exists\, j \in \{1, \ldots, m/2\}$ such that $\alpha_j \neq \alpha_{m+j}$;

$\Phi(\alpha_{m/2+1} \ldots \alpha_m\alpha_{m+m/2+1} \ldots \alpha_{2m}, 1\$) = \bar{1}$ if $\alpha_i = \alpha_{m+i}$ for all $i \in \{m/2+1, \ldots, m\}$;

$\Phi(\alpha_{m/2+1} \ldots \alpha_m\alpha_{m+m/2+1} \ldots \alpha_{2m}, 1\$) = \bar{0}$ if $\exists\, j \in \{m/2+1, \ldots, m\}$ such that $\alpha_j \neq \alpha_{m+j}$;

$\Phi(\alpha_{m/2+1} \ldots \alpha_m\alpha_{m+m/2+1} \ldots \alpha_{2m}, 0\$) = \bar{0}$.

Obviously, D_{2m} computes $h_{2m}(Eq)$, and $\mathrm{cc}(h_{2m}(Eq)) \leq \mathrm{cc}(D_{2m}) = 1$. □

2.3.3 Lower Bound Methods

Since the general communication complexity is defined as the minimum over all partitions, the task of proving nontrivial lower bounds on communication complexity is much harder than that of getting lower bounds on communication complexity according to a fixed partition. Proving a lower bound on the communication complexity $\mathrm{cc}(P_n^r)$ of a computing problem P_n^r of size n means proving this lower bound on $\mathrm{cc}(P_n^r, \Pi)$ for any of the exponentially many balanced partitions Π. Following this and the lower bound proof methods foolfix, rankfix and tilingfix for communication complexity according to a fixed partition, we get the following methods for the complexity measures cc and acc defined in Definitions 2.3.2.1 and 2.3.2.2.

Method fool

Input: A problem P_n^r of size n with a set X of input variables and a set Y of output variables.

Step 1: For each $\Pi \in \mathrm{Bal}(X,Y)$ [$\Pi \in \mathrm{Abal}(X,Y)$], find a fooling set $\mathcal{A}(P_n^r, \Pi)$.

Step 2: Compute $d = \min\{|\mathcal{A}(P_n^r, \Pi)| \mid \Pi \in \mathrm{Bal}(X,Y)\}$
[$d = \min\{|\mathcal{A}(P_n^r, \Pi)| \mid \Pi \in \mathrm{Abal}(X,Y)\}$].

Output: "$\mathrm{cc}(P_n^r) \geq \lceil \log_2 d \rceil$" ["$\mathrm{acc}(P_n^r) \geq \lceil \log_2 d \rceil$"].

Method mrank

Input: A Boolean function f_n defined on a set X of input variables.

Step 1: For each $\Pi \in \mathrm{Bal}(X)$ [$\Pi \in \mathrm{Abal}(X)$], construct the matrix $M(f_n, \Pi)$.

Step 2: For some positive integer d, prove that

$$d \leq \min\{\text{Rank}(M(f_n, \Pi)) \mid \Pi \in \text{Bal}(X)\}$$

$$[d \leq \min\{\text{Rank}(M(f_n, \Pi)) \mid \Pi \in \text{Abal}(X)\}].$$

Output: "$\text{cc}(f_n) \geq \lceil \log_2 d \rceil$" ["$\text{acc}(f_n) \geq \lceil \log_2 d \rceil$"].

Method tiling

Input: A Boolean function f_n defined on a set X of input variables.

Step 1: For each $\Pi \in \text{Bal}(X)[\Pi \in \text{Abal}(X)]$, construct the matrix $\text{M}(f_n, \Pi)$.

Step 2: For some positive integer d, prove that

$$d \leq \min\{\text{Til}(M(f_n, \Pi)) \mid \Pi \in \text{Bal}(X)\}$$

$$[d \leq \min\{\text{Til}(M(f_n, \Pi)) \mid \Pi \in \text{Abal}(X)]\}]$$

Output: "$\text{cc}(f_n) \geq \lceil \log_2 d \rceil - 1$" ["$\text{acc}(f_n) \geq \lceil \log_2 d] \rceil - 1$"].

Obviously, none of the methods fool and mrank can be used automatically (algorithmically) to compute a lower bound on $\text{cc}(P_n^r)$ of a problem P_n^r of size n. For sufficiently large n, nobody is able to investigate the rank of exponentially many matrices according to n or to try to find exponentially many fooling sets (matrix covers), each for another partition. The problem of proving a lower bound becomes even more complicated if one wants to prove a lower bound on a general computing problem $\mathcal{P} = \{h_n\}_{n=1}^{\infty}$, i.e., if one wants to prove $\text{cc}(h_n) \geq g(n)$ for any $n \in \mathbb{N}$ and some function $g : \mathbb{N} \to \mathbb{N}$. The only way to realize this task is to find some crucial, inherent properties of the computed problem considered which allow to generally formulate some assertions about the fooling sets for the partitions under consideration or about the ranks (covers) of all matrices corresponding to partitions. We present some results of this kind now. We start with one-output problems.

We consider the language $\tilde{L} = \{\alpha \in \{0,1\}^+ \mid \#_0(\alpha) = \#_1(\alpha)\}$ of Example 2.2.2.1. We start with the language $\tilde{L}$ as a very simple example, because everybody can see that no balanced partition has any influence on the hardness of checking whether $\#_0(\alpha) = \#_1(\alpha)$ or not. Let $X_{2m} = \{x_1, \ldots, x_{2m}\}$ be the set of input variables of $h_{2m}(\tilde{L})$ for any $m \in \mathbb{N}$.

Theorem 2.3.3.1 *For any $m \in \mathbb{N}$ and any $\Pi \in \text{Bal}(X_{2m})$,*

$$\text{cc}(h_{2m}(\tilde{L})) = \text{cc}(h_{2m}(\tilde{L}), \Pi) = \lceil \log_2(m+1) \rceil \ .$$

Proof. Let us first show $\text{cc}(h_{2m}(\tilde{L}), \Pi) \leq \lceil \log_2(m+1) \rceil$ for any balanced partition Π. A protocol $D_{2m}\langle \Pi, \Phi \rangle$ computing $h_{2m}(\tilde{L})$ acts as follows. The first

computer sends the number i of 1's in its part of the input to the second computer, and the second computer only checks whether $m = i+$ the number of 1's in the part of the input of the second computer. Obviously, $i \in \{0,1,2,\ldots,m\}$, and the number of distinct binary words (messages) of length $\lceil \log_2(m+1) \rceil$ is at least $m+1 = |\{0,1,2,\ldots m\}|$. Thus, the submitted messages ($\mathrm{BIN}_k(i)$) all have length $k = \lceil \log_2(m+1) \rceil$ and $\mathrm{cc}(D_{2m}) = k = \lceil \log_2(m+1) \rceil$.

Now we prove $\mathrm{cc}(h_{2m}(\tilde{L}), \Pi) \geq k$ for any $\Pi \in \mathrm{Bal}(X_{2m})$. Let us consider the $2^m \times 2^m$ matrix $M(h_{2m}(\tilde{L}), \Pi)$. Let M' be the submatrix of $M(h_{2m}(\tilde{L}), \Pi)$ defined as the intersection of the $m+1$ rows corresponding to the input assignments 0^m, 10^{m-1}, 1^20^{m-2}, $\ldots$, $1^{m-1}0$, 1^m to the input variables in Π_L and of the $m+1$ columns corresponding to the input assignments 0^m, 10^{m-1}, 1^20^{m-2}, $\ldots$, $1^{m-1}0$, 1^m to the input variables in Π_R. Obviously, the $(m+1) \times (m+1)$ matrix M' contains exactly $m+1$ 1's for the $m+1$ input assignments $1^i0^j1^j0^i$ ($i+j = m$, $i,j \in \{0,\ldots,m\}$). Since 0^m, 10^{m-1}, $\ldots$, 1^m appear in this order in the sequence of row labels and also in the sequence of column labels of $M(h_{2m}(\tilde{L}), \Pi)$, all 1's of the matrix M' lie exactly on the diagonal leading from the lower left corner of M' to the upper right corner. Clearly, $\mathrm{rank}(M') = m+1$, and so $\mathrm{cc}(h_{2m}(\tilde{L}), \Pi) \geq \lceil \log_2(\mathrm{rank}(M(h_{2m}(\tilde{L}), \Pi))) \rceil \geq \lceil \log_2(\mathrm{rank}(M')) \rceil = \lceil \log_2(m+1) \rceil$. □

Let $L_\Delta = \{\alpha \in \{0,1\}^+ \mid |\alpha| = \binom{m}{2}$ for some $m \in \mathbb{N}$, and $G(\alpha)$ contains a triangle $\}$. We shall prove a linear lower bound ($\Omega(n)$) on the communication complexity of L_Δ. The proof technique we use in this case is based on the reduction of the task of proving a lower bound on $\mathrm{cc}(h_n(L_\Delta))$ to some easier task – of proving a lower bound on the communication complexity of some other computing problem according to a fixed partition.

Theorem 2.3.3.2 *For every* $n = \binom{m}{2}$, $m \in \mathbb{N}$,

$$\mathrm{cc}(h_n(L_\Delta)) \geq \frac{n}{8 \cdot 10^{10}} \ .$$

Proof. Let $X = \{x_1,\ldots,x_n\}$, $n = \binom{m}{2}$, be the set of input variables of $h_n(L_\Delta)$. Obviously, each variable x_i corresponds to some "potential" edge between some nodes u and v of G. If, for an input $\alpha = \alpha_1 \ldots \alpha_n$, $\alpha_i = 1$, then $G(\alpha)$ contains the edge (u,v). If $\alpha_i = 0$, then $G(\alpha)$ does not contain the edge (u,v). Let us now, for any $\alpha \in \{0,1\}^n$, view $G(\alpha)$ as a complete graph whose edges are labelled by zeros and ones. Let Π be an arbitrary balanced partition of X, and let Q_m denote the complete graph of m nodes. We color the edges of Q_m according to Π in the following way. Each edge corresponding to a variable in Π_L is red, and each edge corresponding to a variable in Π_R (i.e., any other edge) is blue. Since Π is balanced, the number of red edges $\overline{\mathrm{re}}$ is equal to the number of blue edges $\overline{\mathrm{bl}}$ (which is equal to $n/2$).

The informal idea of the proof is as follows. We shall find three pairwise disjoint sets B_1, B_2, $A \subseteq V(G(\alpha))$ such that

(i) $|B_1| = |B_2| \in \Omega(m)$, $|A| \in \Omega(m)$,

(ii) there is a matching $M \subseteq B_1 \times B_2$ with the cardinality $|M| \in \Omega(m)$, such that

(ii-1) for each $e = (v,u) \in M$, $\exists\, \overline{w}(e)$ triangles (v,u,x), $x \in A$, and exactly one of the edges (v,x) and (u,x) is red (the other one is blue),

(ii-2) $\sum_{e \in M} \overline{w}(e) \in \Omega(m^2) = \Omega(n)$.

This means we will find a set Δ of $\Omega(n)$ triangles, each having a special red edge in $(B_1 \cup B_2) \times A$ which is not included in any other triangle in Δ, and having one special blue edge in $(B_1 \cup B_2) \times A$ which is not included in any other triangle in Δ. Thus, considering graphs containing all matching edges of M (the variables assigned to these edges have the value 1) we have $\Omega(n) = \Omega(m^2)$ potential triangles such that the information about the existence of each of them is distributed to both computers (none of the two computers has the complete information about the existence of all three edges, because at least one edge is red (blue)). Additionally, the existence or non-existence of one triangle in Δ does not say anything about the existence of any other triangle in Δ. Clearly, to decide about the existence of a triangle from the set Δ of potential triangles, the communication protocol must exchange at least $|\Delta|$ bits. This fact will be shown by constructing a 0-fooling set of cardinality $2^{|\Delta|}$.

Now let us formalize the above idea.

A vertex is called **red** (**blue**) if less than $m/50$ of its $m-1$ incident edges are blue (red). All other vertices are called **mixed**.

First, we prove

(1) There are at least $m/100$ mixed vertices.

Suppose that fewer than $m/100$ mixed vertices exist. Let $\overline{\text{mix}}$, $\overline{\text{r}}$, $\overline{\text{b}}$ denote the number of mixed, red and blue vertices respectively. Since each red node is incident to at least $\frac{98}{100} \cdot m$ red edges, the number of red edges incident to red nodes (note that one edge may be incident to two nodes) must be at least

$$\frac{1}{2} \cdot \overline{\text{r}} \cdot \frac{98}{100} \cdot m \ .$$

Since the number of all red edges is $\overline{\text{re}} = \frac{m^2}{4} - \frac{m}{4}$, we get

$$\begin{aligned} \overline{\text{re}} = \frac{m^2}{4} - \frac{m}{4} &\geq \frac{1}{2} \cdot \overline{\text{r}} \cdot \frac{98}{100} \cdot m \\ \overline{\text{r}} &\leq \frac{50}{98} \cdot m - \frac{50}{98} \leq \frac{50}{98} \cdot m \ . \end{aligned}$$

The same consideration for $\overline{\text{b}}$ yields

$$\overline{\text{b}} \leq \frac{50}{98} \cdot m \ .$$

Now let us consider how to cover the $\overline{\text{r}} \cdot \frac{98}{100} \cdot m$ endpoints of the red edges incident to the red nodes. There are at most $\binom{\overline{\text{r}}}{2}$ edges among the red nodes

covering at most $2 \cdot \binom{\overline{\mathrm{r}}}{2}$ endpoints. The mixed vertices can cover at most $\overline{\mathrm{mix}} \cdot \frac{98}{100} \cdot m$ endpoints, and the blue vertices may cover at most $\overline{\mathrm{b}} \cdot \frac{2}{100} \cdot m$ endpoints. Thus, we get

$$\overline{\mathrm{r}} \cdot \frac{98}{100} \cdot m \leq 2 \cdot \binom{\overline{\mathrm{r}}}{2} + \overline{\mathrm{mix}} \cdot \frac{98}{100} \cdot m + \overline{\mathrm{b}} \cdot \frac{2}{100} \cdot m \ .$$

Since $\overline{\mathrm{mix}} < \frac{1}{100} \cdot m$ and $\overline{\mathrm{b}} \leq \frac{50}{98} \cdot m$ we get

$$\overline{\mathrm{r}} \cdot \frac{98}{100} \cdot m \leq \overline{\mathrm{r}}^2 - \overline{\mathrm{r}} + \frac{98}{10000} \cdot m + \frac{100}{9800} \cdot m \ .$$

This leads to

$$100 \cdot \overline{\mathrm{r}}^2 - (98 \cdot m + 100) \cdot \overline{\mathrm{r}} + 3 \cdot m \geq 0$$

which can be true only for $\overline{\mathrm{r}} > m/2$. The same consideration for the endpoints of blue edges incident with the blue nodes provides $\overline{\mathrm{b}} > m/2$. But this is impossible because $\overline{\mathrm{b}} + \overline{\mathrm{r}} \leq m$. Thus, (1) holds.

Let us select $m/100$ of the mixed nodes, and call them **top** nodes. Let $A = \{v_1, v_2, \ldots, v_{m/100}\}$ denote the set of top nodes. Let the nodes outside A be called **non-top** nodes, and let B denote the set $V(Q_m) - A$ of non-top nodes. Since each top node $v_i \in A$ is a mixed node there exist at least $m/100$ red edges leading from v_i to $m/100$ nodes in B, and at least $m/100$ blue edges leading from v_i to $m/100$ nodes in B. For each $i \in \{1, \ldots, m/100\}$, let $C_i = \{u \in B \mid (v_i, u) \text{ is red}\}$ and $D_i = \{u' \in B \mid (v_i, u') \text{ is blue}\}$. Let $E_i = C_i \times D_i$ be the set of all directed edges from nodes in C_i to nodes in D_i. Since $|C_i| \geq m/100$, and $|D_i| \geq m/100$ for any $i \in \{1, \ldots, m/100\}$ we get $|E_i| \geq m^2/10000$. We observe that each edge $(p, q) \in E_i$ corresponds to a triangle with the nodes v_i, p, q, where the edge (v_i, p) is red and the edge (v_i, q) is blue.

Note that $B = \bigcup_{i=1}^{m/100} (D_i \cup C_i)$, and set $E' = \bigcup_{i=1}^{m/100} E_i$. For each $e \in E'$, let the **weight** of e be $\overline{w}(e) = \max\{|\{i_1, \ldots, i_k\}| \mid i_j \in \{1, \ldots, m/100\} \text{ for } j \in 1, \ldots, k, k \in \mathbb{N}, \text{ and } e \in \bigcap_{j=1}^{k} E_{i_j}\}$, i.e., the number of sets E_i which contain e. We observe that $\overline{W} = \sum_{e \in E'} \overline{w}(e) = \sum_{i=1}^{m/100} |E_i| \geq m^3/1000000$.

Let us now consider the graph $G' = (B, E')$. We want to find a matching M of G' such that the sum of the weights of the matching edges is in $\Omega(m^2)$. Since $|E'| \leq \binom{|B|}{2} \leq m^2/2$, we get $\overline{W}/|E'| \geq m/500000$.

Let $E'_1 = \{e \in E' \mid \overline{w}(e) \geq m/10^6\}$.

Since every edge in $E' - E'_1$ has a weight smaller than one half of the average weight $\overline{W}/|E'|$, the sum of the weights of all edges in $E' - E'_1$ is at most $\overline{W}/2$ (the half of the weights of all edges in E'). Thus

$$\overline{w}(E'_1) = \sum_{e \in E'_1} \overline{w}(e) \geq \frac{\overline{W}}{2} \geq \frac{m^3}{2 \cdot 10^6}.$$

Since every edge $e \in E'$ (and so every edge $e \in E'_1$ too) has $\overline{w}(e)$ bounded by $m/100$

$$|E'_1| \geq \frac{\overline{w}(E'_1)}{m/100} \geq \frac{m^3/2 \cdot 10^6}{m/100} = \frac{m^2}{20000}.$$

Since $|E'_1| \geq m^2/20000$ we can find a matching $M \subseteq E'_1$ of the cardinality $|M| \geq |E'_1|/(2 \cdot (m-2)) \geq m/40000$. Consequently,

$$\overline{W_1} = \sum_{e \in M} \overline{w}(e) \geq m^2/(4 \cdot 10^{10}).$$

The matching M determines two disjoint sets of nodes B_1 and B_2 such that $M \subseteq B_1 \times B_2 \cup B_2 \times B_1$ (remember that the edges in E' are considered to be directed). Without loss of generality, we may assume that $\overline{W_2} = \sum_{e \in M'} \overline{w}(e) \geq m^2/(8 \cdot 10^{10})$ for the matching $M' = M \cap (B_1 \times B_2)$. Let $B'_1 \subseteq B_1$ and $B'_2 \subseteq B_2$ be the set of nodes with $M' \subseteq B'_1 \times B'_2$ and $|M'| = |B'_1| = |B'_2|$. We observe that then for each $e = (p, q) \in M'$, we have $\overline{w}(e)$ red edges incident to the nodes in A and $\overline{w}(e)$ blue edges incident to the top nodes in A. Thus, we have $d = \overline{W_2}$ triangles consisting of one red edge, one blue edge, and one matching edge from M'. Let $\{u_1, r_1, s_1\}$, $\{u_2, r_2, s_2\}$, ... , $\{u_d, r_d, s_d\}$ be all these triangles, where $\{u_1, \ldots, u_d\} \subseteq A$, $\{r_1, \ldots, r_d\} \subseteq B'_1$, $\{s_1, \ldots, s_d\} \subseteq B'_2$.

Next, let us consider some selected input assignments corresponding to the subclass of graphs defined as follows. We determine these graphs by

(i) fixing label 0 for some edges in Q_m (i.e., edges which do not appear in any graph of this subclass); let $\overline{E_0}$ denote the set of those edges,

(ii) fixing label 1 for some edges in Q_m (i.e., edges which exist in all graphs of the subclass); let $\overline{E_1}$ denote the set of edges with the fixed label 1.

All other edges are "free", i.e., they may be labelled by 0 or by 1. If the number of free edges is k, then we get a subclass of 2^k graphs.

Let the set of free edges be denoted by E_{free}.

We set $\overline{E_1} = M'$, $E_{\text{free}} = \{(u_i, r_i), (u_i, s_i) \mid i = 1, \ldots, d\}$, and $\overline{E_0} = E(Q_m) - (\overline{E_1} \cup E_{\text{free}})$. A graph of this subclass of graphs has a triangle iff $\exists\, i \in \{1, \ldots, d\}$ such that the red edge (u_i, r_i) and the blue edge (u_i, s_i) are labelled by 1. If $z_j \in X$ is the input variable corresponding to the edge (u_j, r_j) for $j \in \{1, \ldots, d\}$, and $y_k \in X$ is the input variable corresponding to the edge (u_k, s_k) for $k = 1, \ldots, d$, then a graph from this subclass contains a triangle iff

$$f(z_1, \ldots, z_d, y_1, \ldots, y_d) = \bigvee_{i=1}^{d} (z_i \wedge y_i) = 1 \ .$$

Since all z_i's are in $\Pi_{L,X}$ (note that (u_i, r_i)'s are red), and all y_i's are in $\Pi_{L,R}$ we get

$$\text{cc}(h_n(L_\Delta)) \geq \text{cc}(f(z_1, \ldots, z_d, y_1, \ldots, y_d), \overline{\Pi}), \text{ where } \overline{\Pi_L} = \{z_1, \ldots, z_d\} \ .$$

It remains to show that

$$\mathrm{cc}(f, \overline{\Pi}) \geq d \geq m^2/(8 \cdot 10^{10}) \geq n/(8 \cdot 10^{10}) \ .$$

To do so, we take the following set
$\mathcal{A}(f, \overline{\Pi}) = \{\alpha_1 \ldots \alpha_d \alpha'_1 \ldots \alpha'_d \mid \alpha_i \in \{0,1\}$ for any $i \in \{1, \ldots, d\}$, and $\alpha'_i = \alpha_i^{\mathbf{C}}$ for $i = 1, \ldots, d\}$.

We show that $\mathcal{A}(f, \overline{\Pi})$ is a 0-fooling set for f and $\overline{\Pi}$. Obviously, $f(\gamma) = 0$ for any $\gamma \in \mathcal{A}(f, \overline{\Pi})$. Let $\delta = \delta_1 \ldots \delta_d \delta_1^{\mathbf{C}} \ldots \delta_d^{\mathbf{C}}$ and $\beta = \beta_1 \ldots \beta_d \beta_1^{\mathbf{C}} \ldots \beta_d^{\mathbf{C}}$ be two words in $\mathcal{A}(f, \overline{\Pi})$, and $\delta \neq \beta$. Without loss of generality, we may assume that $\exists j$ that $\delta_j = 1$ and $\beta_j = 0$. Then $\delta_j \wedge \beta_j^{\mathbf{C}} = 1$, which implies

$$f(\overline{\Pi}^{-1}(\delta_{\overline{\Pi},L}, \beta_{\overline{\Pi},R})) = f(\delta_1 \ldots \delta_d \beta_1^{\mathbf{C}} \ldots \beta_d^{\mathbf{C}}) = 1 \ .$$

□

Theorem 2.3.3.2 provides a concrete example of a computing problem with linear communication complexity. Obviously, this example is only of theoretical importance, because the constant $8 \cdot 10^{10}$ is very large. But later we show linear lower bounds $d \cdot n$ for reasonable constants d.

Next we focus on the problem of how to prove lower bounds on the communication complexity of computing problems with several outputs. First, we will show how the lower bounds on communication complexity of such computing problems may be achieved by proving lower bounds on the communication complexity of some Boolean functions according to some subclasses of balanced partitions.

Observation 2.3.3.3 *Let $r, n \in \mathbb{N} - \{0\}$, and let $P_n^r = \{f_1, f_2, \ldots, f_r\}$ be a problem of size n with a set of input variables X. For each $i \in \{1, \ldots, r\}$, and each $\Pi \in \mathrm{Abal}(X)$,*

$$\mathrm{acc}(P_n^r, \Pi) \geq \mathrm{acc}(f_i, \Pi) \ .$$

Corollary 2.3.3.4 *Let $P_n^r = \{f_1, f_2, \ldots, f_r\}$ be a problem of size n for some $n, r \in \mathbb{N}$. Then*

$$\mathrm{acc}(P_n^r) \geq \max\{\mathrm{acc}(f_i) \mid i = 1, \ldots, r\} \ .$$

Theorem 2.3.3.5 *Let r, n be some positive integers, and let $P_n^r = \{f_1, f_2, \ldots, f_r\}$ be a problem of size n with a set of input variables X. Let $A_1, A_2, \ldots, A_r$ be subsets of $\mathrm{Bal}(X)$ such that $\bigcup_{i=1}^{r} A_i = \mathrm{Bal}(X)$. If, for each $i = 1, \ldots, r$, $\mathrm{cc}(f_i, A_i) = \min\{\mathrm{cc}(f_i, \Pi) \mid \Pi \in A_i\}$, then*

$$\mathrm{cc}(P_n^r) \geq \min\{\mathrm{cc}(f_i, A_i) \mid i = 1, \ldots, r\} \ .$$

Proof. Following Observation 2.3.3.3, for each $i = 1, \ldots, r$, we get $\mathrm{cc}(P_n^r, \Pi) \geq \mathrm{cc}(f_i, \Pi)$ for each $\Pi \in A_i$. Let Y be the set of output variables of P_n^r. Following Definition 2.3.2.1 of $\mathrm{cc}(P_n^r)$, we see that

$$\begin{aligned} cc(P_n^r) &= \min\{cc(P_n^r, \Pi) \mid \Pi \in \mathrm{Bal}(X,Y)\} \\ &= \min\{\min\{cc(P_n^r, \Pi) \mid \Pi \in \mathrm{Bal}(X,Y) \text{ and } \Pi \text{ restricted to } X \text{ is in} \\ &\qquad A_i\} \mid i = 1, \ldots, r\} \\ &\geq \min\{\min\{cc(f_i, \Pi) \mid \Pi \in A_i\} \mid i = 1, \ldots, r\} \\ &= \min\{cc(f_i, A_i) \mid i = 1, \ldots, r\}. \end{aligned}$$

□

Due to Theorem 2.3.3.5 the task of proving a lower bound on the communication complexity of a problem with several outputs may be easier than that of proving a lower bound on the communication complexity of a Boolean function. The idea is that we do not need to search directly for a fooling set $\mathcal{A}(P_n^r, \Pi)$ for each of the exponentially many partitions Π, but it is sufficient to show that for each $f_i \in P_n^r$ there is a subset of partitions A_i for which f_i is hard (i.e., $cc(f_i, A_i)$ is high), and that all A_i's together cover the set $\mathrm{Bal}(X)$ (or $\mathrm{Abal}(X)$). Note that this technique may lead to essentially higher lower bounds than the lower bound $\max\{cc(f_i) \mid i = 1, \ldots, r\}$ of Corollary 2.3.3.4, as shown in the following example.

Example 2.3.3.6 Let $P_{2m}^m = \{f_1, f_2, \ldots, f_m\}$ be a problem of size $2m$ defined on variables $x_1, \ldots x_{2m}$ as follows:

$$\begin{aligned} f_1(x_1, \ldots, x_{2m}) &= \bigwedge_{i=1}^{m} (x_i \equiv x_{m+i}) \\ f_2(x_1, \ldots, x_{2m}) &= \bigwedge_{i=1}^{m} (x_i \equiv x_{m+(i+1) \bmod m}) \\ &\vdots \\ f_j(x_1, \ldots, x_{2m}) &= \bigwedge_{i=1}^{m} (x_i \equiv x_{m+(i+j-1) \bmod m}) \\ &\vdots \\ f_m(x_1, \ldots, x_{2m}) &= \bigwedge_{i=1}^{m} (x_i \equiv x_{m+(i+m-1) \bmod m}). \end{aligned}$$

We show that for each $\Pi \in \mathrm{Bal}(\{x_1, \ldots, x_{2m}\})$, $\exists\, j \in \{1, \ldots, m\}$ such that $cc(f_j, \Pi) \geq m/4$, which yields $cc(P_{2m}^m) \geq m/4$. (Note that it is easy to observe that $cc(f_i) \leq 1$ for each $i = 1, \ldots, m$.)

We shall say that a pair of variables (x_r, x_s) is **compared** by a function f_k, if $x_r \equiv x_s$ occurs in the formula representation of f_k stated above. Obviously, for each $j = 1, \ldots, m$, it holds that f_j compares exactly m pairs. Additionally, if B_i denotes the set of all pairs compared by f_i for $i = 1, \ldots, m$, then $B_i \cap B_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^{m} B_i = \{(u, v) \mid u \in \{x_1, \ldots, x_m\}, v \in \{x_{m+1}, \ldots, x_{2m}\}\}$.

Now we define an $m \times m$ matrix $M = (m_{ij})_{i,j=1,\ldots,m}$ with columns labelled by $x_1, \ldots x_m$ and rows labelled by $x_{m+1}, \ldots, x_{2m}$ as follows: $m_{ij} = k$ for some $k \in \{1, \ldots, m\}$ iff (x_i, x_j) is compared by f_k. Following the above considerations, we see that each number from $\{1, \ldots, m\}$ appears exactly m times in M, and M looks like Figure 2.4.

	x_1	x_2	x_3	$\cdots$	x_{m-2}	x_{m-1}	x_m
x_{m+1}	1	m	$m-1$	$\cdots$	4	3	2
x_{m+2}	2	1	m	$\cdots$	5	4	3
x_{m+3}	3	2	1	$\cdots$	7	5	4
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$
x_{2m-1}	$m-1$	$m-2$	$m-3$	$\cdots$	2	1	m
x_{2m}	m	$m-1$	$m-2$	$\cdots$	3	2	1

Fig. 2.4.

Let Π be an arbitrarily balanced partition of $\{x_1, \ldots, x_{2m}\}$. Obviously, either Π_L contains at least $\lceil m/2 \rceil$ variables of $\{x_1, \ldots, x_m\}$ or Π_R contains at least $\lceil m/2 \rceil$ variables of $\{x_1, \ldots, x_m\}$. Without loss of generality we assume that Π_L does. Then Π_R contains at least $\lceil m/2 \rceil$ variables of $\{x_{m+1}, \ldots, x_{2m}\}$. Let M' be a $d_1 \times d_2$ submatrix of M defined by the column labels $\Pi_L \cap \{x_1, \ldots, x_m\}$ and by the row labels $\Pi_R \cap \{x_{m+1}, \ldots, x_{2m}\}$. Since $d_1 \geq \lceil m/2 \rceil$ and $d_2 \geq \lceil m/2 \rceil$, the number of elements in M' is at least $d_1 \cdot d_2 \geq m^2/4$. Since the elements of M' are numbers $1, 2, \ldots, m$, there exists $j \in \{1, \ldots, m\}$ appearing at least $\frac{m}{4}$ times in M'. Thus, we have $\frac{m}{4}$ pairs $(x_{i_1}, x_{r_1}), \ldots, (x_{i_{m/4}}, x_{r_{m/4}})$ [note that $r_k = m + (i_k + j - 1) \bmod m$] compared by f_j with the property $\{x_{i_1}, \ldots, x_{i_{m/4}}\} \subseteq \Pi_L$ and $\{x_{r_1}, \ldots, x_{r_{m/4}}\} \subseteq \Pi_R$. Following Example 2.2.2.4 and the fact that

$$f_j(x_1, \ldots, x_{2m}) = (\bigwedge_{i=1}^{m/4} (x_{i_d} \equiv x_{r_d})) \wedge f(x_1, \ldots x_{2m}),$$

where $f(x_1, \ldots, x_m) = 1$ for the setting $x_z = 1$ for all $z \in \{1, \ldots, 2m\} - \{i_1, \ldots, i_{m/4}, r_1, \ldots, r_{m/4}\}$, we immediately see that $cc(f_j, \Pi) \geq m/4$. □

2.3.4 Theoretical Properties of Communication Complexity

The results presented in this and the next subsection help us to understand the nature of communication complexity beyond the way in which communication complexity is applied to real computing devices. So these two subsections may be skipped by anybody interested primarily in the relations to real parallel computations.

As in the case of communication complexity according to a fixed partition, we shall define the communication complexity classes and investigate their properties.

Definition 2.3.4.1 *We set for any* $n, m \in \mathbb{N}$, $m \leq n/2$, $\mathbf{COMM_n(m)} = \{f \in B_2^n \mid \mathrm{cc}(f) \leq m\}$ *as the set of all Boolean functions of* n *variables computable within communication complexity* m.

Definition 2.3.4.2 *Let* $g : \mathbb{N} \to \mathbb{N}$ *be a function with the property* $g(n) \leq n/2$ *for any* $n \in \mathbb{N}$. *We set* $\mathbf{COMM(g(n))} = \{L \subseteq \{0,1\}^* \mid \mathrm{cc}(h_n(L)) \leq g(n)$ *for any* $n \in \mathbb{N}\}$.

We first study the hierarchy of the communication complexity classes. We start by showing that almost all Boolean functions of $2n$ variables have the highest possible communication complexity n.

Lemma 2.3.4.3 $|\mathrm{COMM}_{2n}(n-1)| \in o(2^{2^{2n}})$.

Proof. We have proved in Theorem 2.2.3.3 that for any balanced partition Π $e(n-1) = |\mathrm{COMM}_{2n,\Pi}(n-1)| \in o(2^n \cdot 2^{-2^{n/2}} \cdot 2^{2^{2n}})$. Since $|\mathrm{COMM}_{2n}(n-1)| \leq \sum_{\Pi \in \mathrm{Bal}(X)} |\mathrm{COMM}_{2n,\Pi}(n-1)|$, and $|\mathrm{Bal}(X)| = \binom{2n}{n} \leq 2^{2n}$ we get

$$|\mathrm{COMM}_{2n}(n-1)| = o(2^{3n} \cdot 2^{-2^{n/2}} \cdot 2^{2^{2^n}}).$$

□

Corollary 2.3.4.4 *For any natural number* n

$$\mathrm{COMM}_{2n}(n-1) \subsetneq \mathrm{COMM}_{2n}(n) = B_2^{2n},$$

i.e., $\exists f \in B_2^{2n}$ *which belongs to* $\mathrm{COMM}_{2n}(n) - \mathrm{COMM}_{2n}(n-1)$.

Now we extend this result in order to show that one additional communication bit always increases the power of communication protocols.

Lemma 2.3.4.5 *For any* $m, n \in \mathbb{N}$, $m \leq n$, $B_2^{2n}(n+m) \subseteq \mathrm{COMM}_{2n}(m)$.

Proof. Let f be a Boolean function of $2n$ variables which essentially depends on exactly $n+m$ variables. ($n-m$ variables are dummies for f). Then f can be computed by a protocol $D_{2n} = \langle \Pi, \Phi \rangle$, where

(1) Π_L contains all $n-m$ dummy variables of f,

(2) Φ is defined in such a way that the left computer sends the values of its m essential variables for f to the right computer, and the right computer, knowing the values of all $n+m$ essential variables of f, computes the output.

So each $f \in B_2^{2n}$ essentially depending on at most $n+m$ variables is in $\mathrm{COMM}_{2n}(m)$. □

Corollary 2.3.4.6 *For any $m, n \in \mathbb{N}$, $m \leq n$,*

$$|\mathrm{COMM}_{2n}(m)| \geq 2^{2^{n+m}} \ .$$

Proof. $|B_2^{2n}(n+m)| \geq 2^{2^{n+m}}$. □

Lemma 2.3.4.7 *For any m, $n \in \mathbb{N}$, $m \leq n$,*

$$|B_2^{2n}(n+m) \cap \mathrm{COMM}_{2n}(m-1)| = o(2^{2^{n+m}}) \ .$$

Proof. To get this result, we modify the proof of Theorem 2.2.3.3. Let $f \in B_2^{2n}(n+m)$, and let f be defined on the set of variables $X = \{x_1, \ldots, x_{2n}\}$. Let Π be any balanced partition of X. Obviously, f can be unambiguously described by a matrix $\overline{M}(f, \Pi)$ such that $\overline{M}(f, \Pi)$ is a submatrix of $M(f, \Pi)$, and $\overline{M}(f, \Pi)$ has 2^{n+m} elements. Now we estimate the number of Boolean functions $f \in B_2^{2n}(n+m) \cap \mathrm{COMM}_{2n,\Pi}(m-1)$.

Each $f \in \mathrm{COMM}_{2n,\Pi}(m-1)$ can be computed by a protocol using at most 2^{m-1} different communications. Thus, there is a communication c corresponding to a submatrix $M(c)$ of $\overline{M}(f, \Pi)$ such that $M(c)$ contains at least $2^{n+m}/2^{m-1} = 2^{n+1}$ elements.

Following exactly the steps of the proof of Theorem 2.2.3.3 we get that the number of matrices $\overline{M}(f, \Pi)$ for $f \in B_2^{2n}(n+m) \cap \mathrm{COMM}_{2n,\Pi}(m-1)$ is limited by

$$2^{2^{n+m}-2^{n+1}} \cdot \sum_{j=2}^{2^n} Q_j R_j \ .$$

Since we know that $2^{-2^{n+1}} \cdot Q_j \cdot R_j \in o(2^{-2^{n/2}})$ for any $j = 2, \ldots, 2^n$ we get

$$|B_2^{2n}(n+m) \cap \mathrm{COMM}_{2n,\Pi}(m-1)| \leq 2^{2^{n+m}} \cdot o(2^n \cdot 2^{-2^{n/2}})$$

for any balanced partition Π. Since

$$|\mathrm{COMM}_{2n}(m-1)| \leq \sum_{\Pi \in \mathrm{Bal}(X)} |\mathrm{COMM}_{2n,\Pi}(m-1)|,$$

we get

$$|B_2^{2n}(n+m) \cap \mathrm{COMM}_{2n}(m-1)| \leq 2^{2^{n+m}} \cdot o(2^n \cdot 2^{2n} \cdot 2^{-2^{n/2}}) \in o(2^{2^{n+m}}) \ .$$

□

Thus, we get an important result providing a strong hierarchy of the communication complexity classes.

Theorem 2.3.4.8 *For any* $m, n \in \mathbb{N}$, $m \leq n$,

$$\mathrm{COMM}_{2n}(m-1) \subsetneq \mathrm{COMM}_{2n}(m) \ .$$

Proof. Lemma 2.3.4.5 claims that every $f \in B_2^{2n}(n+m)$ is also in $\mathrm{COMM}_{2n}(m)$, and Lemma 2.3.4.7 claims that almost all functions from $B_2^{2n}(n+m)$ do not belong to $\mathrm{COMM}_{2n}(m-1)$. □

The consequence of Theorem 2.3.4.8 for language classes is included in the next assertion.

Theorem 2.3.4.9 *Let* $g : \mathbb{N} \to \mathbb{N}$ *be a function with the property* $1 \leq g(n) \leq n/2$ *for any* $n \in \mathbb{N}$. *Then*

$$\mathrm{COMM}(g(n)-1) \subsetneq \mathrm{COMM}(g(n)) \ .$$

The rest of Section 2.3.4 is devoted to the closure properties of communication complexity classes. First, we make the following trivial observation.

Observation 2.3.4.10 *The classes* $\mathrm{COMM}_{2n}(m)$ *and* $\mathrm{COMM}(g(n))$ *are closed under complement for any* $m, n \in \mathbb{N}$, $m \leq n$ *and any function* $g(n) : \mathbb{N} \to \mathbb{N}$, $g(n) \leq n/2$.

We shall show that communication complexity is "extremly unclosed" under disjunction and conjunction. Namely, for any $n \in \mathbb{N}$ we show that there exist two Boolean functions f_1 and f_2 of $2n$ variables, such that $\mathrm{cc}(f_1) = 0$, $\mathrm{cc}(f_2) = 1$, and $\mathrm{cc}(f_1 \vee f_2) \geq dn$ for some constant d independent of n. Obviously, this surprisingly differs from the properties of communication complexity according to a fixed partition, for which $\mathrm{cc}(f_1 \vee f_2, \Pi) \leq \mathrm{cc}(f_1, \Pi) + \mathrm{cc}(f_2, \Pi) + 1$ holds for any functions f_1, f_2 and any balanced partition Π. Let us now explain the reason for this difference. Since $\mathrm{cc}(f_1)$ is defined as the minimum of $\mathrm{cc}(f_1, \Pi)$ over all balanced partitions, $\mathrm{cc}(f_1)$ can be small, because there is some partition Π for which $\mathrm{cc}(f_1, \Pi)$ is small. But if $\mathrm{cc}(f_1, \Pi)$ is small for partitions Π for which $\mathrm{cc}(f_2, \Pi)$ is large, and if $\mathrm{cc}(f_2, \Pi')$ is small for partitions Π' for which $\mathrm{cc}(f_1, \Pi')$ is large, then one can get that $\mathrm{cc}(\{f_1, f_2\})$ is large. If f_1 and f_2 are convieniently constructed, a large $\mathrm{cc}(\{f_1, f_2\})$ also implies a large $\mathrm{cc}(f_1 \vee f_2)$. Before proving the existence of such functions, we prove two technical lemmas.

Lemma 2.3.4.11 *Let* $f \in B_2^{2n}$ *have the maximal possible communication complexity (i.e.,* $\mathrm{cc}(f) = n$*). Then* $\mathrm{acc}(f) \geq \lfloor 2n/3 \rfloor - 1$.

Proof. We prove this fact by contradiction. Let $f \in B_2^{2n}$, $\mathrm{cc}(f) = n$, and let $\mathrm{acc}(f) < \lfloor 2n/3 \rfloor - 1$. Then there is an almost balanced partition Π' such that $\mathrm{acc}(f, \Pi') < \lfloor 2n/3 \rfloor - 1$, i.e., there is a protocol $D'_{2n} = \langle \Pi', \Phi' \rangle$ computing f and $\mathrm{cc}(D'_{2n}) < \lfloor 2n/3 \rfloor - 1$.

Now we describe the construction of another protocol $D_{2n} = \langle \Pi, \Phi \rangle$ computing f with a balanced partition Π and working with $\mathrm{cc}(D_{2n}) < n$. Without loss of generality, we may assume that $|\Pi'_L| \geq |\Pi'_R|$. Let $\Pi'_L = \{x_1, \ldots, x_{n+i}\}$, $\Pi'_R = \{z_1, \ldots, z_{n-i}\}$ for some $i \leq n/3$. We set $\Pi_L = \{x_1, \ldots, x_n\}$ and $\Pi_R = \{x_{n+1}, \ldots, x_{n+i}, z_1, \ldots z_{n-i}\}$. Now let D_{2n} act as follows. First, the left computer sends 0 (start of the communication) to the right computer. Then the right computer sends the values of all variables $x_{n+1}, \ldots, x_{n+i}$ to the left computer. After this information exchange of $i+1$ bits the left computer knows the values of all input variables of Π'_L, and the right computer knows the values of all input variables in $\Pi_R \supseteq \Pi'_R$. So D_{2n} can simulate the communication of D'_{2n} in order to compute the output. This implies $\mathrm{cc}(D_{2n}) = i + 1 + \mathrm{cc}(D'_{2n}) < i + 1 + \lfloor 2n/3 \rfloor - 1 \leq n/3 + \lfloor 2n/3 \rfloor \leq n$. □

Lemma 2.3.4.12 *Let f and f' be Boolean functions defined over the same set of variables X. Let r be a variable which does not belong to X. Then, for the functions $g = r \wedge f$ and $g' = r^{\mathbf{C}} \wedge f'$,*

$$\mathrm{cc}(g \vee g', \Pi) \geq \max\{\mathrm{cc}(g, \Pi), \mathrm{cc}(g', \Pi)\} - 2$$

for every $\Pi \in \mathrm{Bal}(X \cup \{r\})$.

Proof. Let Π be a balanced partition of the set $X \cup \{r\}$. Let $D = \langle \Pi, \Phi \rangle$ be a protocol computing $g \vee g'$ with the property $\mathrm{cc}(D) = \mathrm{cc}(g \vee g', \Pi)$. We describe a protocol $\overline{D} = \langle \Pi, \overline{\Phi} \rangle$ computing g with $\mathrm{cc}(\overline{D}) = \mathrm{cc}(D) + 2$. The first two rounds are used to secure that both computers know the value of the variable r. Obviously, the number of exchanged bits in the first two rounds is exactly two. Now, if the value of r is zero, both computers know that the result is 0 (note that $g = r \wedge f$). If the value of r is one, then $g(\alpha) = g(\alpha) \vee g'(\alpha)$ for the actual input assignment α, because $g'(\alpha) = 0$. Then $\overline{D}$ simulates the computation of D on α to compute $g(\alpha) \vee g'(\alpha) = g(\alpha)$. Thus, $\mathrm{cc}(\overline{D}) = \mathrm{cc}(D) + 2$, and $\mathrm{cc}(g, \Pi) \leq \mathrm{cc}(D) + 2$. Realizing the same consideration for the function g', we also get $\mathrm{cc}(g', \Pi) \leq \mathrm{cc}(D) + 2 = \mathrm{cc}(g \vee g', \Pi) + 2$, which completes the proof. □

Now we are ready to prove the main non-closure property. Let $X = \{x_1, \ldots, x_{2n-2}, z_1, \ldots, z_{2n-2}, r_1, r_2, r_3, r_4\}$ be a set of input variables of the Boolean functions $f_1, f_2 \in B_2^{4n}$. Let $f \in B_2^{2n-2}$ be a function defined on the variables $x_1, \ldots, x_{2n-2}$ with $\mathrm{cc}(f) = n - 1$ (such a function f must exist according to Cororally 2.3.4.4). We define

$$f_1(x_1, \ldots, x_{2n-2}, z_1, \ldots, z_{2n-2}, r_1, \ldots, r_4) = r_1 \wedge f(x_1, \ldots, x_{2n-2})$$

$$f_2(x_1, \ldots, x_{2n-2}, z_1, \ldots, z_{2n-2}, r_1, \ldots, r_4) = r_1^{\mathbf{C}} \wedge \left(\bigwedge_{i=1}^{2n-2} (x_i \equiv z_i) \right) .$$

Note that r_2, r_3, r_4 are dummy variables for f_1 and f_2.

Theorem 2.3.4.13 *Let $f_1, f_2 \in B_2^{4n}$ be functions as defined above. Then*

(i) $\mathrm{cc}(f_1) = 0$, $\mathrm{cc}(f_2) = 1$, *and*

(ii) $\mathrm{cc}(f_1 \vee f_2) \geq 2n/3 - 6$.

Proof. Taking the partition Π^1 with $\Pi^1_L = \{x_1, \ldots, x_{2n-2}, r_1, r_2\}$ we immediately get $\mathrm{cc}(f_1) \leq \mathrm{cc}(f_1, \Pi^1) = 0$. Taking the partition Π^2 with $\Pi^2_L = \{x_1, z_1, x_2, z_2, \ldots, x_{n-1}, z_{n-1}, r_1, r_2\}$ it is easy to see that $\mathrm{cc}(f_2) \leq \mathrm{cc}(f_2, \Pi^2) = 1$.

Now let us prove *(ii)*. Following Lemma 2.3.4.12 we have $\mathrm{cc}(f_1 \vee f_2, \Pi) \geq \max\{\mathrm{cc}(f_1, \Pi), \mathrm{cc}(f_2, \Pi)\} - 2$ for any $\Pi \in \mathrm{Bal}(X)$. Thus, to prove $\mathrm{cc}(f_1 \vee f_2) \geq 2n/3 - 6$, it is sufficient to prove that for every $\Pi \in \mathrm{Bal}(X)$, $\max\{\mathrm{cc}(f_1, \Pi), \mathrm{cc}(f_2, \Pi)\} \geq 2n/3 - 4$. We distinguish the following two possibilities according to the partition Π:

(1) $|\Pi_L \cap \{x_1, \ldots, x_{2n-2}\}| \leq \lfloor (2n-2)/3 \rfloor \cdot 2$ and
$|\Pi_R \cap \{x_1, \ldots, x_{2n-2}\}| \leq \lfloor (2n-2)/3 \rfloor \cdot 2$,

(2) $|\Pi_L \cap \{x_1, \ldots, x_{2n-2}\}| \geq 2 \cdot \lfloor (2n-2)/3 \rfloor$ or
$|\Pi_R \cap \{x_1, \ldots, x_{2n-2}\}| \geq 2 \cdot \lfloor (2n-2)/3 \rfloor$.

(1) If $X' = \{x_1, \ldots, x_{2n-2}\}$, $|\Pi_L \cap X'| \leq \lfloor (2n-2)/3 \rfloor \cdot 2$, and $|\Pi_R \cap X'| \leq \lfloor (2n-2)/3 \rfloor \cdot 2$, then Π' defined as $\Pi'_L = \Pi_L \cap X'$ and $\Pi'_R = \Pi_R \cap X'$ is an almost balanced partition of X'. Since $\mathrm{cc}(f_1, \Pi) \geq \mathrm{cc}(f_1, \Pi') - 2$ (see Lemma 2.3.4.12), we get from Lemma 2.3.4.11 (note that $\Pi' \in \mathrm{Abal}(X')$) $\mathrm{cc}(f_1, \Pi) \geq \mathrm{cc}(f, \Pi) - 2 \geq \lfloor (2n-2)/3 \rfloor - 1 - 2 \geq 2n/3 - 4$. Thus, we have for any $\Pi \in \mathrm{Bal}(X)$ with property (1) $\mathrm{cc}(f_1 \vee f_2, \Pi) \geq 2n/3 - 6$.

(2) Let $\Pi \in \mathrm{Bal}(X)$ have the property (2). Without loss of generality, we may assume that $|\Pi_L \cap \{x_1, \ldots, x_{2n-2}\}| \geq 2 \cdot \lfloor (2n-2)/3 \rfloor$. Since Π is balanced, $|\Pi_R \cap \{z_1, \ldots, z_{2n-2}\}| \geq 2n - 4 - |\Pi_R \cap \{x_1, \ldots, x_{2n-2}\}| \geq 2 \cdot \lfloor (2n-2)/3 \rfloor - 2$. Then there exist at least

$$k = 2 \cdot \lfloor (2n-2)/3 \rfloor + 2 \cdot \lfloor (2n-2)/3 \rfloor - 2 - (2n-2) \geq 2n/3 - 2$$

positive integers $i_1, i_2, \ldots, i_k \in \{1, 2, \ldots, 2n-2\}$ such that

(i) $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\} \subseteq \Pi_L$,

(ii) $\{z_{i_1}, z_{i_2}, \ldots, z_{i_k}\} \subseteq \Pi_R$,

(iii) $f_2(x_1, \ldots, x_{2n-2}, z_1, \ldots, z_{2n-2}, r_1, \ldots, r_4) =$
$\bigwedge_{j=1}^{k} (x_{i_j} \equiv z_{i_j}) \wedge f_2'(x_1, \ldots, x_{2n-2}, z_1, \ldots, z_{2n-2}, r_1, \ldots, r_4)$,
where $f_2'(x_1, \ldots, x_{2n-2}, z_1, \ldots, z_{2n-2}, r_1, r_2, r_3, r_4) = 1$
for each input assignment with $r_1 = 0$ and $u = 1$ for each $u \in X - \{r_1, x_{i_1}, \ldots, x_{i_k}, z_{i_1}, \ldots, z_{i_k}\}$.

Using our standard proof scheme, we see that (i) $\wedge$ (ii) $\wedge$ (iii) implies $cc(f_2, \Pi) \geq k$. Thus, for each $\Pi \in Bal(X)$ with the property (2)

$$cc(f_1 \vee f_2, \Pi) \geq cc(f_2, \Pi) - 2 \geq 2n/3 - 4.$$

□

Corollary 2.3.4.14 *For the functions $f_1, f_2 \in B_2^{4n}$ of Theorem 2.3.4.13*

$$cc(\{f_1, f_2\}) \geq 2n/3 - 8 \ .$$

Proof. Each protocol computing $\{f_1, f_2\}$ can be modified by using 2 additional communication bits to secure that both computers know boths output values (for f_1 and f_2) at the end of the computation.

Obviously, this means that each computer knows $f_1 \vee f_2$ at the end of the computation, i.e., $cc(\{f_1, f_2\}, \Pi) + 2 \geq cc(f_1 \vee f_2, \Pi)$ for any $\Pi \in Bal(X)$. □

Next, we show the strong non-closure property of the conjunction.

Theorem 2.3.4.15 *Let $f_1, f_2 \in B_2^{4n}$ be the functions considered in Theorem 2.3.4.13. Then*

(i) $cc(f_1^C) = 0$, $cc(f_2^C) = 1$, *and*

(ii) $cc(f_1^C \wedge f_2^C) \geq 2n/3 - 6$.

Proof. Since $COMM_{4n}(m)$ is closed under complement for any $m \in \mathbb{N}$, (i) is obvious. For the same reason (Observation 2.3.4.10) $cc((f_1 \vee f_2)^C) \geq 2n/3 - 6$. The fact $(f_1 \vee f_2)^C = f_1^C \wedge f_2^C$ completes the proof. □

The interpretation of the results of Theorem 2.3.4.13 and Theorem 2.3.4.15 for formal languages can be formulated as follows.

Theorem 2.3.4.16 *There exist two languages L_1 and L_2 such that*

(i) $L_1 \in COMM(0), L_2 \in COMM(1)$, *and*

(ii) $L_1 \cup L_2 \notin COMM(n/6 - 7)$
$L_1^C \cap L_2^C \notin COMM(n/6 - 7)$.

Proof. It is sufficient to consider the languages L_1 and L_2 defined as follows: For any $n = 4m$, $h_n(L_1) = f_1$ and $h_n(L_2) = f_2$, for $f_1, f_2 \in B_2^n$. □

Note that our proof of Theorem 2.3.4.13 (as of Theorem 2.3.4.15 and 2.3.4.16) is existential. Further below, we give a constructive proof of this non-closure property, yielding interesting practical consequences. The practical importance of these strong non-closure properties of communication complexity

lies in the fact that by using communication complexity we shall be able to show such strong non-closure properties for real complexity measures of real circuit models. Thus, knowing that the circuit complexity of $f = f_1 \vee f_2$ is much greater than the sum of the circuit complexities of f_1 and f_2 helps the circuit designers to decide not to design the circuit for f (or for $\{f_1, f_2\}$), but to design two circuits, one for f_1 and one for f_2. If $f_1 \vee f_2$ has to be computed, this can be done by software.

2.3.5 Communication Complexity and Chomsky Hierarchy

In this section we study how hard the languages of the Chomsky hierarchy are for communication complexity. We show that regular languages have small (constant) communication complexity, but already the context-free languages (the second level of the Chomsky hierarchy) may be the hardest ones for communication complexity (linear communication complexity is required). Furthermore, we shall show that there are also languages which are the hardest ones in the Chomsky hierarchy but lie in COMM(0). This result is not surprising, because all languages of the Chomsky hierarchy are described uniformly by devices with a finite description and the protocols are non-uniform computing devices (i.e., each language is described by a potentially infinite sequence of protocols which means that in this way we can also describe languages for which no finite descriptions exist).

First, we study the communication complexity of regular languages.

Theorem 2.3.5.1 *For each regular language $L \subseteq \{0,1\}^*$ there exists a constant k such that $L \in \mathrm{COMM}(k)$.*

Proof. If L is regular, then there exists a deterministic finite automaton A recognizing L. Let A have s states $q_0, q_1, \ldots, q_{s-1}$, and let $k = \lceil \log_2 s \rceil$.

For each $n \in \mathbb{N}$ we construct a protocol $D_n = \langle \overline{\Pi}, \Phi \rangle$ computing $h_n(L)$. $\overline{\Pi}$ is defined by $\overline{\Pi_L} = \{x_1, \ldots, x_{\lceil n/2 \rceil}\}$. The work of Φ can be described as follows. $\Phi(\alpha_1 \ldots \alpha_{\lceil n/2 \rceil}, \lambda)$ is $\mathrm{BIN}_k^{-1}(r)$ iff the automaton A working on the input word $\alpha_1 \ldots \alpha_{\lceil n/2 \rceil}$ (from its initial state) ends in the state q_r. Then $\Phi(\alpha_{\lceil n/2 \rceil + 1} \ldots \alpha_n, \mathrm{BIN}_k^{-1}(r)\$) = \overline{1}$ ($\overline{0}$) iff the automaton A starting in the state q_r with the postfix $\alpha_{\lceil n/2 \rceil + 1} \ldots \alpha_n$ ends in an accepting (rejecting) state. Obviously, D_n computes $h_n(L)$ and $\mathrm{cc}(D_n) = k = \lceil \log_2 s \rceil$. □

Next, we show that no positive integer m exists such that all regular languages are in $\mathrm{COMM}(m)$.

Theorem 2.3.5.2 *For every positive integer m, there exists a regular language $L(m)$ such that $L(m) \notin \mathrm{COMM}(m)$.*

Proof. Let m be a positive integer, and let $L(m) = \{w \in \{0,1\}^* \mid \#_1(w) = 2^{m+1}\}$. Let, for any $n \geq 2^{m+2}$, $h_n(L(m))$ be defined over the set of variables $X = \{x_1, \ldots, x_n\}$. We shall prove $\mathrm{cc}(h_n(L(m))) > m$ for every $n \geq 2^{m+2}$.

Let Π be any balanced partition of X. Let $k \in \{\lfloor n/2 \rfloor, \lceil n/2 \rceil\}$ be the cardinality of Π_L. Let M' be a submatrix of $M(h_n(L(m)), \Pi)$ defined as the intersection of the $2^{m+1}+1$ rows labelled by the input assignments 0^k, 10^{k-1}, $\ldots, 1^{2^{m+1}}0^{k-2^{m+1}}$ and of the $2^{m+1}+1$ columns labelled by the input assignments 0^k, 10^{k-1}, 1^20^{k-2}, $\ldots$, $1^{2^{m+1}}0^{k-2^{m+1}}$. Obviously, the $(2^{m+1}+1) \times (2^{m+1}+1)$ matrix M' has exactly $(2^{m+1}+1)$ 1's, all lying on the diagonal leading from the lower left corner to the upper right corner. Thus, for any $n \geq 2^{m+2}$ and any $\Pi \in \mathrm{Bal}(X)$

$$\mathrm{rank}(M(h_n(L(m)), \Pi)) \geq \mathrm{rank}(M') \geq 2^{m+1} + 1,$$

which implies

$$\mathrm{cc}(h_n(L(m)), \Pi) \geq \lceil \log_2(2^{m+1}+1) \rceil \geq m+2$$

for any $n \geq 2^{m+2}$ and any balanced partition Π. □

Following Theorems 2.3.5.1 and 2.3.5.2, it is easy to see that the fact $\mathrm{COMM}(m-1) \subsetneq \mathrm{COMM}(m)$ for any positive integer m [a special case of the hierarchy $\mathrm{COMM}(g(n)-1) \subsetneq \mathrm{COMM}(g(n))$ claimed in Theorem 2.3.4.9] can be proved constructively [in constract to the existential (counting) proof of Theorem 2.3.4.9] by showing $L(m) \in \mathrm{COMM}(m+2) - \mathrm{COMM}(m+1)$ for any $m \in \mathbb{N}$.

Now let us consider the following context-free language

$$L_R = \{01^d0ww^R1^r \mid d \geq 0, r \in \{0,1\}, r+d \text{ is even }, w \in \{0,1\}^*\} \cup$$

$$\{1^sww^R01^k \mid k \geq 0, s \in \{1,2\}, s+k \text{ is odd}, w \in \{0,1\}^*\}.$$

We will show that L_R has linear communication complexity. Before doing so, we give a useful, technical lemma.

Lemma 2.3.5.3 *Let n and m be positive integers. Let $X = \{x_1, \ldots, x_n\}$, and let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function defined on the variables in X. Let $f(x_1, \ldots, x_n) = \bigvee_{i=1}^{m} f_i(x_1, \ldots, x_m)$ for some Boolean functions $f_1, \ldots, f_m \in B_2^n$, and let Π be a partition of X. Let $\mathcal{A}(f_i, \Pi)$ be a 1-fooling set for f_i and Π (i.e., $f_i(\alpha) = 1$ for every $\alpha \in \mathcal{A}(f_i, \Pi)$), and additionally*

(i) for every $\alpha, \beta \in \mathcal{A}(f_i, \Pi)$, and every $j \in \{1, \ldots, m\} - \{i\}$:
$f_j(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})) = f_j(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R})) = 0$.

Then $\mathcal{A}(f_i, \Pi)$ is also a 1-fooling set for f and Π.

Proof. Obviously, for every $\alpha \in \mathcal{A}(f_i, \Pi)$: $f(\alpha) = f_i(\alpha) = 1$. Let $\alpha, \beta \in \mathcal{A}(f_i, \Pi)$, and $\alpha \neq \beta$. Then $f_i(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})) = 0$ or $f_i(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R})) = 0$.

Following property (i) we get $f(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})) = 0$ or $f(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R})) = 0$. □

Theorem 2.3.5.4 *For every positive integer n,*

$$\mathrm{cc}(h_n(L_R)) \geq n/8 - 2 \ .$$

Proof. We have to prove $\mathrm{cc}(h_n(L_R)) \geq n/8 - 2$ for any $n \in \mathbb{N}$. So, let n be an arbitrary, positive, even integer (for odd n the proof can be realized in the same way), and let $X = \{x_1, \ldots, x_n\}$ be the set of variables over which $h_n(L_R)$ is defined. We can write $L_R = L_{R,0} \cup L_{R,1}$, where

$$\begin{aligned} L_{R,0} &= \{01^d0ww^R1^r \mid d \geq 0, r \in \{0,1\}, r+d \text{ is even}, w \in \{0,1\}^*\} \text{ and} \\ L_{R,1} &= \{1^sww^R01^k \mid k \geq 0, s \in \{1,2\}, s+k \text{ is odd}, w \in \{0,1\}^*\}. \end{aligned}$$

Further,

$$\begin{aligned} L_{R,0}[n] &= L_{R,0} \cap \{0,1\}^n = \bigcup_{i=0}^{n-2} L_{R,0,i}[n], \\ L_{R,1}[n] &= L_{R,1} \cap \{0,1\}^n = \bigcup_{j=0}^{n-2} L_{R,1,j}[n], \end{aligned}$$

where

$$L_{R,0,i}[n] = \{01^i0ww^R1^r \mid w \in \{0,1\}^*, r \in \{0,1\}, |w| = (n-i-r-2)/2\}$$

and

$$L_{R,1,j}[n] = \{1^sww^R01^j \mid w \in \{0,1\}^*, s \in \{1,2\}, |w| = (n-s-j-1)/2\} \ .$$

Thus,

$$\begin{aligned} h_n(L_R)(x_1, \ldots, x_n) = &(\bigvee_{i=0}^{n-2} h_n(L_{R,0,i})(x_1, \ldots, x_n)) \vee \\ &(\bigvee_{j=0}^{n-2} h_n(L_{R,1,j})(x_1, \ldots, x_n)), \end{aligned}$$

where

$$\begin{aligned} h_n(L_{R,0,i})(x_1, \ldots, x_n) = \ & x_1^{\mathtt{C}} \wedge (\bigwedge_{r=2}^{i+1} x_r) \wedge x_{i+2}^{\mathtt{C}} \wedge \\ & (\bigwedge_{s=i+3}^{(n-i-2)/2+i+2} (x_s \equiv x_{n-s+i+2})) \\ & \text{for all even } i \in \{0, \ldots, n-2\}, \end{aligned}$$

$$
\begin{aligned}
h_n(L_{R,0,i})(x_1,\ldots,x_n) &= x_1^{\mathtt{C}} \wedge (\bigwedge_{r=2}^{i+1} x_r) \wedge x_{i+2}^{\mathtt{C}} \wedge x_n \wedge \\
&\quad (\bigwedge_{s=i+3}^{(n-i-3)/2+i+2} (x_s \equiv x_{n-s+i+2})) \\
&\quad \text{for all odd } i \in \{1,\ldots,n-3\}, \\
h_n(L_{R,1,j})(x_1,\ldots,x_n) &= x_1 \wedge (\bigwedge_{r=2}^{1+(n-j-2)/2} (x_r \equiv x_{(n-j-2)-r+3}) \wedge \\
&\quad x_{n-j}^{\mathtt{C}} \wedge (\bigwedge_{m=n-j+1}^{n} x_m) \\
&\quad \text{for all even } j \in \{0,\ldots,n-2\}, \text{ and} \\
h_n(L_{R,1,j})(x_1,\ldots,x_n) &= x_1 \wedge x_2 \wedge (\bigwedge_{r=3}^{2+(n-j-3)/2} (x_r \equiv x_{(n-j-3)-r+5}) \wedge \\
&\quad x_{n-j}^{\mathtt{C}} \wedge (\bigwedge_{m=n-j+1}^{n} x_m) \\
&\quad \text{for all odd } j \in \{1,\ldots,n-3\}.
\end{aligned}
$$

Since $L_{R,0}[n] \cap L_{R,1}[n] = \emptyset$, and $L_{R,0,i}[n] \cap L_{R,0,j}[n] = L_{R,1,i}[n] \cap L_{R,1,j}[n] = \emptyset$ for any $i \neq j$, we observe that if $\mathcal{A}$ is a 1-fooling set for some Π and some $f \in \{h_n(L_{R,0,i}), h_n(L_{R,1,j}) \mid i,j = 0,\ldots n-2\}$ with the property $f(\alpha) = 1$ for every $\alpha \in \mathcal{A}$, then $\mathcal{A}$ has the property (i) of Lemma 2.3.5.3 according to the functions in $\{h_n(L_{R,k,i}) \mid k \in \{1,0\}, i \in \{0,\ldots,n-2\}\} - \{f\}$. So the assumption of Lemma 2.3.5.3 holds for $h_n(L_R)$ which means that it is sufficient to show that for each $\Pi \in \mathrm{Bal}(X)$ there exist $k \in \{1,2\}$, $i \in \{0,\ldots,n-2\}$ such that we can find a large fooling set $\mathcal{A}(h_n(L_{R,k,i}), \Pi)$ for $h_n(L_{R,k,i})$ and Π.

Let us do this. In what follows we say that $h_n(L_{R,k,i})$ compares a pair of variables (x_s, x_r) iff $x_s \equiv x_r$ is included in the above stated formula of $h_n(L_{R,k,i})$. Let

$$S(k,i) = \{(x_s, x_r) \mid (x_s, x_r) \text{ are compared by } h_n(L_{R,k,i})\}$$

be the set of all pairs compared by $h_n(L_{R,k,i})$ for any $k \in \{0,1\}$, $i \in \{0,\ldots,n-2\}$. We observe that $(k_1 \neq k_2)$ or $(i_1 \neq i_2)$ implies $S(k_1,i_1) \cap S(k_2,i_2) = \emptyset$ for every $k_1, k_2 \in \{0,1\}$, $i_1, i_2 \in \{0,\ldots,n-2\}$. Further, we see that

$$|S(d,m)| \geq \lceil (n-m-3)/2 \rceil$$

for every $d \in \{0,1\}$, and every $m \in \{0,\ldots,n-2\}$. Thus, the number of distinct pairs compared by $h_n(L_R)$ is

$$
\begin{aligned}
\sum_{d=0,1} \sum_{m=0}^{n-2} |S(d,m)| &\geq 2 \cdot \sum_{m=0}^{n-3} \lceil (n-m-3)/2 \rceil \\
\geq \sum_{m=0}^{n-3} (n-m-3) &= \sum_{z=1}^{n-3} z = \frac{n^2-5n+6}{2}.
\end{aligned}
$$

Since the number of all unordered pairs (x_s, x_r) of input variables is exactly $\binom{n}{2} = \frac{n^2-n}{2}$ we have that at most $2n-3$ pairs of variables are not compared by $h_n(L_R)$.

Let Π be an arbitrary balanced partition of X. For any pair (x_s, x_r), $r, s = 1, \ldots, n$, we say that Π separates the pair (x_s, x_r) iff $(x_r \in \Pi_L$ and $x_s \in \Pi_R)$ or $(x_r \in \Pi_R$ and $x_s \in \Pi_L)$. Each $\Pi \in \mathrm{Bal}(X)$ separates exactly $n^2/4$ unordered pairs. Let $\mathrm{Sep}(\Pi)$ denote the set of all by Π separated pairs.

Following our enumerations we see

$$|(\bigcup_{d=0,1}(\bigcup_{m=0}^{n-2} S(d,m))) \cap \mathrm{Sep}(\Pi)| \geq \frac{n^2}{4} - 2n \ .$$

So there must exist $k \in \{0,1\}$ and $i \in \{0, \ldots, n-2\}$ such that

$$|S(k,i) \cap \mathrm{Sep}(\Pi)| \geq (n^2/4 - 2n)/(2n-2) \geq \frac{n}{8} - 2 \ .$$

Thus, the function $h_n(L_{R,k,i})$ compares at least $n/8 - 2$ pairs of variables separated by Π. Using the standard technique (already presented above (e.g., in Example 2.2.2.4)), one can construct a 1-fooling set $\mathcal{A}(h_n(L_{R,k,i}), \Pi)$ of cardinality $2^{n/8-2}$. Following Lemma 2.3.5.3, $\mathcal{A}(h_n(L_{R,k,i}), \Pi)$ is also a 1-fooling set for $h_n(L_R)$ and Π which completes the proof. □

Following Theorem 2.3.5.4, we see that one of the fundamental computing problems, the recognition of context-free languages, has already the highest (linear) communication complexity. Another contribution of Theorem 2.3.5.4 and Lemma 2.3.5.3 is the proof technique showing that sometimes the proof of a lower bound on $\mathrm{cc}(f)$ for some Boolean function f can be "reduced" to prove a lower bound on $\mathrm{cc}(\{f_1, \ldots, f_r\})$, where $f_1, \ldots, f_r$ have some special properties related to f.

Finally, we show that COMM(0) contains some languages which are not recursively enumerable. Note that this result is only due to the fact that communication protocols are non-uniform computing models. Let $\mathcal{L}_{RE}$ denote the class of recursively enumerable languages.

Theorem 2.3.5.5

$$\mathrm{COMM}(0) - \mathcal{L}_{RE} \neq \emptyset$$

Proof. There are infinitely many languages $L \notin \mathcal{L}_{RE}$ with the property $|L[n]| = 1$ for every $n \in \mathbb{N}$, $L \subseteq \{0,1\}^*$. Let L be such a language. We construct $L' = \{wy \mid w \in L \text{ and } y \in \{0,1\}^{|w|}\}$. Obviously, L' is not in $\mathcal{L}_{RE}$. On the other hand, $\mathrm{cc}(h_{2n}(L')) = 0$ for any $n \in \mathbb{N}$ because if the left computer has the first half of input variables, then it can directly decide whether the input assignment is the only word in $L[n]$ or not. Obviously, $\mathrm{cc}(h_{2n+1}(L')) = 0$ for any $n \in \mathbb{N}$ because $L'[2n+1] = \emptyset$. □

2.3.6 Exercises

Exercise 2.3.6.1 *Prove* $\mathrm{cc}(h_{2m}(\tilde{L})) \leq \lceil \log_2(m+1) \rceil$ *(claimed in Theorem 2.3.3.1) by using the fool (tiling) method.*

Exercise 2.3.6.2 *Prove the lower bound presented in Theorem 2.3.3.2 by using the mrank method.*

Exercise 2.3.6.3 * *Prove* $\mathrm{acc}(h_n(L_\Delta)) = \Omega(n)$ *for any* $n = \binom{m}{2}, m \in \mathbb{N} - \{0,1\}$.

Exercise 2.3.6.4 *Let* P^m_{2m} *be the problem defined in Example 2.3.3.6. Prove* $\mathrm{acc}(P^m_{2m}) = \Omega(m)$.

Exercise 2.3.6.5 * *Find a language* $L \subseteq \{0,1\}^*$ *such that*

(i) $\mathrm{cc}(h_n(L)) = \Omega(n)$,

(ii) $\mathrm{acc}(h_n(L)) \leq 1$ *for every* $n \in \mathbb{N}$.

Exercise 2.3.6.6 *Define, for any positive integers* $n, m, m \leq \frac{n}{2}$, $\mathrm{aCOMM}_n(m) = \{f \in B^n_2 \mid \mathrm{acc}(f) \leq m\}$. *Prove a strong hierarchy of such complexity classes.*

Exercise 2.3.6.7 * *Find two languages* U *and* V *such that*

(i) $\mathrm{acc}(h_n(U)) \leq 1$, *and* $\mathrm{acc}(h_n(V)) \leq 1$ *for every* $n \in \mathbb{N}$, *and*

(ii) $\mathrm{acc}(h_n(U \cup V)) = \Omega(n)$.

Exercise 2.3.6.8 * *Construct two specific languages* L_1 *and* L_2 *such that*

(i) $\mathrm{cc}(h_n(L_1)) \leq 1$, *and* $\mathrm{cc}(h_n(L_2)) \leq 1$ *for every* $n \in \mathbb{N}$, *and*

(ii) $\mathrm{cc}(h_n(L_1 \cup L_2)) = \Omega(n^k)$ *for some positive, real number* $k \leq 1$.

Note that Theorem 2.3.4.16 does not render such results, because the language L_1 *is chosen existentially.*

Exercise 2.3.6.9 *A Boolean function* $f \in B^{2n}_2$ *is* **symmetric** *if for every* $\alpha, \beta \in \{0,1\}^{2n}$, $\#_1(\alpha) = \#_1(\beta)$ *implies* $f(\alpha) = f(\beta)$. *Prove that, for every symmetric Boolean function* $f \in B^{2n}_2$, $\mathrm{cc}(f) \leq \lceil log_2 n \rceil$, *and that there exists a symmetric Boolean function* $h \in B^{2n}_2$ *such that* $\mathrm{cc}(h) = \lceil log_2 n \rceil$.

Exercise 2.3.6.10 * *Let* c *be a fixed constant. Let* f *be a Boolean function of* n *variables,* $n \in \mathbb{N}$. *Let* Π *be a balanced partition of the input variables of* f *such that* $\mathrm{cc}(f, \Pi) = \Omega(n)$. *Give a general construction which for given* f *provides a function* f^* *such that* $\mathrm{cc}(f^*) = \Omega(n)$ *[*f^* *is hard for all partitions] and* f^* *is defined on* $c \cdot n$ *variables.*

2.3.7 Research Problems

Problem 2.3.7.1 * *Find a specific language* $L \subseteq \{0,1\}^*$ *such that*

(i) $\mathrm{cc}(h_n(L)) \geq \frac{n}{4}$ *for all sufficiently large* n, *or*

*(ii)*** $\mathrm{cc}(h_n(L)) \geq \frac{n}{2} - o(n)$.

Note that the highest lower bound we know for a specific language is $\frac{n}{8}$.

Problem 2.3.7.2 *Define, for every Boolean function* f *defined on a set* X *of input variables,*

(i) $\mathrm{Fool}(f) = \min\{|\ \mathrm{Fool}(f, \Pi)\ ||\ \Pi \in \mathrm{Bal}(X)\}$,

(ii) $\mathrm{Rank}(f) = \min\{\mathrm{Rank}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\}$, *and*

(iii) $\mathrm{Til}(f) = \min\{\mathrm{Til}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\}$.

Obviously, the numbers $\mathrm{Fool}(f)$, $\mathrm{Rank}(f)$, *and* $\mathrm{Til}(f)$ *describe the potency of the lower bound proof methods fool, mrank, and tiling.*

Find the maximal possible differences between $\mathrm{cc}(f)$, $\mathrm{Fool}(f)$, $\mathrm{Rank}(f)$, *and* $\mathrm{Til}(f)$ *as done in Section 2.2.2 for fixed partitions. At present, we do not know any paper presenting an exponential gap between any pair from* $\{\mathrm{Fool}(f), \mathrm{Rank}(f), \mathrm{Til}(f), \mathrm{cc}(f)\}^2$ *despite the fact that we gave exponential gaps for some relations between* $\mathrm{Fool}(f, \Pi), \mathrm{Rank}(f, \Pi), \mathrm{Til}(f, \Pi)$, *and* $\mathrm{cc}(f, \Pi)$ *for some fixed partition* Π.

Problem 2.3.7.3 *How hard (from the communication point of view) is the recognition of linear languages?*

Problem 2.3.7.4 *Find an interesting, intensively investigated computing problem* $P = \{f_1, \ldots, f_k\}$ *such that* $\mathrm{cc}(P)[\mathrm{acc}(P)]$ *is large, and there exists a partition of the problem* P *into subproblems* $P_1, P_2, \ldots, P_l$ $(P_1 \cup P_2 \cup \ldots \cup P_l = P)$ *with small* $\mathrm{cc}(P_i)$ *for every* $i \in \{1, \ldots, l\}$.

Problem 2.3.7.5 *Prove or disprove:*

"There exists a context-free language L *such that* $\mathrm{cc}(h_n(L_1)) + \mathrm{cc}(h_n(L_2)) = \Omega(n)$ *for any languages* L_1 *and* L_2 *such that* $L = L_1 \cup L_2$ $(L = L_1 \cap L_2)$."

Consider the above problem for a decompostion of L *into several languages* $(L = L_1 \cup L_2 \cup \ldots \cup L_k,\ k \geq 3)$.

2.4 One-Way Communication Complexity

2.4.1 Introduction

The definition of one-way communication complexity is based on the very hard restriction to (communication) protocols allowing only one communication mes-

sage from the left computer C_I to the right computer C_{II}. After this the right computer must give the output (outputs). The importance of one-way communication complexity is in the combination of the following facts:

1. One-way communication complexity directly provides lower bounds for some important complexity measures of parallel computing models (for instance, the area of VLSI circuits).

2. One-way communication complexity is in many cases essentially higher than the communication complexity.

We know several examples of computing problems for which the communication complexity is too small to provide useful lower bounds, but the one-way communication complexity is high enough to help to prove the optimality of some parallel algorithms according to some parallel complexity measures. We shall show some of these examples later.

Section 2.4 is organized as follows. Section 2.4.2 presents the formal definition of one-way communication complexity. Section 2.4.3 is devoted to the lower bound proof methods for one-way communication complexity. Section 2.4.4 studies the relation between communication complexity and one-way communication complexity, and also some theoretical properties of one-way communication complexity.

2.4.2 Definitions

We start with the definition of one-way protocols.

Definition 2.4.2.1 *Let $D_n = \langle \Pi, \Phi \rangle$ be a protocol computing a computing problem P_n^r of size n for some $n, r \in \mathbb{N} - \{0\}$. For every $k \in \mathbb{N}$, a* ***k*-round computation** *of D_n is any computation of D_n of the form $c = c_1\$c_2\$ \ldots \$c_k\$c_{k+1}d$, where $c_i \in \{0,1\}^+$ for $i = 1, \ldots, k$ (i.e., c contains exactly k messages exchanged), $c_{k+1} \in \{\bar{0}, \bar{1}\}^+$, and $d \in \{\lambda, \$\$c_{k+2}\}$ for some $c_{k+2} \in \{\bar{0}, \bar{1}\}^+$. We say also that the* **computation *c* has *k* rounds**. *D_n is called a* ***k*-round protocol** *if each computation of D_n has at most k rounds. A one-round protocol is also called a* **one-way protocol**.

We observe that every one-way protocol computing a Boolean function has either one-round computations or zero-round computations, i.e., there is no one-way protocol having a one-round computation and also a zero-round computation (see condition (*iii*) of Definition 2.2.1.6 and condition (*iv*) of Definition 2.2.1.9). Obviously, a protocol $\langle \Pi, \Phi \rangle$ computing a Boolean function f can have zero-round computations only if f does not essentially depend on any variable in Π_R.

Definition 2.4.2.2 *Let P_n^r be a computing problem with the set of input variables X and the set of output variables Y for some $n, r \in \mathbb{N} - \{0\}$. The* **one-way**

communication complexity of P_n^r according to an almost balanced partition $\Pi \in \text{Abal}(X, Y)$ *is*

$$\mathbf{cc_1}(P_n^r, \Pi) = \min_{\Phi}\{\text{cc}(D) \mid D = \langle \Pi, \Phi \rangle \textit{ is a one way protocol computing } P_n^r\} \ .$$

The **one-way communication complexity of P_n^r** *is*

$$\mathbf{cc_1}(P_n^r) = \min\{\text{cc}_1(P_n^r, \Pi) \mid \Pi \in \text{Bal}(X, Y)\},$$

and the **one-way α-communication complexity of P_n^r** *is*

$$\mathbf{acc_1}(P_n^r) = \min\{\text{cc}_1(P_n^r, \Pi) \mid \Pi \in \text{Abal}(X, Y)\} \ .$$

Despite the fact that Definition 2.4.2.2 works also for P_n^1 problems, we give a separate definition of one-way communication complexity of Boolean functions.

Definition 2.4.2.3 *Let f be a Boolean function of n variables in X for some $n \in \mathbb{N} - \{0\}$. The* **one-way communication complexity of f according to a partition $\Pi \in \text{Abal}(X)$** *is*

$$\mathbf{cc_1}(f, \Pi) = \min_{\Phi}\{\text{cc}(D) \mid D = \langle \Pi, \Phi \rangle \textit{ is a one-way protocol computing } f\} \ .$$

The **one-way communication complexity of f** *is*

$$\mathbf{cc_1}(f) = \min\{\text{cc}_1(f, \Pi) \mid \Pi \in \text{Bal}(X)\},$$

and the **one-way α-communication complexity of f** *is*

$$\mathbf{acc_1}(f) = \min\{\text{cc}_1(f, \Pi) \mid \Pi \in \text{Abal}(X)\} \ .$$

Observation 2.4.2.4 *For every computing problem P_n^r of size n, $n, r \in \mathbb{N}$, $\text{acc}_1(P_n^r) \leq \text{cc}_1(P_n^r)$, $\text{acc}(P_n^r) \leq \text{acc}_1(P_n^r)$, and $\text{cc}(P_n^r) \leq \text{cc}_1(P_n^r)$.*

We illustrate Definition 2.4.2.3 by one problem for which the communication complexity is the same as the one-way communication complexity.

Example 2.4.2.5 Let us consider the language $\tilde{L} = \{\alpha \in \{0,1\}^+ \mid \#_0(\alpha) = \#_1(\alpha)\}$ from Example 2.2.2.1. In Theorem 2.3.3.1 we have proved $\text{cc}(h_{2m}(\tilde{L})) = \lceil \log_2(m+1) \rceil$ for every $m \in \mathbb{N} - \{0\}$. The protocol used to get the upper bound was a one-way protocol which yields $\text{cc}(h_{2m}(\tilde{L})) = \text{cc}_1(h_{2m}(\tilde{L}))$ for $\forall\ m \in \mathbb{N} - \{0\}$. □

Example 2.4.2.5 shows that there exist problems for which there is no difference between one-way communication complexity and communication complexity. Considering the same language $\tilde{L}$ one can easily show that $\text{acc}(h_{2m}(\tilde{L})) = \text{acc}_1(h_{2m}(\tilde{L}))$ for every positive integer m. Later, we shall also show that there

are computing problems for which the gap between one-way communication complexity and communication complexity is exponential.

We close this section by giving the definition of Boolean function classes and language classes determined by one-way communication complexity.

Definition 2.4.2.6 *We set for all* $n, m \in \mathbb{N}$, $m \leq n/2$, $\mathbf{COMM_n^1(m)} = \{f \in B_2^n \mid cc_1(f) \leq m\}$ *as the set of all Boolean functions of* n *variables computable within one-way communication complexity* m.

Definition 2.4.2.7 *Let* $g : \mathbb{N} \to \mathbb{N}$ *be a function with the property* $g(n) \leq n/2$ *for any* $n \in \mathbb{N}$. *We set* $\mathbf{COMM^1(g(n))} = \{L \subseteq \{0,1\}^* \mid cc_1(h_n(L)) \leq g(n)$ *for any* $n \in \mathbb{N}\}$.

2.4.3 Methods for Proving Lower Bounds

We shall present two lower bound methods for one-way communication complexity that are close to the lower bound methods for communication complexity presented in Sections 2.2.2 and 2.3.3. In fact the following methods are simplifications of the methods introduced for communication complexity. We start with the lower bound proof technique based on the idea of fooling sets.

Definition 2.4.3.1 *Let* P_n^r *be a computing problem of size* n *with the set of input variables* $X = \{x_1, \ldots, x_n\}$, *and the set of output variables* $Y = \{y_1, \ldots, y_r\}$. *Let* Π *be a partition of* X *and* Y. *Then a* **one-way fooling set** $\mathcal{A}_1(P_n^r, \Pi)$ **for** P_n^r **and** Π *is any set of input assignments from* $\Pi_{L,X}$ *to* $\{0,1\}$ *such that, for any distinct* α *and* β *in* $\mathcal{A}_1(P_n^r, \Pi)$, *there exists an input assignment* γ *from* $\Pi_{R,X}$ *to* $\{0,1\}$ *such that* $P_n^r(\Pi^{-1}(\alpha,\gamma))$ *differs from* $P_n^r(\Pi^{-1}(\beta,\gamma))$ *on some variable in* $\Pi_{R,Y}$.

If $P_n^r = \{f\}$ for some Boolean function f we get the following version of the previous definition.

Definition 2.4.3.2 *Let* f *be a Boolean function defined on* n *variables from* $X = \{x_1, \ldots, x_n\}$. *Let* Π *be a partition of* X. *Then a* **one-way fooling set** $\mathcal{A}_1(f, \Pi)$ **for** f **and** Π *is any set of input assignments from* $\Pi_{L,X}$ *to* $\{0,1\}$ *such that, for any distinct* α *and* β *in* $\mathcal{A}_1(f, \Pi)$, *there exists an input assignment* γ *from* $\Pi_{R,X}$ *to* $\{0,1\}$ *such that* $f(\Pi^{-1}(\alpha,\gamma)) \neq f(\Pi^{-1}(\beta,\gamma))$.

To see an example of a one-way fooling set we shall consider the function $f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6$. Let Π be a partition of $X = \{x_1, \ldots, x_6\}$ defined by $\Pi(x_1) = \Pi(x_2) = \Pi(x_3) = 1$ and $\Pi(x_4) = \Pi(x_5) = \Pi(x_6) = 2$. Then $\mathcal{A}_1 = \{000, 001\}$, $\mathcal{A}_2 = \{010, 110\}$, and $\mathcal{A}_3 = \{111, 000\}$ are one-way fooling sets for f and Π because $1 = f(000001) \neq f(001001) = 0$ ($\gamma = 001$), $1 = f(010110) \neq f(110110) = 0$ ($\gamma = 110$), and $1 = f(111\gamma) \neq f(000\gamma) = 0$ for every $\gamma \in \{000, 110, 011, 101\}$.

Theorem 2.4.3.3 *Let P_n^r be a computing problem of size n with the set of input variables X and the set of output variables Y. Let $\Pi \in \mathrm{Abal}(X,Y)$. If $\mathcal{A}_1(P_n^r, \Pi)$ is a fooling set for P_n^r and Π, then*

$$\mathrm{cc}_1(P_n^r, \Pi) \geq \lceil \log_2 |\mathcal{A}_1(P_n^r, \Pi)| \rceil \ .$$

Proof. The idea is to show that any one-way protocol computing P_n^r with the partition Π must have different communications (the messages from the left computer to the right computer) for every two different assignments from $\mathcal{A}_1(P_n^r, \Pi)$. Assuming $|\mathcal{A}_1(P_n^r, \Pi)| \geq 2$ we get that $\Pi_{R,Y} \neq \emptyset$ which implies that every protocol $\langle \Pi, \Phi \rangle$ computing P_n^r with the partition Π has all the computation of the form $c_1\$c_2w$, where $c_1 \in \{0,1\}^+$, $c_2 \in \{\bar{0},\bar{1}\}^+$, and $w \in \{\lambda, \$\$d\}$ for a $d \in \{0,1\}^+$. Let us assume that there are two different $\alpha, \beta \in \mathcal{A}_1(P_n^r, \Pi)$ such that $\Phi(\alpha,\lambda) = \Phi(\beta,\lambda)$. (i.e., α and β have the same communication). Then $\Phi(\delta, \Phi(\alpha,\lambda)) = \Phi(\delta, \Phi(\beta,\lambda))$ for all input assignments δ from $\Pi_{R,X}$ to $\{0,1\}$. But this contradicts the assumption that $\mathcal{A}_1(P_n^r, \Pi)$ is a one-way fooling (i.e., that there exists an input assignment $\gamma : \Pi_{R,X} \to \{0,1\}$ such that $\Phi(\gamma, \Phi(\alpha,\lambda)) \neq \Phi(\gamma, \Phi(\beta,\lambda))$. □

So, following Theorem 2.4.3.3 we get the method "1fool" for proving lower bounds on one-way communication complexity.

Method 1fool

Input: A problem P_n^r of size n with a set of input variables X and a set of output variables Y.

Step 1: For each $\Pi \in \mathrm{Bal}(X,Y)$ $[\Pi \in \mathrm{Abal}(X,Y)]$, find a fooling set $\mathcal{A}_1(P_n^r, \Pi)$.

Step 2: Compute $d = \min\{|\mathcal{A}_1(P_n^r, \Pi)| \mid \Pi \in \mathrm{Bal}(X,Y)\}$
$[d = \min\{|\mathcal{A}_1(P_n^r, \Pi)| \mid \Pi \in \mathrm{Abal}(X,Y)\}]$

Output: "$\mathrm{cc}(P_n^r) \geq \lceil \log_2 d \rceil$" ["$\mathrm{acc}(P_n^r) \geq \lceil \log_2 d \rceil$"]

We illustrate the use of the method 1fool to get a lower bound for the recognition of the language $\bar{L} = \{u = w_1w_2 \ldots w_m \in \{0,1\}^* \mid m = 2^r$ for some $r \in \mathbb{N}$, and $|w_i| = r$ for all $i \in \{1,\ldots,m\}$, and $w_j = 1^r$ for $j = \mathrm{BIN}(w_{\mathrm{BIN}(w_1)})\}$. Note that the recognition of the language $\bar{L}$ can be viewed as the test for the content of a register by indirect addressing.

Theorem 2.4.3.4 *For any $n \in \mathbb{N}$, $n = r \cdot 2^r$ for some $r \geq 4$, $r \in \mathbb{N}$,*

$$\mathrm{cc}_1(h_n(\bar{L})) \geq (n/\log_2 n)^{1/2} \ .$$

Proof. Let $X = \bigcup_{i=1}^{2^r} W_i$, $W_i = \{x_{(i-1)r+1}, x_{(i-1)r+2}, \ldots, x_{ir}\}$ for every $i \in \{1,\ldots,2^r\}$, be the set of input variables of $h_n(\bar{L})$ for each $n = r \cdot 2^r$, $r \geq 2$. We

have to prove $cc_1(h_n(\bar{L}), \Pi) \geq \sqrt{n/\log_2 n}$ for every $\Pi \in \text{Bal}(X)$. We distinguish two possibilities according to the partition Π. The first one assumes that, for all $i \in \{2, \ldots, m\}$, $|\Pi_L \cap W_i| \geq 1$. The second possibility assumes that there is a $j \in \{2, \ldots, m\}$ such that $|\Pi_{R,X} \cap W_j| = r$ ($W_j \subseteq \Pi_{R,X}$). We deal with these two cases separately.

1. Let, for all $i \in \{2, \ldots, m\}$, $\Pi_L \cap W_i \neq \emptyset$. Clearly, there must exist $j \in \{2, \ldots, m\}$ such that $|\Pi_R \cap W_j| \geq r/2$. The informal idea is based on setting w_1 in a way such that $\text{BIN}(w_1) = j$. Since $|\Pi_R \cap W_j| \geq r/2$ the left computer does not know $\text{BIN}(w_j)$. Moreover, the left computer "must" consider at least $2^{r/2}$ different, possible values of $\text{BIN}(w_j)$ depending on α_R. The input is accepted iff $w_{\text{BIN}(w_j)} = 1^r$, and we assume $\Pi_L \cap W_i \neq \emptyset$ for all $i \in \{1, \ldots, m\}$. This means that, for each of (at least $2^{r/2}$) possible $k = \text{BIN}(w_j)$, the left computer "has to submit" a message containing information whether all the variables in $W_k \cap \Pi_L$ have assigned the actual values 1 or not. Only with this information can the right computer immediately decide whether $w_{\text{BIN}(w_j)} = 1^r$ for each possible actual value of $\text{BIN}(w_j)$.

   We now formalize this idea. Let $\beta \in \{0,1\}$ be a value such that if we set $x = \beta$ for all $x \in \Pi_L \cap W_j$ then no input assignment from $\Pi_R \cap W_j$ to $\{0,1\}$ causes $\text{BIN}(w_j) = j$ (obviously, such β must exist). We claim that $\mathcal{A}_1(\beta, j)$ is a one-way fooling set for $h_n(\bar{L})$ and Π, where

   $\mathcal{A}_1(\beta, j) = \{\alpha_L : \Pi_L \to \{0,1\} \mid \exists\, \alpha_R : \Pi_R \to \{0,1\}$ such that:

   - $\Pi^{-1}(\alpha_L, \alpha_R) = w_1 w_2 \ldots w_m$, $m = 2^r$, $|w_i| = r$ for $r = 1, \ldots, m$;
   - $h_n(\bar{L})(\Pi^{-1}(\alpha_L, \alpha_R)) = 1$;
   - $\text{BIN}(w_1) = j$
   - $\alpha_L(z) = \beta$ for all $z \in W_j \cap \Pi_L$;
   - for all $i \in U = \{\text{BIN}(w_j) \mid w_j : W_j \to \{0,1\}, w_j(z) = \alpha_L(z) = \beta$ for all $z \in W_j \cap \Pi_L\}$, if δ is an assignment of variables in $\Pi_L \cap W_i$ then $\delta \in \{0\}^* \cup \{1\}^*$;
   - for every $i \in \{2, \ldots, m\} - (U \cup \{j\})$ the assignments of variables in $\Pi_L \cap W_i$ belongs to $\{0\}^*\}$.

   To see that $\mathcal{A}_1(\beta, j)$ is a one-way fooling set for $h_n(\bar{L})$, let α_1 and $\alpha_2 \in \mathcal{A}_1(\beta, j)$, $\alpha_1 \neq \alpha_2$. $\alpha_1 \neq \alpha_2$ implies that there is a $k \in U$ such that α_1 and α_2 differ on the assignment of variables in $\Pi_L \cap W_k$. All assignments $\gamma : \Pi_R \to \{0,1\}$ causing $\text{BIN}(w_1) = j$, $\text{BIN}(w_j) = k$, and $\gamma(x) = 1$ for all $x \in \Pi_R \cap W_k$ have the property $h_n(\bar{L})(\Pi^{-1}(\alpha_1, \gamma)) \neq h_n(\bar{L})(\Pi^{-1}(\alpha_2, \gamma))$.

   $|U| \geq 2^{r/2}$ implies $|\mathcal{A}_1(\beta, j)| \geq 2^{2^{r/2}}$, which means that the length of the message submitted is at least $2^{r/2}$.

2. Let us assume that there is a $j \in \{2, \ldots, m\}$ such that $|\Pi_R \cap W_j| = r$ ($W_j \subseteq \Pi_R$). The informal idea of the proof is based on the fact that

we set $\mathrm{BIN}(w_1) = j$. Since the left computer does not know any bit of w_j it "must" take all 2^r possibilities into account. This means that the only message submitted from the left computer to the right computer has to contain, for any $i \in \{2, 3, \ldots, 2^r\} - \{j\}$ such that $W_i \cap \Pi_L \neq \emptyset$, the information whether all values of the variables in $W_i \cap \Pi_L$ have assigned the value 1 or not. Since there are at least 2^{r-1} numbers such that $W_i \cap \Pi_L \neq \emptyset$, the length of the message must be at least 2^{r-1}.

We formalize this idea. Let $\bar{U} = \{i \in \{2, 3, \ldots, 2^r\} - \{j\} \mid W_i \cap \Pi_L \neq \emptyset\}$. We claim that $\mathcal{A}_1(j)$ is a one-way fooling set for $h_n(\bar{L})$ and Π, where

$\mathcal{A}_1(j) = \{\alpha_L : \Pi_L \to \{0,1\} \mid \exists\, \alpha_R : \Pi_R \to \{0,1\}$ such that:

- $\Pi^{-1}(\alpha_L, \alpha_R) = w_1 w_2 \ldots w_m$, $m = 2^r$, $|w_i| = r$ for $r = 1, \ldots, m$;
- $h_n(\bar{L})(\Pi^{-1}(\alpha_L, \alpha_R)) = 1$;
- $\mathrm{BIN}(w_1) = j$;
- each of the input assignments $\delta_i : \Pi_L \cap W_i \to \{0,1\}$ belongs to either $\{1\}^+$ or $\{0\}^*$ for each $i \in \bar{U}$.

Let $\alpha, \beta \in \mathcal{A}_1(j)$, and $\alpha \neq \beta$. $\alpha \neq \beta$ implies that there is a $k \in \bar{U}$ such that α and β differ on the assignment of variables in $\Pi_L \cap W_k$. All assignments $\gamma : \Pi_R \to \{0,1\}$ causing $\mathrm{BIN}(w_1) = j$, $\mathrm{BIN}(w_j) = k$, and $\gamma(x) = 1$ for all $x \in \Pi_R \cap W_k$ have the property $h_n(\bar{L})(\Pi^{-1}(\alpha, \gamma)) \neq h_n(\bar{L})(\Pi^{-1}(\beta, \gamma))$. Since $|\bar{U}| \geq 2^r/2$ implies $|\mathcal{A}_1(j)| \geq 2^{2^{r-1}}$, the length of the message is at least 2^{r-1}.

Obviously, the minimum of $2^{r/2}$ (case 1) and of 2^{r-1} (case 2) is $2^{r/2}$, and we conclude that, for every $\Pi \in \mathrm{Bal}(X)$, $\mathrm{cc}_1(h_n(\bar{L}, \Pi)) \geq 2^{r/2} \geq (n/\log_2 n)^{1/2}$. □

Note that Theorem 2.4.3.4 cannot be substantially improved. There is a one-way protocol D for $h_n(\bar{L})$ working within $2n^{1/2} + 2\log_2 n$ communication bits. The idea of the construction of D is already described in part 1 of the lower bound proof of Theorem 2.4.3.4. We choose a $\Pi \in \mathrm{Bal}(X)$ such that $W_1 \subseteq \Pi_L$ and the remaining W_i are split approximately equally ($r/2 - 1 \leq |W_i \cap \Pi_L| \leq r/2$) between the two computers. The only message flowing from the first computer to the second one is the concatenation of three words $w_1 u v$, where

– w_1 is the actual input assignment from W_1 to $\{0,1\}$ providing the information that $j = \mathrm{BIN}(w_1)$, $|w_1| = r$;

– u is the actual input assignment from $W_j \cap \Pi_L$ to $\{0,1\}$, $|u| \leq r/2$;

– v is of length $|U| \leq 2^{r/2+1}$, and if $U = \{i_1, \ldots i_m\}$, then the ℓ-th bit of v is 1 iff the input assignment from $W_{i_\ell} \cap \Pi_L$ to $\{0,1\}$ is in $\{1\}^+$.

Since $r + r/2 + 2^{r/2+1} \leq 2\log_2 n + 2n^{1/2}$ the upper bound is established.

Next, we introduce the lower bound proof method for one-way communication complexity based on the matrix representation $M(f, \Pi)$ of a Boolean function f according to a partition Π. In what follows we shall show that the method for one-way communication complexity is much simpler than the method for communication complexity based on the evaluation of the rank of $M(f, \Pi)$. For any matrix M, let $|\mathrm{Row}(M)|$ denote the number of different rows in M (i.e., the cardinality of $\{\mathrm{row}_1(M), \mathrm{row}_2(M), \ldots, \mathrm{row}_m(M)\}$ if the size of M is $m \times k$ for some $m, k \in \mathbb{N} - \{0\}$).

Theorem 2.4.3.5 *Let $f \in B_n^2$ for some $n \in \mathbb{N}$, and let X be the set of input variables for f. Let Π be a partition from* $\mathrm{Abal}(\Pi)$. *Then*

$$\mathrm{cc}_1(f, \Pi) \geq \lceil \log_2 |\mathrm{Row}(M(f, \Pi))| \rceil .$$

Proof. Let Π be a partition in $\mathrm{Abal}(\Pi)$, and let $m = |\Pi_L|$, $k = |\Pi_R|$. Let $D = \langle \Pi, \Phi \rangle$ be a one-way protocol computing f. We shall show by contradiction that D must have different communications for any two different rows of $M(f, \Pi)$. Let $\mathrm{row}_i(M(f, \Pi)) \neq \mathrm{row}_j(M(f, \Pi))$ for some $i \neq j$, $i, j \in \{1, \ldots, 2^m\}$, and let $\Phi(\mathrm{BIN}_m^{-1}(i-1), \lambda) = \Phi(\mathrm{BIN}_m^{-1}(j-1), \lambda)$. Then $\Phi(\gamma, \Phi(\mathrm{BIN}_m^{-1}(i-1)\$)) = \Phi(\gamma, \Phi(\mathrm{BIN}_m^{-1}(j-1)\$)) \in \{\bar{0}, \bar{1}\}$ for every input asssignment $\gamma : \Pi_R \to \{0,1\}$. But this implies (assuming that D computes f) $f(\Pi^{-1}(\mathrm{BIN}_m^{-1}(i-1), \gamma)) = f(\Pi^{-1}(\mathrm{BIN}_m^{-1}(j-1), \gamma))$ for every $\gamma : \Pi_R \to \{0,1\}$ which contradicts the fact that $\mathrm{row}_i(M(f, \Pi)) \neq \mathrm{row}_j(M(f, \Pi))$.

So, the number of different communication messages must be at least as large as the number of different rows in $M(f, \Pi)$, which completes the proof. □

Following Theorem 2.4.3.5 we get the following lower bound method.

Method mrow

Input: A Boolean function f with a set of input variables X.

Step 1: For each $\Pi \in \mathrm{Bal}(X)$ [$\Pi \in \mathrm{Abal}(X)$], construct the matrix $M(f, \Pi)$.

Step 2: Compute $d = \min\{|\mathrm{Row}(M(f, \Pi))| \mid \Pi \in \mathrm{Bal}(X)$ [$\Pi \in \mathrm{Abal}(X)$]$\}$.

Output: "$\mathrm{cc}_1(f) \geq \lceil \log_2 d \rceil$" ["$\mathrm{acc}_1(f) \geq \lceil \log_2 d \rceil$"].

We call attention to the fact that the methods fool (based on fooling sets) and the method mrank (based on the matrix representation of f) may yield distinct lower bounds (as shown in Section 2.2) for communication complexity. Here, both methods 1fool and mrow always yield the same lower bound for every Boolean function f.

This fact is proved in the following lemmas.

Lemma 2.4.3.6 *Let $f \in B_n^2$ for some $n \in \mathbb{N}$, and let $\Pi \in \mathrm{Abal}(X)$, where X is the set of input variables of f. Let $m = |\Pi_L|$. Let $\{\mathrm{row}_{i_1}(M(f,\Pi)), \mathrm{row}_{i_2}(M(f,\Pi)), \ldots, \mathrm{row}_{i_k}(M(f,\Pi)) \mid \mathrm{row}_{i_r}(M(f,\Pi)) \neq \mathrm{row}_{i_s}(M(f,\Pi))$ for any $r \neq s$, $r,s \in \{1,\ldots,k\}\}$ be an arbitrary set of pairwise different rows of $M(f,\Pi)$. Then $\mathcal{A}_1 = \{\mathrm{BIN}_m^{-1}(i_1-1), \mathrm{BIN}_m^{-1}(i_2-1), \ldots, \mathrm{BIN}_m^{-1}(i_k-1)\}$ is a one-way fooling set for f and Π.*

Proof. To prove Lemma 2.4.3.6 it is sufficient to show that, for any $u,v \in \{i_1,i_2,\ldots i_k\}$, $u \neq v$ implies the existence of a $\gamma : \Pi_R \to \{0,1\}$ such that $f(\Pi^{-1}(\mathrm{BIN}_m^{-1}(u-1),\gamma)) \neq f(\Pi^{-1}(\mathrm{BIN}_m^{-1}(v-1),\gamma))$. But this is obvious, because the existence of γ follows directly from the fact that $\mathrm{row}_u(M(f,\Pi)) \neq \mathrm{row}_v(M(f,\Pi))$. □

Lemma 2.4.3.7 *Let $f \in B_n^2$ for some $n \in \mathbb{N}$, and let X be the set of input variables for f. Let $\Pi \in \mathrm{Abal}(X)$, and let $m = |\Pi_L|$. If $\mathcal{A}_1 = \{\alpha_1,\alpha_2,\ldots,\alpha_k \mid \alpha_i \neq \alpha_j$ for $i \neq j\}$ is a one-way fooling set for f and Π, then $|\{\mathrm{row}_{i_1}(M(f,\Pi)), \mathrm{row}_{i_2}(M(f,\Pi)), \ldots, \mathrm{row}_{i_k}(M(f,\Pi)) \mid i_j = \mathrm{BIN}(\alpha_j) + 1$ for $j = 1,\ldots,k\}| = k$.*

Proof. Since, for any two $u,v \in \{1,\ldots,k\}$, $u \neq v$ implies the existence of an input assignment $\gamma : \Pi_R \to \{0,1\}$ such that $f(\Pi^{-1}(\alpha_u,\gamma)) \neq f(\Pi^{-1}(\alpha_v,\gamma))$ we get $\mathrm{row}_u(M(f,\Pi)) \neq \mathrm{row}_v(M(f,\Pi))$ for any $u,v \in \{1,\ldots,k\}$, $u \neq v$. □

So, the abilities of the methods 1fool and mrow are the same. The choice of the method used to prove a lower bound on $\mathrm{cc}_1(f)$ for some Boolean function f is only a question of convenience (which of the two methods 1fool and mrow yields a shorter (more readable, easier) proof of the lower bound).

The only remaining question is how much close are the lower bounds provided by the methods 1fool and mrow to the one-way communication complexity. The answer is very pleasant in this case. These method are able to provide the exact estimation on the communication complexity of any computing problem. This claim is an obvious consequence of the following theorem.

Theorem 2.4.3.8 *Let f be a Boolean function defined over a set X of input variables. Let $\Pi \in \mathrm{Abal}(X)$. Then*

$$\mathrm{cc}_1(f,\Pi) = \lceil \log_2 |\mathrm{Row}(M(f,\Pi))| \rceil .$$

Proof. The fact $\mathrm{cc}_1(f,\Pi) \geq \lceil \log_2 |\mathrm{Row}(M(f,\Pi))| \rceil$ has been proved in Theorem 2.4.3.5. To see the opposite inequality it is sufficient to realize that, for all f and Π, one can construct a one-way protocol with C_I sending exactly $|\mathrm{Row}(M(f,\Pi))|$ different messages (one message for each group of equal rows). □

2.4.4 Communication Complexity Versus One-way Communication Complexity

In this subsection we shall compare the computational power of protocols and one-way protocols. We shall show an exponential gap between these two models for the recognition of the language $\bar{L}$ defined in the previous Section 2.4.3. On the other hand we prove, for every $k \in \mathbb{N}$, that there exists a Boolean function computable within one-way communication complexity $k+1$ but not within communication complexity k. Finally we show a very strong unclosure property of one-way communication complexity classes according to disjunction and conjunction.

First, we show that the difference between communication complexity and one-way communication complexity cannot be unboundedly large.

Theorem 2.4.4.1 *For all positive integers n, m, $m \leq n/2$*

$$\mathrm{COMM}_n(m) \subseteq \mathrm{COMM}_n^1(2^{m+1}) \ .$$

Proof. To prove Theorem 2.4.4.1 it is sufficient to show that, for each protocol D working within communication complexity m, there exists an equivalent (computing the same Boolean function) one-way protocol working within communication complexity 2^{m+1}. Let $D = \langle \Pi, \Phi \rangle$ work in communication complexity m, i.e., there are at most 2^{m+1} different computations of D. Since each computation can be coded as a word over the alphabet $\{0,1\}$ we can linearly order all possible 2^{m+1} communications of D. The equivalent, one-way protocol $D_1 = \langle \Pi, \Phi_1 \rangle$ uses the same partition, and, for each $\alpha_L : \Pi_L \to \{0,1\}$, $\Phi_1(\alpha_L, \lambda) = c_1 c_2 \ldots c_{2^{m+1}} \in \{0,1\}^{2^{m+1}}$, where $c_i = 1$ iff the i-th word from $\{0,1\}^{m+1}$ is a valid (possible) computation from the first computer's point of view (corresponding to the actions of the first computer on the input α_L for all possible messages submitted by the second computer). Then, the second computer can unambiguously choose from the computations labelled by 1 (by the first computer) the unique valid computation from its point of view (i.e., corresponding to the actions of the second computer on a given input assignment $\alpha_R : \Pi_R \to \{0,1\}$). Knowing the whole computation the second computer knows obviously the output. □

Corollary 2.4.4.2 *For any function $g : \mathbb{N} \to \mathbb{N}$, $g(n) \leq n/2$,*

$$\mathrm{COMM}(g(n)) \subseteq \mathrm{COMM}^1(2^{g(n)+1}) \ .$$

Now, we show that the simulation presented in Theorem 2.4.4.1 is optimal in the sence that there is a language having exponentially greater one-way communication complexity than communication complexity.

Theorem 2.4.4.3 *For any integer* $n = r \cdot 2^r$, $r \in \mathbb{N}$, $r \geq 4$

$$\mathrm{cc}_1(h_n(\bar{L})) \geq (n/\log_2 n)^{1/2} \text{ and } \mathrm{cc}(h_n(\bar{L})) \leq 2\log_2 n + 1.$$

Proof. The fact that $\mathrm{cc}_1(h_n(\bar{L})) \geq (n/\log_2 n)^{1/2}$ is proved in Theorem 2.4.4.3. It remains to prove the upper bound on the communication complexity of $\bar{L}$ recognition. Let $X = \bigcup_{i=1}^{2^r} W_i$ be the set of input variables of $h_n(\bar{L})$ as described in the proof of Theorem 2.4.3.4. We construct a (two-round) protocol $D = \langle \Pi, \Phi \rangle$, where Π is a partition from $\mathrm{Bal}(X)$ fulfilling

(i) $W_1 \subseteq \Pi_L$

(ii) for every $i \in \{2, 3, \ldots, 2^r\}$, $|W_i \cap \Pi_L| \leq r/2$.

Then, for every input assignment $\alpha : X \to \{0,1\}$, D works as follows. $\Phi(\alpha_L, \lambda) = w_1 v$, where w_1 is the input assignment from $\Pi_L \cap W_1 = W_1$ to $\{0,1\}$, and v is the input assignment from $\Pi_L \cap W_{\mathrm{BIN}(w_1)}$ to $\{0,1\}$. Knowing $j = \mathrm{BIN}(w_1)$ after receiving the message $\Phi(\alpha_L, \lambda)$ the second computer sends the message ud to the first computer, where u is the input assignment from $\Pi_R \cap W_j$ to $\{0,1\}$ and $d \in \{0,1\}$ is equal to 1 iff the input assignment $w_{\mathrm{BIN}(w_j)}$ from $\Pi_R \cap W_{\mathrm{BIN}(w_j)}$ to $\{0,1\}$ is in $\{1\}^*$. After receiving ud the first computer computes the result $\bar{1}$ if $d = 1$ and the input assignment from $\Pi_L \cap W_{\mathrm{BIN}(j)}$ to $\{0,1\}$ is in $\{1\}^+$, and $\bar{0}$ if $d = 0$ or the input assignment from $\Pi_L \cap W_{\mathrm{BIN}(j)}$ to $\{0,1\}$ is not in $\{1\}^+$. □

So, we see that there is an exponential gap between communication complexity and one-way communication complexity. On the other hand we note there are computing problems whose one-way communication complexity is equal to their communication complexity. To see examples, consider the language $\tilde{L}$ from Example 2.4.2.5 and also the regular languages $L(m)$ for any positive integer m from Theorem 2.3.5.2. All the above mentioned languages have the property that they can be recognized with some one-way communication complexity $g(n)$ but not with communication complexity $g(n) - 1$. Thus, one additional communication bit to one-way protocols can bring more computational power than allowing the (one-way) protocols to use an arbitrary large number of rounds. We formulate this result more strongly:

Theorem 2.4.4.4 *For any* $m \leq n$:

$$\lim_{n\to\infty} \frac{|B^2_{2n}(n+m) \cap \mathrm{COMM}^1_{2n}(m)|}{|B^2_{2n}(n+m) \cap \mathrm{COMM}_{2n}(m-1)|} = \infty \ .$$

Proof. From Lemma 2.3.4.5 we have that $B^2_{2n}(n+m) \subseteq \mathrm{COMM}^1_{2n}(m)$ because the protocol described in the proof of Lemma 2.3.4.5 is a one-way protocol. So $|B^2_{2n}(n+m) \cap \mathrm{COMM}^1_{2n}(m)| = |B^2_{2n}(n+m)| = 2^{2^{n+m}}$. On the other hand Lemma 2.3.4.7 yields the fact $|B^2_{2n}(n+m) \cap \mathrm{COMM}_{2n}(m-1)| = o(2^{2^{n+m}})$. □

Now, we investigate the closure properties of one-way communication complexity. Obviously $\mathrm{COMM}_n^1(m)$ is closed under complementation for any positive integers, n, m, $m \leq n$. Using the same Boolean functions (languages) as in Section 2.3.4 it is possible to prove that for any $n \in \mathbb{N} - \{0\}$ there are two Boolean functions $f_1, f_2, \in B_n^2 \cap \mathrm{COMM}_n^1(1)$ such that $f_1 \vee f_2 \notin \mathrm{COMM}_n^1(n/6 - 7)$ [Note that Theorem 2.3.4.13 proves that $f_1 \vee f_2 \notin \mathrm{COMM}_n(n/6 - 7) \supseteq \mathrm{COMM}_n^1(n/6 - 7)$, and the protocols used to compute f_1 and f_2 are one-way protocols.] Because the proof of this strong unclosure property in Section 2.3 is existential (the function f_1 is not constructed), we prefer to present a constructive proof here.

We consider the following two languages:

$$R_0 = \{1w_1w_2 \ldots w_m \mid m \in \mathbb{N}, w_i \in \{0\}^m \cup \{1\}^m \text{ for } i = 1, \ldots, m\},$$

and

$$\begin{aligned} C_0 \;=\; & \{0x_1x_2 \ldots x_m \mid m \in \mathbb{N}, x_i = x_{i1} \ldots x_{im} \in \{0,1\}^m \text{ for } i = 1, \ldots, m, \\ & \text{and } x_{1j} = x_{2j} = \ldots = x_{mj} \text{ for every } j \in \{1, \ldots, m\}\}. \end{aligned}$$

Theorem 2.4.4.5 *For any positive integer $n = m^2 + 1$, $m \in \mathbb{N} - \{0\}$:*

(i) $\mathrm{cc}_1(h_n(R_0)) \leq 2$ *and* $\mathrm{cc}_1(h_n(C_0)) \leq 2$, *and*

(ii) $\mathrm{cc}_1(h_n(R_0) \vee h_n(C_0)) \geq m/2$.

Proof. The facts $\mathrm{cc}_1(h_n(R_0)) \leq 2$ and $\mathrm{cc}_1(h_n(C_0)) \leq 2$ are obvious. To construct one-way protocols providing $\mathrm{cc}_1(h_n(R_0)) \leq 2$ and $\mathrm{cc}_1(h_n(C_0)) \leq 2$ is a simple exercise left to the reader.

The idea of the proof of the fact *(ii)* is again based on the fact that $\mathrm{cc}_1(h_n(R_0), \Pi)$ is small only for partitions Π for which $\mathrm{cc}_1(h_n(C_0), \Pi)$ is large, and vice versa. Let the set of input variables of $h_n(R_0) \vee h_n(C_0)$ be $X = \{x_0\} \cup \bigcup_{i=1}^{m} W_i = \{x_0\} \cup \bigcup_{i=1}^{m} X_j$, where $W_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ for every $i = 1, \ldots, m$, and $X_j = \{x_{1j}, x_{2j}, \ldots, x_{mj}\}$ for every $j = 1, \ldots, m$. Let us call W_i's the rows of X, and X_j's the columns of X. The informal idea of the proof uses the fact that each partition $\Pi \in \mathrm{Bal}(X)$ divides either at least $m/2$ rows into two nonempty parts or at least $m/2$ columns into two nonempty parts. If this is clear, then it remains to prove that:

if $x_0 = 1$, then each divided row requires one bit in the message submitted,

and

if $x_0 = 0$, then each divided column requires one bit in the message submitted.

Now, let us formalize this idea. To prove *(ii)* it is sufficient to show that $\mathrm{cc}_1(h_n(R_0) \vee h_n(C_0), \Pi) \geq m/2$ for every $\Pi \in \mathrm{Bal}(X)$. We distinguish the following three possibilities according to Π:

(1) $\exists r, s \in \{1, \ldots, m\}$ such that $W_r \subseteq \Pi_L$ and $W_s \subseteq \Pi_R$;

(2) $\forall i \in \{1, \ldots, m\} : W_i \cap \Pi_L \subsetneq W_i$;

(3) $\forall j \in \{1, \ldots, m\} : W_j \cap \Pi_R \subsetneq W_j$.

We deal with these possibilities separately.

1. (1) implies that, for every $k \in \{1, \ldots, m\}$, $X_k \cap \Pi_L \neq \emptyset$ and $X_k \cap \Pi_R \neq \emptyset$. Without loss of generality we assume that $x_0 \in \Pi_L$. We observe that

$$\mathcal{A}_1 = \{0 z_1 z_2 \ldots z_m \mid z_i : X_i \cap \Pi_L \to \{0,1\} \text{ and } z_i \in \{0\}^+ \cup \{1\}^+\}$$

is a one-way fooling set for $h_n(R_0) \vee h_n(C_0)$ [also for $h_n(R_0)$] and Π. Since $|\mathcal{A}_1| = 2^m$ we get $\mathrm{cc}_1(h_n(R_0) \vee h_n(C_0), \Pi) \geq m$.

2. If Π_L contains a nonempty part of each row W_i and Π is balanced, then there exists a set of positive integers $S_\Pi = \{i_1, i_2, \ldots, i_d\} \subseteq \{1, \ldots, m\}$ such that $d > m/2$ and, for every $\ell \in S_\Pi$, $W_\ell \cap \Pi_R \neq \emptyset$. Without loss of generality we assume that $x_0 \in \Pi_L$. Following this we can simply observe that

$$\begin{aligned} \mathcal{A} \;=\; & \{0 w'_1 w'_2 \ldots w'_m \mid w'_i : W_i \cap \Pi_L \to \{0,1\} \text{ for } i = 1, \ldots, m; \\ & \text{for every } j \in S_\Pi,\ w'_j \in \{0\}^+ \cup \{1\}^+; \text{ and} \\ & \text{for every } r \in \{1, \ldots, m\} - S_\Pi,\ w'_r \in \{1\}^+\} \end{aligned}$$

is a one-way fooling set for $h_n(R_0) \vee h_n(C_0)$ and Π. Obviously $|\mathcal{A}| = 2^{|S_\Pi|} > 2^{m/2}$. So, $\mathrm{cc}_1(h_n(R_0) \vee h_n(C_0)) \geq m/2$.

3. The case (3) can be handled in the same way as case (2) because we have divided at least $m/2$ rows of X by Π. So, we get again $\mathrm{cc}_1(h_n(R_0) \vee h_n(C_0)) \geq m/2$ for every Π having the property (3).

□

2.4.5 Exercises

Exercise 2.4.5.1 *Find some languages L such that*

(i) $\mathrm{cc}(h_n(L)) = \mathrm{cc}_1(h_n(L))$ *for every* $n \in \mathbb{N}$, *and*

(ii) $\mathrm{acc}(h_n(L)) = \mathrm{acc}_1(h_n(L))$ *for every* $n \in \mathbb{N}$.

Give some sufficient conditions (having nothing common with communication complexity in their formulation) for a language L to have the property (i).

Exercise 2.4.5.2 *Give, for every Boolean function f and every partition Π, the formal description of a one-way protocol using exactly* $|\mathrm{Row}(M(f, \Pi))|$ *distinct messages.*

Exercise 2.4.5.3 * *Prove that there exists a language L with* $\mathrm{cc}(h_n(L)) = O(\log_2 n)$ *and* $\mathrm{cc}_1(h_n(L)) = \Omega(n)$.

Exercise 2.4.5.4 *Establish the exact one-way communication complexity of the languages R_0 and C_0.*

Exercise 2.4.5.5 * *Improve the result of Theorem 2.4.4.5 by constructing some specific languages L_1 and L_2 such that* $\mathrm{cc}_1(h_n(L_1)) \in O(1)$, $\mathrm{cc}_1(h_n(L_2)) \in O(1)$ *and* $\mathrm{cc}_1(h_n(L_1 \cup L_2)) \in \Omega(n^{\frac{99}{100}})$.

Exercise 2.4.5.6 *Find some specific languages L_1,L_2 such that* $\mathrm{cc}_1(h_n(L_1))$ *and* $\mathrm{cc}_1(h_n(L_2))$ *are small, but* $\mathrm{cc}_1(h_n(L_1 \cap L_2))$ *is large.*

Exercise 2.4.5.7 *Prove that* $\mathrm{acc}_1(h_n(R_0 \cup C_0)) = \Omega(\sqrt{n})$.

Exercise 2.4.5.8 *Find some languages L_1 and L_2 distinct from R_0 and C_0 such that* $\mathrm{acc}_1(h_n(L_1)) + \mathrm{acc}_1(h_n(L_2))$ *is small and* $\mathrm{acc}_1(h_n(L_1 \cup L_2))$ *is large.*

Exercise 2.4.5.9 *Prove that* $\mathrm{cc}_1(\{h_n(C_0), h_n(R_0)\}) = \Omega(\sqrt{n})$. *Prove also a similar result for one-way a-communication complexity.*

Exercise 2.4.5.10 *Find an interesting, intensively investigated computing problem $P = \{f_1, \ldots, f_k\}$ such that* $\mathrm{cc}_1(P)(\mathrm{acc}_1(P))$ *is large, and there exists a partition of the problem P into subproblems $P_1, P_2, \ldots, P_l (P_1 \cup P_2 \cup \ldots \cup P_l = \{f_1, \ldots, f_k\})$ with small* $\mathrm{cc}_1(P_i)$ *for every $i \in \{1, \ldots, l\}$. Obviously, the formulated task corresponds to the search for the decompsition of a hard problem into a small number of easy problems. We know (from the unclosure properties proved) that such problems exist, but the interest is in searching for a nice pattern from the class of the fundamental computing problems.*

2.4.6 Research Problems

Problem 2.4.6.1 * *Find a specific language L such that*

$$\mathrm{cc}(h_n(L)) = O(\log_2 n) \text{ and } \mathrm{cc}(h_n(L)) = \Omega(\log_2 n) \text{ and } \mathrm{cc}_1(h_n(L)) = \Omega(n).$$

Note that the existence of such a language has been proved (Exercise 2.4.5.3), but nobody knows a concrete language with this property. Any improvement of the gap $\mathrm{cc}_1(h_n(\overline{L})) = \Omega((\frac{n}{\log n})^{\frac{1}{2}})$ *and* $\mathrm{cc}(h_n(\overline{L})) = O(\log_2 n)$ *is of interest.*

Problem 2.4.6.2 * *Find some specific languages L_1 and L_2 such that*

$$\mathrm{cc}_1(h_n(L_1)) = O(1), \mathrm{cc}_1(h_n(L_2)) = O(1), \text{ and } \mathrm{cc}_1(h_n(L_1 \cup L_2)) = \Omega(n).$$

Note that the existence of such languages has been proved in Theorem 2.3.4.13.

Problem 2.4.6.3 * *Find a specific language L such that*

(i) $cc_1(h_n(L)) \geq \frac{n}{4}$ *for all sufficiently large* $n \in \mathbb{N}$, *or*

(ii) ** $cc_1(h_n(L)) \geq \frac{n}{2} - o(n)$.

2.5 Nondeterministic Communication Complexity and Randomized Protocols

2.5.1 Introduction

The communication protocols investigated in the previous sections were deterministic devices. Despite the fact that nondeterminism is not a natural property of any real computing model, the experience with the study of nondeterministic computing models and complexity measures shows us that by investigating nondeterminism we can learn new, essential knowledge about realistic, deterministic computations. On the other hand nondeterminism provides a base for randomization (probabilistic computations) which recently has been frequently used in practice in order to speed up large deterministic computations. These observations are also the main reasons for introducing nondeterministic (communication) protocols and for investigating them.

This section is organized as follows. Section 2.5.2 contains the definition of nondeterministic protocols, and it shows that there is no difference between one-way nondeterministic communication complexity and nondeterministic communication complexity. The methods for proving lower bounds on nondeterministic communication complexity are presented in Section 2.5.3. It is shown there that 1-fooling sets provide lower bounds on nondeterministic communication complexity, and that the cardinality of the minimal exact-cover of a matrix $M(f, \Pi)$ is equal to the nondeterministic communication complexity of f according to Π. The ("deterministic") communication complexity is compared with nondeterministic communication complexity in Section 2.5.4. Among others an exponential gap between these two complexity measures is proved for a concrete language (L_Δ). On the other hand a special kind of a polynomial relation is shown: If $cc(f_n)$ and the nondeterministic communication complexity of f_n ($ncc(f_n)$) are not polynomially related for some computing problem $\{f_n\}_{n=1}^{\infty}$, then at least $cc(f_n)$ and $ncc(f_n^C)$ are polynomially related. The Las Vegas randomized protocols and Monte Carlo randomized protocols are introduced in Section 2.5.5, where also some nice examples showing the power of randomized computations are presented. In Section 2.5.6 some comparisons among determinism, nondeterminism, Las Vegas randomness, and Monte Carlo randomness are realized. Using concrete languages it is shown that:

(i) two-sided error Monte Carlo communication complexity can be exponentially smaller than nondeterministic communication complexity (than one-sided error Monte Carlo communication complexity),

(ii) one-sided error Monte Carlo communication complexity can be eponentially smaller than communication complexity, and

(iii) Las Vegas communication complexity can be approximately the root of communication complexity.

Note that the above mentioned results (i), (ii), and (iii) are of special interest, because for almost all fundamental complexity measures the proofs showing that randomness (nondeterminism) is more powerful than determinism are missing.

2.5.2 Nondeterministic Protocols

In this section the nondeterministic protocols and nondeterministic communication complexity are defined. Note that we shall investigate only nondeterministic protocols computing Boolean functions (i.e. one-output problems).

Definition 2.5.2.1 *Let f be a Boolean function with a set of input variables $X = \{x_1, \ldots, x_n\}$ for some $n \in \mathbb{N}$. A* **nondeterministic protocol** *over X is a pair $D_n = \langle \Pi, \Phi \rangle$, where*

(a) Π is a partition of X,

(b) Φ is a **communication relation** *in*

$$(\{0,1\}^m \times \{0,1,\$\}^* \cup \{0,1\}^{n-m} \times \{0,1,\$\}^*) \times (\{0,1\}^+ \cup \{\bar{0},\bar{1}\}),$$

where

(i) $m = |\Pi_{L,X}|$, $k = n - m = |\Pi_{R,X}|$,

(ii) Φ has the **prefix freeness property**:
For all $((\alpha, c), d)$, $((\alpha', c), d') \in \Phi$ d is not a proper prefix of d'.

(iii) if $((\alpha, c), d) \in \Phi$ and $d \in \{\bar{0},\bar{1}\}$ for some $\alpha \in \{0,1\}^m$, $c \in (\{0,1\}^+\$)^{2p}$, $p \in \mathbb{N}$ [for some $\alpha \in \{0,1\}^k$, $c \in (\{0,1\}^+\$)^{2p+1}$, $p \in \mathbb{N}$], then, for any $d' \in \{\bar{0},\bar{1}\}$, $q \in \mathbb{N}$, $\gamma \in \{0,1\}^k$, $c' \in (\{0,1\}^+\$)^{2q+1}$ [for any $d' \in \{\bar{0},\bar{1}\}$, $q \in \mathbb{N}$, $\gamma \in \{0,1\}^m$, $c' \in (\{0,1\}^+\$)^{2q}$], $((\gamma, c'), d') \notin \Phi$ (this property secures that the output value is always computed by the same computer independently of the input assignment).

A **computation of D_n on an input assignment $\alpha \in \{0,1\}^n$** *is a string $c = c_1\$c_2\$\ldots\$c_r\$c_{r+1}$, where*

(1) *$r \in \mathbb{N}$, $c_1, \ldots, c_r \in \{0,1\}^+$, $c_{r+1} \in \{\bar{0},\bar{1}\}$,*

(2) *for each integer ℓ, $0 \leq \ell \leq k$ we have*

(2.1) *if ℓ is even, then $((\alpha_{\Pi,L}, c_1\$c_2 \ldots\$c_\ell\$), c_{\ell+1}) \in \Phi$*

(2.2) *if ℓ is odd, then* $((\alpha_{\Pi,R}, c_1\$c_2 \ldots \$c_\ell\$), c_{\ell+1}) \in \Phi$.

r is called the **number of rounds of c**. *D_n is called an* **r-round nondeterministic protocol** *if every computation of D_n has at most r rounds. A one-round nondeterministic protocol is also called a* **one-way nondeterministic protocol**.

We say that $\boldsymbol{D_n}$ **computes 1 [0]** *for an input assignment $\alpha \in \{0,1\}^n$,* $\boldsymbol{D_n(\alpha) = 1}$ **[0]**, *if there exists a computation $c = c_1\$\ldots\$c_{r+1}$ of D_n on α with $c_{r+1} = \bar{1}$ [if every computation of D_n on α either ends with $\bar{0}$ or does not end with any result from $\{\bar{0},\bar{1}\}$]. We say that* $\boldsymbol{D_n}$ **computes f** *if, for each $\alpha \in \{0,1\}^n$, $f(\alpha) = D_n(\alpha)$. In what follows we shall also say that a computation is* **accepting [unaccepting]** *if it ends with $\bar{1}$ [$\bar{0}$].*

The **length of a computation c** *of D_n is the total length of all messages in c. For each $\alpha \in \{0,1\}^n$ such that $D_n(\alpha) = 1$, let* $\mathbf{ncc}(\boldsymbol{D_n}, \boldsymbol{\alpha})$ *denote the length of the shortest accepting computation of D_n on α.*

The **nondeterministic communication complexity of the nondeterministic protocol D_n** *is*

$$\mathbf{ncc}(\mathbf{D_n}) = \max\{\mathrm{ncc}(D_n, \alpha) \mid \alpha \in \{0,1\}^n \text{ and } D_n(\alpha) = 1\} \ .$$

Definition 2.5.2.2 *Let f be a Boolean function over a set of input variables $X = \{x_1, \ldots, x_n\}$. Let Π be a partition of X. The* **nondeterministic communication complexityof f according to Π** *is*

$$\mathbf{ncc}(\boldsymbol{f}, \boldsymbol{\Pi}) = \min\{\mathrm{ncc}(D) \mid D = \langle \Pi, \Phi\rangle \text{ for a communication relation } \Phi \text{ and } D \text{ computes } f\} \ .$$

The **nondeterministic communication complexity of f** *is*

$$\mathbf{ncc}(\boldsymbol{f}) = \min\{\mathrm{ncc}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\} \ .$$

Definition 2.5.2.3 *Let f be a Boolean function over a set X of input variables and let Π be a partition of X. The* **one-way nondeterministic communication complexity of f according to Π** *is*

$$\mathbf{ncc_1}(\boldsymbol{f}, \boldsymbol{\Pi}) = \min\{\mathrm{ncc}(D) \mid D = \langle \Pi, \Phi\rangle, \text{for a } \Phi, \text{is a one-way nondeterministic protocol computing } f\}.$$

The **one-way nondeterministic communication complexity of f** *is*

$$\mathbf{ncc_1}(\boldsymbol{f}) = \min\{\mathrm{ncc}_1(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\}.$$

Now, we illustrate the above stated definitions. First, we consider the language $\overline{L} = \{u = w_1w_2\ldots w_m \in \{0,1\}^* \mid m = 2^r \text{ for some } r \in \mathbb{N}, \text{ and } |w_i| = r$

for all $i = \{1, \ldots, m\}$, and $w_j = 1^r$ for $j = \mathrm{BIN}(w_{\mathrm{BIN}(w_1)})\}$ for which Theorem 2.4.3.4 establishes $\mathrm{cc}_1(h_n(\overline{L})) \geq (n/\log_2 n)^{1/2}$.

Example 2.5.2.4 Let $X = \bigcup_{i=1}^{2^r} W_i$, $W_i = \{x_{(i-1)r+1}, x_{(i-1)r+2}, \ldots, x_{ir}\}$ for every $i \in \{1, \ldots, 2^r\}$, be the set of input variables of $h_n(\overline{L})$ for each $n = r \cdot 2^r$, $r \geq 2$. We shall construct a one-way nondeterministic protocol $D_n = \langle \overline{\Pi}, \overline{\Phi} \rangle$ computing $h_n(\bar{L})$.

First, we give an informal description of D_n. $\overline{\Pi}$ divides any input $\alpha = w_1 w_2 \ldots w_{2_r}$ into the first half and the second half. So, C_I knows w_1. If $z = \mathrm{BIN}(w_1) \in \{1, \ldots, 2^{r-1}\}$ and $\mathrm{BIN}(w_z) \in \{1, \ldots, 2^{r-1}\}$, then C_I decides whether $\alpha \in \overline{L}$ or $\alpha \notin \overline{L}$ and C_I submits the result to C_{II}. If $z = \mathrm{BIN}(w_1) \in \{1, \ldots, 2^{r-1}\}$, and $\mathrm{BIN}(w_z) \notin \{1, \ldots, 2^{r-1}\}$, then C_I submits w_z to C_{II}. Then, C_{II} can decide whether $w_{\mathrm{BIN}(w_z)} = 1^r$ ($\alpha \in \overline{L}$) or $w_{\mathrm{BIN}(w_z)} \neq 1^r$ ($\alpha \notin \overline{L}$). If $z = \mathrm{BIN}(w_1) \notin \{1, \ldots, 2^{r-1}\}$, then C_I guesses $w_z = w_{\mathrm{BIN}(w_1)} \in \{0,1\}^r$ and it sends w_1 and w_z to C_{II}. Additionaly, if $j = \mathrm{BIN}(w_z) \in \{1, 2, \ldots, 2^{r-1}\}$ for the guessed word w_z, then C_I sends w_j to C_{II}. Now, C_{II} checks whether the guess w_z of C_I is correct or not. If not, then C_{II} reject the input. If the guess has been correct, C_{II} has all input parts sufficient to decide whether $\alpha \in \overline{L}$ or $\alpha \notin \overline{L}$.

Let $\overline{\Pi}_L = \bigcup_{i=1}^{2^{r-1}} W_i$. The communication relation Φ is defined as follows. For every input assignment $\alpha = w_1 w_2 \ldots w_{2^r}$:

- if $z = \mathrm{BIN}(w_1) \in \{1, \ldots, 2^{r-1}\}$, $j = \mathrm{BIN}(w_z) \in \{1, \ldots, 2^{r-1}\}$, and $w_j = 1^r$ [$\neq 1^r$], then

$$\begin{aligned} ((\alpha_{\overline{\Pi},L}, \lambda), 11) &\in \Phi, \\ [((\alpha_{\overline{\Pi},L}, \lambda), 00) &\in \Phi] \end{aligned}$$

- if $z = \mathrm{BIN}(w_1) \in \{1, \ldots, 2^{r-1}\}$, and $j = \mathrm{BIN}(w_z) \notin \{1, \ldots, 2^{r-1}\}$, then

$$((\alpha_{\overline{\Pi},L}, \lambda), 10 w_z) \in \Phi,$$

- if $z = \mathrm{BIN}(w_1) \notin \{1, \ldots, 2^{r-1}\}$, then, for each $\beta \in \{0,1\}^r$,

$$((\alpha_{\overline{\Pi},L}, \lambda), 01 w_1 \beta y) \in \Phi,$$

  where $y = 0^{r+1}$ for $\mathrm{BIN}(\beta) \in \{2^{r-1}+1, 2^{r-1}+2, \ldots, 2^r\}$, and $y = 1 w_{\mathrm{BIN}(\beta)}$ if $\mathrm{BIN}(\beta) \in \{1, 2, \ldots, 2^{r-1}\}$, [$\beta$ is the nondeterministic guess of the first computer for the input part $w_{\mathrm{BIN}(w_1)}$, assigned to the second computer],

- for every $\alpha_{\overline{\Pi},R}$

$$\begin{aligned} ((\alpha_{\overline{\Pi},R}, 11\$), \bar{1}) &\in \Phi, \\ ((\alpha_{\overline{\Pi},R}, 00\$), \bar{0}) &\in \Phi, \end{aligned}$$

- for every $\gamma \in \{0,1\}^r$ such that $w_{\mathrm{BIN}(\gamma)} = 1^r$ [$\neq 1^r$]

$$\begin{aligned} ((\alpha_{\overline{\Pi},R}, 10\gamma\$), \bar{1}) &\in \Phi \\ [((\alpha_{\overline{\Pi},R}, 10\gamma\$), \bar{0}) &\in \Phi], \end{aligned}$$

- if $\beta \neq w_{\mathrm{BIN}(w_1)} = w_z$, then, for every $y \in \{\lambda\} \cup \{0,1\}^r$, [i.e. if the guess was wrong]
$$((\alpha_{\overline{\Pi},R}, 01w_1\beta y\$), \bar{0}) \in \Phi,$$
- if $\beta = w_{\mathrm{BIN}(w_1)} = w_z$ ($z \in \{2^{r-1}+1, \ldots, 2^r\}$) and ($y = 1^{r+1}$ or $w_{\mathrm{BIN}(\beta)} = 1^r$ for $\mathrm{BIN}(\beta) \in \{2^{r-1}+1, \ldots, 2^r\}$), then
$$((\alpha_{\overline{\Pi},R}, 01w_1\beta y\$), \bar{1}) \in \Phi,$$
- if $\beta = w_{\mathrm{BIN}(w_1)}$ and ($y \neq 0^{r+1}$ or ($y = 0^{r+1}$ and $w_{\mathrm{BIN}(\beta)} \neq 1^r$)), then
$$((\alpha_{\overline{\Pi},R}, 01w_1\beta y\$), \bar{0}) \in \Phi.$$

Obviously, D_n computes $h_n(\overline{L})$ and $\mathrm{ncc}_1(D_n) = 3 \cdot r + 3 \leq 3 \cdot \log_2 n + 3$. □

Example 2.5.2.5 We consider the language $Star = \{w \in \{0,1\}^* \mid |w| = \binom{m}{2}$ for some $m \in \mathbb{N}$, $m \geq 2$, and $G(w)$ is a graph containing at least one star (a node which is connected directly via edges to all other nodes in $G(w)$, i.e. a node with the degree $n-1$) $\}$. Let $X = \{x_{ij} \mid i < j,\ i,j \in \{1,\ldots,m\}\}$ be the set of input variables of $h_n(Star)$ for some $n = \binom{m}{2}$, and let Π be any balanced partition of X. We claim $\mathrm{ncc}_1(h_n(Star), \Pi) \leq \log_2 n + 1$ for every $\Pi \in \mathrm{Bal}(X)$. We informally describe the work of the nondeterministic protocol $D = \langle \Pi, \Phi \rangle$ [with $\mathrm{ncc}(D) = \log_2 n + 1$] on any input $\alpha = \alpha_{12}\ldots\alpha_{1m}\alpha_{23}\ldots\alpha_{2m}\ldots\alpha_{m-1\,m}$. Let $d = \lceil \log_2 m \rceil$.

The first computer C_I of D checks whether there exists an $i = \{1,\ldots,m\}$ [a vertex v_i of G] such that $\alpha_{ij} = 1$ for every $j \in \{r \mid x_{ir} \in \Pi_L\}$. If not, then the first computer C_I sends message 0 and the second computer C_{II} gives the output $\bar{0}$. If there exist some i's with this property [the candidates among the nodes of $G(\alpha)$ for the degree $m-1$], then the first computer nondeterministically chooses one of them and submits the message $1\mathrm{BIN}_d^{-1}(i)$ to the second computer. The second computer accepts iff $\alpha_{iz} = 1$ for every $z \in \{\ell \mid x_{i\ell} \in \Pi_R\}$. □

In both previous examples we have used one-way nondeterministic protocols to compute the given computing problems. This is not by chance because, as we shall show in what follows, the one-way nondeterministic protocols are as powerful as the general nondeterministic protocols.

Theorem 2.5.2.6 *For every Boolean function f defined over a set of input variables X, and for every partition Π of X,*
$$\begin{aligned} \mathrm{ncc}_1(f,\Pi) &= \mathrm{ncc}(f,\Pi), \\ \textit{i.e.,}\quad \mathrm{ncc}_1(f) &= \mathrm{ncc}(f). \end{aligned}$$

Proof. Let $D = \langle \Pi, \Phi \rangle$ be a nondeterministic protocol such that $\mathrm{ncc}(D) = \mathrm{ncc}(f,\Pi) = k$. We construct a one-way nondeterministic protocol $D' = \langle \Pi, \Phi' \rangle$

computing f with $\mathrm{ncc}(D') = k$. The only message submitted from the first computer of D' to the second one is a guess of D' $[\Phi']$ of the communication of an accepting computation of D on a given input α. In fact $((\alpha_{\Pi,L}, \lambda), c) \in \Phi$ for every "valid" communication $c \in \{0,1\}^+$ corresponding to an accepting computation from $\alpha_{\Pi,L}$ (first computer) point of view. If c is not valid from the $\alpha_{\Pi,R}$ (second computer) point of view, then $((\alpha_{\Pi,L}, c\$), \bar{0}) \in \Phi$. If c is valid also from the second computer point of view, then $((\alpha_{\Pi,R}, c\$), \bar{1}) \in \Phi$.

Thus, we have proved $\mathrm{ncc}_1(f, \Pi) \leq \mathrm{ncc}(f, \Pi)$ for every f, and every Π. Since the opposite unequality $\mathrm{ncc}(f, \Pi) \leq \mathrm{ncc}_1(f, \Pi)$ is obvious, the proof of Theorem 2.5.2.6 is completed. □

Since there is no difference between nondeterministic communication complexity and one-way nondeterministic communication complexity we define the complexity classes only for nondeterministic communication complexity.

Definition 2.5.2.7 *We set for any* n, $m \in \mathbb{N}$, $m \leq n/2$, $\mathbf{NCOMM_n(m)} = \{f \in B_2^n \mid \mathrm{ncc}(f) \leq m\}$ *as the set of all Boolean functions of* n *variables computable within communication complexity* m.
Let $g : \mathbb{N} \to \mathbb{N}$ *be a function with the property* $g(n) \leq n/2$ *for any* $n \in \mathbb{N}$. *We set* $\mathbf{NCOMM(g(n))} = \{L \subseteq \{0,1\}^* \mid \mathrm{ncc}(h_n(L)) \leq g(n)$ *for any* $n \in \mathbb{N}\}$.

The definition of the nondeterministic protocols above (and so of nondeterministic communication complexity) is not the only possibility how to add nondeterminism to the protocols. We observe, that in Definition 2.5.2.1 of nondeterministic protocols the computers C_I and C_{II} are two independent nondeterministic devices. This means (in contrast to the fact that both of the computers C_I and C_{II} know the whole communication relation Φ, the partition Π and the complete definition of the problem) that one computer C receiving a message does not know automatically which nondeterministic decision (guess) has been done by the computer submitting this message. This computer C can learn it if this nondeterministic decision is coded in the message. Note, that we did it in this way in Examples 2.5.2.4 and 2.5.2.5. So, each of the two computers can be viewed as a nondeterministic computer having a private source of "advice" variables whose values decide about the nondeterministic choice of the message submitted. But none of C_I and C_{II} know the actual values of the advice variables of the other computer. This is the reason why one call the terms introduced above **private nondeterministic protocols** and **private nondeterministic communication complexity**.

Another possibility to define nondeterministic protocols is to consider a common (so-called **public**) source of advice variables. This means that each computer knows the values of advice variables which has been used to take a nondeterministic decision by the other computer. This can be viewed as an additional free exchange of the values of advice bits between the computers of the protocol model introduced in Definition 2.5.2.1.

There are several possibilities how to formalize the definition of public nondeterministic protocol. To do this formalization we prefare to consider a public nondeterministic protocol as a set of deterministic protocols (one deterministic protocol for each assignment to the advice variables), where an input α is accepted if at least one of these protocols accept α (i.e., if there exists an assignment to the advice variables leading to an accepting computation on α). The advantages of this formalization are:

(i) the possibility to measure the degree of nondeterminism by the number of advice variables, and

(ii) a good base for defining randomized protocols in Section 2.5.5.

Definition 2.5.2.8 *Let n, m be positive integers. Let $X = \{x_1, x_2, \ldots, x_n\}$ and $U = \{u_1, \ldots, u_m\}$ be sets of Boolean variables, and let Π be a partition of X. A* **public nondeterministic protocol D over X and U** *is any sequence of $2^{|U|} = 2^m$ protocols*

$$D^1 = \langle \Pi, \Phi^1 \rangle, D^2 = \langle \Pi, \Phi^2 \rangle, \ldots, D^{2^m} = \langle \Pi, \Phi^{2^m} \rangle$$

(each D^i corresponds to the assignment $\mathrm{BIN}_m^{-1}(i-1)$ from U to $\{0,1\}$ for any $i \in \{1, \ldots, 2^m\}$). A public nondeterministic protocol over X and U is also called a **$|U|$-public nondeterministic protocol over X**. *The set U is called* **the set of advice variables**.

For every input assignment $\alpha : X \to \{0,1\}$, we say that **D accepts α, $D(\alpha) = 1$** *(D computes $\bar{1}$ for α), if there exists an $i \in \{1, \ldots, 2^m\}$ such that D_i accepts α ($D_i(\alpha) = 1$). For every input assignment $\beta : X \to \{0,1\}$, we say that* **D rejects β, $D(\beta) = 0$** *($\bar{D}$ computes $\bar{0}$ for β), if D_j rejects β ($D_j(\beta) = 0$) for all $j \in \{1, \ldots, 2^m\}$. Let f be a Boolean function defined over the set X. We say that the* **public nondeterministic protocol D over X and U computes f** *if $D(\alpha) = f(\alpha)$ for every input assignment $\alpha : X \to \{0,1\}$.*

The **nondeterministic communication complexity of the public nondeterministic protocol D** *is*

$$\mathbf{pncc}(D) = \max\{\mathrm{cc}(D_i) \mid i = 1, 2, \ldots, 2^m\}.$$

Definition 2.5.2.9 *Let f be a Boolean function over a set X of input variables. Let m be a positive integer, and let Π be a partition of X.*

The **m-public nondeterministic communication complexity of f according to Π** *is*

$$m\text{-}\mathbf{pncc}(f, \Pi) = \min\{\mathrm{pncc}(D) \mid D = \langle \Pi, \Phi \rangle \text{ for a } \Phi \text{ is a } m\text{-public nondeterministic protocol computing } f\}.$$

The **public nondeterministic communication complexity of f according to Π** *is*

$$\mathbf{pncc}(f, \Pi) = \min\{m\text{-pncc}(f, \Pi) \mid 0 \leq m \leq |X|\}.$$

The ***m*-public nondeterministic communication complexity of f** *is*

$$m\text{-}\mathbf{pncc}(f) = \min\{m\text{-pncc}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\}.$$

The **public nondeterministic communication complexity of f** *is*

$$\mathbf{pncc}(f) = \min\{\mathrm{pncc}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\}.$$

Now we informally describe public nondeterministic protocols for the languages $\bar{L}$ and *Star* from Examples 2.5.2.4 and 2.5.2.5 in order to illustrate the power of public nondeterminism.

Example 2.5.2.10 Let X and $\overline{\Pi}$ have the same meaning as in Example 2.5.2.4, $|X| = n = r \cdot 2^r$, $r \geq 2$. We describe a $2r$-public nondeterministic protocol $D = \langle \overline{\Pi}, \Phi' \rangle$ computing $h_n(\bar{L})$ with $\mathrm{pncc}(D) = 1$. Let the set of advice variables be $U = U_1 \cup U_2$, $|U_i| = r$ for $i = 1, 2$. An assignment from U_1 to $\{0,1\}$ is a guess for the input part w_1. An assignment from U_2 to $\{0,1\}$ is a guess for the input part $w_{\mathrm{BIN}(w_1)}$.

If C_I has the input assignments $\alpha : \overline{\Pi}_L \to \{0,1\}$ and $\beta_1 : U_1 \to \{0,1\}$, $\beta_2 : U_2 \to \{0,1\}$, then it submits 1 iff $w_1 = \beta_1$ and $(w_{\mathrm{BIN}(w_1)} = \beta_2$ for $\mathrm{BIN}(w_1) \in \{1, 2, \ldots, 2^{r-1}\}$ or $\mathrm{BIN}(w_1) \notin \{1, 2, \ldots, 2^{r-1}\})$ and $(w_{\mathrm{BIN}(\beta_2)} = 1^r$ for $\mathrm{BIN}(\beta_2) \in \{1, 2, \ldots, 2^{r-1}\}$ or $\mathrm{BIN}(\beta_2) \notin \{1, 2, \ldots, 2^{r-1}\})$. Else C_I sends 0. C_{II} accepts if and only if the message submitted was 1 and $(w_{\mathrm{BIN}(w_1)} = \beta_2$ if $\mathrm{BIN}(w_1) \notin \{1, 2, \ldots, 2^{r-1}\})$ and $(w_{\mathrm{BIN}(\beta_2)} = 1^r$ if $\mathrm{BIN}(\beta_2) \notin \{1, 2, \ldots, 2^{r-1}\})$. Thus, an input is accepted if the nondeterministic guesses β_1 and β_2 for w_1 and $w_{\mathrm{BIN}(w_1)}$ are correct and $w_{\mathrm{BIN}(B_2)} = 1^r$. □

Example 2.5.2.11 We consider the language *Star* from Example 2.5.2.5. Let the set X of input variables be as described in Example 2.5.2.5, $|X| = n = \binom{m}{2}$ for some positive integer m. Let $d = \lceil \log_2 m \rceil$. We shall informally describe an r-public nondeterministic protocol computing $h_n(Star)$ for an arbitrary partition Π of X. Let the set of advice bits be U. An assignment β from U to $\{0,1\}$ is a guess for the name of one node which has to be connected to all other nodes. C_I sends 1 if its input does not contradict the guess β. Else, C_I sends 0. C_{II} accepts iff the bit submitted is 1 and the input part of C_{II} does not contradict the guess β. Thus, independently on Π, the protocol computes $h_n(Star)$ within communication complexity 1. □

The examples above show that public nondeterministic protocols can save communication by guessing the values of the crutial input variables distributed between C_I and C_{II}. In fact the whole input or a whole computation of a (deterministic) protocol may be guessed. This leads to the following observation.

Observation 2.5.2.12 *For every Boolean function f defined over a set X of input variables, and for every $\Pi \in \mathrm{Bal}(X)$,*

$$\lceil |X|/2 \rceil\text{-pncc}(f, \Pi) \leq 1.$$

Proof. For every input α, the public nondeterministic protocol guesses $\alpha_{\Pi,L}$. If the guess is correct C_I sends 1. If the guess has been false, then C_I sends 0. If the bit submitted is 1, then C_{II} knows $\alpha_{\Pi,R}$ and the guessed word which is $\alpha_{\Pi,L}$. Thus, C_{II} knows the whole input $\alpha = \Pi^{-1}(\alpha_{\Pi,L}, \alpha_{\Pi,R})$ and it can compute $f(\alpha)$. □

A direct consequence of Observation 2.5.2.12 is that $\text{pncc}(f) \leq 1$ for every Boolean function f. This means that there is no reason to deal with the complexity measure $\text{pncc}(f)$. But, it may be reasonable to study the trade-off between the public nondeterministic complexity and the number of advice variables. The following assertion is a simple extention of the ideas above.

Lemma 2.5.2.13 *Let n be a positive integer, and let X be a set of Boolean variables, $|X| = n$. For every Boolean function f over X, and for every positive integer r, $r \leq n/2$,*

$$r\text{-pncc}(f) \leq \lfloor n/2 \rfloor - r + 1.$$

Proof. Let Π be any partition from $\text{Bal}(X)$ with $|\Pi_L| = \lfloor n/2 \rfloor$. For every input α, the nondeterministic protocol $D = \langle \Pi, \Phi \rangle$ guesses the first r values of the input assignment $\alpha_{\Pi,L}$. If the guess is correct, C_I submits 1γ, where γ contains the rest of $\alpha_{\Pi,L}$. Else, C_I submits 0. After this C_{II} knows the whole input α and it can compute $f(\alpha)$. □

Observation 2.5.2.12 and Lemma 2.5.2.13 are the reasons why we shall not study the public nondeterministic communication complexity in what follows. But, we shall use the model of public nondeterministic protocols as the base for introducing randomized protocols in Section 2.5.5.

2.5.3 Lower Bounds on Nondeterministic Communication Complexity

Again, we present two lower bound proof methods, one based on the fooling set idea and another one based on the matrix representation of the computing problem. The method based on fooling sets is very close to the fooling set method for deterministic communication complexity, namely all fooling sets $\mathcal{A}(f, \Pi)$ with the property $\mathcal{A}(f, \Pi) \subseteq N^1(f)$ provide a direct lower bound on $\text{ncc}(f, \Pi)$. In the case of the matrix representation methods we change the tiling (exact-cover) method (working for communication complexity) for a method based on a cover of all ones in $M(f, \Pi)$ with (not necessarily disjoint) 1-monochromatic submatrices of $M(f, \Pi)$. The later method provides not only a lower bound but also the upper bounds. So, an optimal use of this technique can lead to ex-

act estimations on the nondeterministic communication complexity of concrete problems.

We start with the fooling set method.

Theorem 2.5.3.1 *Let f be a Boolean function over a set X of input variables, and let Π be a partition of X. Then, for every 1-fooling set $\mathcal{A}(f,\Pi)$ for f and Π*

$$\mathrm{ncc}(f,\Pi) \geq \lceil \log_2 |\mathcal{A}(f,\Pi)| \rceil .$$

Proof. Following Theorem 2.5.2.6 it is sufficient to prove

$$\mathrm{ncc}_1(f,\Pi) \geq \lceil \log_2 |\mathcal{A}(f,\Pi)| \rceil = r.$$

We do it by contradiction. Let $D = \langle \Pi, \Phi \rangle$ be a one-way nondeterministic protocol with $\mathrm{ncc}(D) < r$. Since the number of accepting computations $c\$\bar{1}$, $c \in \{0,1\}^+$, is at most $2^{\mathrm{ncc}(D)} \leq 2^{r-1} < |\mathcal{A}(f,\Pi)|$, there exist α, $\beta \in \mathcal{A}(f,\Pi)$ having the same accepting computation $c_1\$\bar{1}$ for some $c_1 \in \{0,1\}^+$. This implies that $c_1\$\bar{1}$ is also an accepting computation of D on $\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})$ and on $\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R})$, and so $D(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})) = D(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R})) = 1$. But this contradicts to the fact that $\mathcal{A}(f,\Pi)$ is a 1-fooling set (i.e. to the fact that $f(\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})) = 0$ or $f(\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R})) = 0$). □

We note that 0-fooling sets do not provide lower bounds on nondeterministic communication complexity (later we shall show also an example of $\mathrm{ncc}(f,\Pi) = \log_2(\log_2 m)$, where m is the size of the largest fooling set for f and Π). The reason is that from the existence of the rejecting computation $c\$\bar{0}$ for two inputs α, β we can only conclude the existence of the rejecting computation $c\$\bar{0}$ also for $\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})$ and $\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R})$. But this says nothing about the acceptance or the rejection of $\Pi^{-1}(\alpha_{\Pi,L}, \beta_{\Pi,R})$ and $\Pi^{-1}(\beta_{\Pi,L}, \alpha_{\Pi,R})$.

Thus, we can formulate the method nfool for proving lower bounds on the nondeterministic communication complexity.

Method nfool

Input: A Boolean function f with a set of input variables X.

Step 1: For each $\Pi \in \mathrm{Bal}(X)$ find a 1-fooling set $\mathcal{A}(f,\Pi)$

Step 2: Compute $d = \min\{|\mathcal{A}(f,\Pi)| \mid \Pi \in \mathrm{Bal}(X)\}$

Output: "$\mathrm{ncc}(f) \geq \lceil \log_2 d \rceil$".

Following the method nfool we see that all lower bounds proved on communication complexity using 1-fooling sets are also lower bounds on nondeterministic communication complexity. For instance, the 1-fooling sets used to prove a linear lower bound on the communication complexity of the context-free language L_R provides also the following lower bound on its nondeterministic communication complexity.

Corollary 2.5.3.2 *For any* $n \in \mathbb{N}$,

$$\mathrm{ncc}(h_n(L_R)) \geq n/8 - 2.$$

We have already noted that 0-fooling sets cannot be used to prove lower bounds on nondeterministic communication complexity. Another question may arise: Do one-way 1-fooling sets provide a direct lower bound on nondeterministic communication complexity? The reason for this question is the fact that $\mathrm{ncc}_1(f) = \mathrm{ncc}(f)$ for every Boolean function f. But already the results established show that the logarithm of the cardinality of a one-way 1-fooling set is not a lower bound on nondeterministic communication complexity because it is not a lower bound on the communication complexity (In Theorem 2.4.3.4 the one-way 1-fooling sets of the cardinality $2^{\sqrt{n/\log_2 n}}$ are constructed for the language $\overline{L}$, but $\mathrm{ncc}(h_n(\overline{L})) \leq \mathrm{cc}(h_n(\overline{L})) = O(\log_2 n)$.) Despite the fact that the answer to our question is clearly "no" we shall try to find the real reasons behind why one-way 1-fooling sets cannot be used to prove lower bounds on nondeterministic communication complexity. This effort will help us to find another lower bound proof method for nondeterministic communication complexity based on the matrix representation $M(f, \Pi)$ of f.

Let $D = \langle \Pi, \Phi \rangle$ be a one-way nondeterministic protocol computing a Boolean function f, and let $\mathcal{A} = \mathcal{A}_1(f, \Pi)$ be a one-way 1-fooling set for f and Π. If $\mathrm{ncc}(D) < \log_2 |\mathcal{A}|$, then there are two input parts $\alpha_{\Pi,L}$, $\beta_{\Pi,L} \in \mathcal{A}$ such that

(i) $\alpha_{\Pi,L} \neq \beta_{\Pi,L}$

(ii) $\exists\, \alpha_{\Pi,R}, \beta_{\Pi,R}$ and γ such that $f(\alpha) = f(\Pi^{-1}(\alpha_{\Pi,L}, \alpha_{\Pi,R})) = f(\beta) = 1$ and $1 = f(\Pi^{-1}(\alpha_{\Pi,L}, \gamma)) \neq f(\Pi^{-1}(\beta_{\Pi,L}, \gamma)) = 0$

(iii) $\exists$ a communication $c \in \{0,1\}^+$ such that $((\alpha_{\Pi,L}, \lambda), c) \in \Phi$, $((\beta_{\Pi,L}, \lambda), c) \in \Phi$, and $c\$\bar{1}$ is accepting computation for boths input α and β.

Why are the conditions (i), (ii) and (iii) not sufficient to get a contradiction? The reason is that (i), (ii) and (iii) do not imply anything about the acceptance of $\Pi^{-1}(\alpha_{\Pi,L}, \gamma)$ or of $\Pi^{-1}(\beta_{\Pi,L}, \gamma)$. This is because it is possible that $((\gamma, c\$), \bar{1}) \notin \Phi$ and yet $D(\Pi^{-1}(\alpha_{\Pi,L}, \gamma)) = 1$. So, for the input part $\alpha_{\Pi,L}$, there may exist several distinct communications from the first computer leading to an accepting computation in the dependence on the input assignment $\gamma : \Pi_R \to \{0,1\}$ read by the second computer.

The main observation following from our considerations is the following one. Let $D = \langle \Pi, \Phi \rangle$ be a one-way nondeterministic protocol. Let $a = |\Pi_L|$ and $b = |\Pi_R|$. Let $c \in \{0,1\}^+$ be a message with the property $((\delta, \lambda), c) \in \Phi$ for a $\delta \in S_1 = \{\mathrm{BIN}_a^{-1}(i_1), \ldots, \mathrm{BIN}_a^{-1}(i_k)\}$ and $((\omega, \lambda), c) \notin \Phi$ for any $\omega \notin S_1$, and with the property $((\gamma, c\$), \bar{1}) \in \Phi$ for $\gamma \in S_2 = \{\mathrm{BIN}_b^{-1}(j_1), \ldots, \mathrm{BIN}_b^{-1}(j_m)\}$ and $((\beta, c\$), \bar{1}) \notin \Phi$ for any $\beta \notin S_2$. Then, the accepting computation $c\$\bar{1}$ of D is the accepting computation exactly for inputs corresponding to the 1-monochromatic

submatrix of $M(f, \Pi)$ given by the intersection of the rows $i_1 + 1$, $i_2 + 1$, ... , $i_k + 1$ and the columns $j_1 + 1$, $j_2 + 1$, ... , $j_m + 1$. So, every accepting computation of D correponds to one 1-monochromatic submatrix of $M(f, \Pi)$. This fact leads to a new lower bound method.

Definition 2.5.3.3 *Let $M = [a_{ij}]$ be a Boolean matrix of a size $n \times m$ for some n, $m \in \mathbb{N} - \{0\}$. Let $S_1 = \{i_1, i_2, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, and $S_2 = \{j_1, j_2, \ldots, j_\ell\} \subseteq \{1, \ldots, m\}$. Then $\boldsymbol{M[S_1, S_2]}$ denotes the $k \times \ell$ submatrix of M consisting exactly of the elements $[b_{rs}]_{r=1,\ldots,k, s=1,\ldots,\ell}$ for $b_{rs} = a_{i_r j_s}$.*

Note that $M[S_1, S_2]$ is exactly the submatrix of M which can be obtained by S_1-row splitting of M immediately followed by S_2-column splitting of $M(S_1)$.

Definition 2.5.3.4 *Let M be a Boolean matrix, and let $M[S_1, R_1]$, $M[S_2, R_2]$, ... , $M[S_k, R_k]$ be some 1-monochromatic submatrices of M (not necessarily pairwise disjoint). We say $\boldsymbol{M[S_1, R_1]}$, $\boldsymbol{M[S_2, R_2]}$, ... , $\boldsymbol{M[S_k, R_k]}$* **cover all ones of** $\boldsymbol{M}$ *if each 1-element of M is also an element of one of the matrices $M[S_1, R_1]$, $M[S_2, R_2]$, ... , $M[S_k, R_k]$. Let* $\mathbf{Cov}(\boldsymbol{M})$ *be the least natural number t such that all 1's of M can be covered by t 1-monochromatic submatrices.*

Theorem 2.5.3.5 *Let f be a Boolean function with a set of input variables X, and let $\Pi \in \mathrm{Bal}(X)$. Then*

$$\mathrm{ncc}(f, \Pi) = \lceil \log_2 \mathrm{Cov}(M(f, \Pi)) \rceil .$$

Proof. Since $\mathrm{ncc}(f, \Pi) = \mathrm{ncc}_1(f, \Pi)$ for any f and Π, it is sufficient to prove $\mathrm{ncc}_1(f, \Pi) = \lceil \log_2 \mathrm{Cov}(M(f, \Pi)) \rceil$. First we show that

$$\mathrm{ncc}_1(f, \Pi) \leq \lceil \log_2 \mathrm{Cov}(M(f, \Pi)) \rceil .$$

To do it we construct, for each set of 1-monochromatic submatrices $M[S_1, R_1]$, $M[S_2, R_2]$, ... , $M[S_k, R_k]$ covering $M(f, \Pi)$, a one-way nondeterministic protocol $D = \langle \Pi, \Phi \rangle$ such that D computes f and $\mathrm{ncc}(D) = \lceil \log_2 k \rceil = d$. The idea of the construction is very simple. If the first computer has an input assignment $\alpha_{\Pi,L} : \Pi_L \to \{0, 1\}$ as a part of an input α , then it looks for 1-monochromatic submatrices from $\{M[S_1, R_1], M[S_2, R_2], \ldots , M[S_k, R_k]\}$ which have an intersection (cover at least one 1) with (of) the row of $M(f, \Pi)$ corresponding to $\alpha_{\Pi,L}$. Then, the first computer nondeterministically guesses a number such that $M[S_i, R_i]$ covers the 1 corresponding to the input α in $M(f, \Pi)$ (i.e., $((\alpha_{\Pi,L}, \lambda), \mathrm{BIN}_d^{-1}(i-1)) \in \Phi$ for every i such that $M[S_i, R_i]$ covers at least one 1 of the $(\mathrm{BIN}(\alpha_{\Pi,L}) + 1)$-th row of $M(f, \Pi)$). If the nondeterministic guess i of the first computer is correct (i.e., if the column corresponding to $\alpha_{\Pi,R}$ intersects the 1-monochromatic submatrix $M[S_i, R_i]$), then the second computer accepts; otherwise the second computer rejects.

To prove $\lceil \log_2 \mathrm{Cov}(M(f, \Pi)) \rceil \leq \mathrm{ncc}_1(f, \Pi)$ we show that, for every one-way nondeterministic protocol $D = \langle \Pi, \Phi \rangle$ computing f, one can construct

a cover of $M(f, \Pi)$ containing so many 1-monochromatic submatrices as the number of different accepting computations of D is. As we have already noted above all elements of $M(f, \Pi)$ corresponding to inputs having the same accepting computation $c\$\bar{1}$ form exactly one 1-monochromatic submatrix $M(c)$ (if $((\delta, \lambda), c) \in \Phi$ exactly for $\delta \in S_1 = \{\mathrm{BIN}(i_1), \ldots, \mathrm{BIN}(i_k)\}$ and if $((\gamma, c\$), \bar{1}) \in \Phi$ exactly for $\gamma \in R_1 = \{\mathrm{BIN}(j_1), \ldots, \mathrm{BIN}(j_m)\}$, then $c\$\bar{1}$ is the accepting computation exactly for inputs corresponding to the 1-monochromatic submatrix $M(c) = M(\{i_1+1, \ldots, i_k+1\}, \{j_1+1, \ldots, j_k+1\})$. Since for every 1 in $M(f, \Pi)$ there exists an accepting computation of D on the corresponding input the 1-monochromatic submatrices in $\{M(c) \mid c$ is an aceepting computation of $D\}$ cover all 1's in $M(f, \Pi)$. The fact that the number of the accepting computations of D is at most $2^{\mathrm{ncc}(D)}$ completes the proof. □

Taking Theorem 2.5.3.5, we get the following lower bound method for nondeterministic communication complexity.

Method cover

Input: A Boolean function f with a set of input variables X.

Step 1: For each $\Pi \in \mathrm{Bal}(X)$ find a minimal set $S(\Pi)$ of 1-monochromatic submatrices covering $M(f, \Pi)$.

Step 2: Compute $d = \min\{|S(\Pi)| \mid \Pi \in \mathrm{Bal}(X)\}$.

Output: "$\mathrm{ncc}(f) = \lceil \log_2 d \rceil$"

We note that the use of the method cover for proving lower bounds on $\mathrm{ncc}(f, \Pi)$ is in general no simple task because it means proving another lower bound – the nonexistence of small covers of the matrix $M(f, \Pi)$. The lower bound on the cardinality of the covers of $M(f, \Pi)$ can be immediately obtained only in the trivial cases when $M(f, \Pi)$ is a diagonal matrix, upper-triangle matrix, etc.

2.5.4 Deterministic Protocols Versus Nondeterministic Protocols

One of the crucial tasks in complexity theory is to decide whether nondeterminism is more powerful than determinism for concrete complexity measures. Most questions of this type remain unanswered (see, for instance, the P versus NP problem, NLOGSPACE versus DLOGSPACE, etc.). It is an interesting property of communication complexity that we are able to answer such questions by showing that nondeterminism may be much more powerful than determinism. First, we observe that there are languages for which the communication complexity is almost the same or the same as the nondeterministic one. For instance, following the lower bound proof of Theorem 2.3.3.1 we see that the proof may be easily adjusted to work also for nondeterministic communication complexity (the minimal cover of M' constructed

in the proof contains exactly $m+1$ 1-monochromatic submatrices), and so $\mathrm{ncc}(h_{2m}(\tilde{L})) = \mathrm{cc}(h_{2m}(\tilde{L})) = \mathrm{ncc}(h_{2m}(\tilde{L}), \Pi) = \lceil \log_2(m+1) \rceil$ for every balanced partition Π of $2m$ input variables and every $m \in \mathbb{N} - \{0\}$. Another example yields Theorem 2.3.5.4, where $\mathrm{cc}(h_n(L_R)) \geq n/8 - 2$ for the context-free language L_R is proved. Since the proof is based on the constructions of large 1-fooling sets it works also for nondeterministic communication complexity, i.e. $\mathrm{ncc}(h_n(L_R)) \geq n/8 - 2$. Thus, several lower bounds on communication complexity established in the literature work also for nondeterministic communication complexity because they are based on 1-fooling sets (the 0-fooling sets are used not so frequently).

We now give a general simulation of nondeterministic protocols by deterministic ones, and we show that this simulation is optimal in some sense.

Theorem 2.5.4.1 *Let m, n be positive integers, $m < n/2$. Then*

$$\mathrm{NCOMM}_n(m) \subseteq \mathrm{COMM}_n^1(2^m).$$

Proof. Let $f \in \mathrm{NCOMM}_n(m)$, i.e., there is a one-way nondeterministic protocol $D = \langle \Pi, \Phi \rangle$ computing f with $\mathrm{ncc}(D) \leq m$. The bound on the communication complexity of D implies that there are at most $2^{\mathrm{ncc}(D)} \leq 2^m$ different communication messages submitted by the first computer of D in the shortest accepting computations on inputs in $N^1(f)$. Now, we construct one-way protocol $D' = \langle \Pi, \Phi' \rangle$ computing f as follows. For an input α, $\Phi'(\alpha_{\Pi,L}, \lambda) = c_1 c_2 \ldots c_{2^m} \in \{0,1\}^{2^m}$, where $c_i = 1$ iff $((\alpha_{\Pi,L}, \lambda), d_i) \in \Phi$, where d_i is the lexicographically i-th communication of D. Then $\Phi'((\alpha_{\Pi,R}, c_1 c_2 \ldots c_{2^m}\$) = \bar{1}$ iff $\exists\, j \in \{1, \ldots, 2^m\}$ such that $c_j = 1$, and $((\alpha_{\Pi,R}, d_j\$), \bar{1}) \in \Phi$ for the j-th communication d_j of D. □

Corollary 2.5.4.2 *For any function $g : \mathbb{N} \to \mathbb{N}$, $g(n) \leq n/2$,*

$$\mathrm{NCOMM}(g(n)) \subseteq \mathrm{COMM}^1(2^{g(n)}).$$

Theorem 2.5.4.3 *For every $n = \binom{m}{2}$, $m \in \mathbb{N}$,*

(i) $\mathrm{cc}(h_n(L_\Delta)) \geq n/(64 \cdot 10^{10})$, *and*

(ii) $\mathrm{ncc}(h_n(L_\Delta)) \leq 2 + \lceil \log_2 n \rceil$.

Proof. The fact *(i)* has been proved in Theorem 2.3.3.2. We prove *(ii)*. Let $k = \lceil \log_2 n \rceil$. We construct a one-way nondeterministic protocol $D = \langle \Pi, \Phi \rangle$ for an arbitrary balanced partition Π of n input variables. For every input α, Φ is defined as follows:

$((\alpha_{\Pi,L}, \lambda), 11) \in \Phi$ if $G(\alpha)$ contains a triangle due to $\alpha_{\Pi,L}$ (independently on $\alpha_{\Pi,R}$),

$((\alpha_{\Pi,L}, \lambda), 01\mathrm{BIN}_k^{-1}(i)) \in \Phi$ for every i such that if the i-th edge is present (by $\alpha_{\Pi,R}$) then it forms together with some present edges of $\alpha_{\Pi,L}$ a triangle,

$((\alpha_{\Pi,L}, \lambda), 10\mathrm{BIN}_k^{-1}(j)) \in \Phi$ for every j such that the j-th edge is present in $\alpha_{\Pi,L}$,

$((\alpha_{\Pi,R}, 11\$), \bar{1}) \in \Phi$,

$((\alpha_{\Pi,R}, 01\mathrm{BIN}_k^{-1}(i)\$), \bar{1}) \in \Phi$ iff the i-th edge is present by $\alpha_{\Pi,R}$,

$((\alpha_{\Pi,R}, 10\mathrm{BIN}_k^{-1}(j)\$), \bar{1}) \in \Phi$ iff there exist two present edges in $\alpha_{\Pi,R}$ which form together with the j-th edge a triangle.

Obviously, D computes $h_n(L_\Delta)$ and $\mathrm{ncc}(D) \leq 2 + \lceil \log_2 n \rceil$. □

We observe that we were able to establish the exponential gap between the communication complexity and the nondeterministic communication complexity for the recognition of the language L_Δ because the lower bound proof on $\mathrm{cc}(h_n(L_\Delta))$ has been realized by 0-fooling sets and there exists no large 1-fooling set for $h_n(L_\Delta)$ and any balanced partition Π. This provides a very strong unclosure property of nondeterministic communication complexity classes according to the complement.

Let $\overline{L_\Delta} = \{\alpha \mid |\alpha| = \binom{m}{2}$ for some $m \in \mathbb{N}$, and $G(\alpha)$ does not contain any triangle $\}$.

Theorem 2.5.4.4 *For every* $n = \binom{m}{2}$, $m \in \mathbb{N}$,

(i) $\mathrm{ncc}(h_n(L_\Delta)) \leq 2 + \lceil \log_2 n \rceil$

(ii) $\mathrm{ncc}(h_n(\overline{L_\Delta})) \geq n/(64 \cdot 10^{10})$

Proof. Fact (i) has been proved in Theorem 2.5.4.3. The claim (ii) follows from the fact that the proof of Theorem 2.3.3.2 provides large 0-fooling sets for $h_n(L_\Delta)$ which implies directly the existence of large 1-fooling sets for $h_n(\overline{L_\Delta}) \equiv h_n((L_\Delta)^C)$ for $n = \binom{m}{2}$. (The fooling sets are the same only ones and zeros are exchanged in $M(h_n(\overline{L_\Delta}), \Pi)$ in the comparison with $M(h_n(L_\Delta), \Pi)$). □

The above results show that nondeterminism can be much more powerful than determinism for some computing problems. On the other hand we shall show in what follows that $m+1$ one-way deterministic communication bits can be more powerful than m nondeterministic communication bits for any positive integer m (i.e., nondeterminism is not able to compensate for one additional communication bit).

Theorem 2.5.4.5 *For every enough large $n \in \mathbb{N}$, and every $m \in \mathbb{N}$, $m < n$,*

$$\mathrm{COMM}^1_{2n}(m+1) - \mathrm{NCOMM}_{2n}(m) \neq \emptyset.$$

Proof. We consider the following two cases:

(1) $m \geq \log_2 n$. Since Lemma 2.4.3.6 claims

$$B_2^{2n}(n+m+1) \subseteq \mathrm{COMM}^1_{2n}(m+1)$$

we get

$$|\mathrm{COMM}^1_{2n}(m+1)| \geq 2^{2^{n+m+1}}.$$

Now, we give an upper bound on $|\mathrm{NCOMM}_{2n}(m)|$ by enumerating the number of all "different" nondeterministic protocols $D_{2n} = \langle \Pi, \Phi \rangle$ with $\mathrm{ncc}(D_{2n}) \leq m$. Obviously, the number of distinct balanced partitions Π is $\binom{2n}{n}$. Since $\mathrm{ncc}(D_{2n}) \leq m$, the number of different accepting computations of D_{2n} is at most 2^m. Let us enumerate the number of different Φ's providing exactly p $(1 \leq p \leq 2^m)$ accepting computations. As we have seen already earlier, each accepting computation of D_n corresponds to a set of inputs described exactly by a 1-monochromatic submatrix of $M(f, \Pi)$. Since $M(f, \Pi)$ is of the size $2^n \times 2^n$ the number of possible ways to choose p 1-monochromatic submatrices of $M(f, \Pi)$ is

$$\binom{(2^{2^n})^2}{p}.$$

Thus, the number of different $f \in B_2^{2n}$ computed by nondeterministic protocols within communication complexity m is at most

$$\begin{aligned}\binom{2n}{n} \cdot \sum_{p=1}^{2^m} \binom{(2^{2^n})^2}{p} &\leq 2^{2n} \cdot 2^m \cdot \binom{(2^{2^{n+1}})^2}{2^m} \leq \frac{2^{2n}\left(2^{2^{n+1}}\right)^{2^m}}{(2^m-2)!} \\ &\leq \frac{2^{2n} \cdot 2^{2^{n+m+1}}}{(n/4)!} < 2^{2^{n+m+1}}\end{aligned}$$

for sufficiently large n and $m \geq \log_2 n$. Thus, $|\mathrm{COMM}^1_{2n}(m+1)| > |\mathrm{NCOMM}_{2n}(m)|$ for sufficiently large n and $m \geq \log_2 n$.

(2) $m \leq \log_2 n$. While the proof for the case (1) was existential we shall prove the case (2) in a constructive way. Let, for any $m \in \mathbb{N}$,

$$L_{\mathrm{mod}\ m} = \{\alpha \in \{0,1\}^* \mid \#_0(\alpha) \bmod 2^m = 0\}.$$

Obviously, $h_{2n}(L_{\mathrm{mod}\,(m+1)}) \in \mathrm{COMM}^1_{2n}(m+1)$. Now assume Π is a balanced partition of $2n$ variables. We can easily observe that

$$\mathcal{A} = \{\alpha^0, \alpha^1, \ldots, \alpha^{2^{m+1}-1} \mid \alpha^i_{\Pi,L} = 0^i 1^{n-i} \text{ and } \alpha^i_{\Pi,R} = 0^{2^{m+1}-i} 1^{n-2^{m+1}+i} \text{ for every } i \in \{0, \ldots, 2^{m+1}-1\}\}$$

is a 1-fooling set for $h_{2n}(L_{\text{mod }(m+1)})$ and Π. Thus,

$$\text{ncc}(h_{2n}(L_{\text{mod }(m+1)}), \Pi) \geq m+1$$

for every balanced partition Π, i.e. $\text{ncc}(h_{2n}(L_{\text{mod }(m+1)})) \geq m+1$.

□

Above in Theorem 2.5.4.3 we have showed that there is an exponential gap between communication complexity and nondeterministic communication complexity. To achieve this result we have found the computing problem $h_n(L_\Delta)$ with large 0-fooling sets but small 1-fooling sets. Theorem 2.5.4.4 claims that this means that we have also large 1-fooling sets for the complementary problem $(h_n(L_\Delta))^{\mathbb{C}}$. So, $\text{ncc}((h_n(L_\Delta))^{\mathbb{C}}) = \Omega(n)$, i.e., nondeterministic communication complexity is "very strongly" unclosed according to complementation. A natural question arises: "Does there exists a Boolean function f with small $\text{ncc}(f)$ and $\text{ncc}(f^{\mathbb{C}})$, and large $\text{cc}(f)$?" In what follows we give the negative answer to this question by showing $\text{cc}(f) \leq \text{ncc}(f) \cdot (\text{ncc}(f^{\mathbb{C}}) + 2)$ for every Boolean function f.

Theorem 2.5.4.6 *Let f be a Boolean function defined on a set X of variables, and let Π be a partition of X. Then*

$$\text{cc}(f, \Pi) \leq \text{ncc}(f, \Pi) \cdot (\text{ncc}(f^{\mathbb{C}}, \Pi) + 2).$$

Proof. Let us consider the matrix $M(f, \Pi)$. The fact that f has nondeterministic communication complexity $r_1 = \text{ncc}(f, \Pi)$ implies that all ones in $M(f, \Pi)$ can be covered by at most 2^{r_1} 1-monochromatic submatrices. The fact $r_0 = \text{ncc}(f^{\mathbb{C}}, \Pi)$ implies that all zeros in $M(f, \Pi)$ can be covered by at most 2^{r_0} 0-monochromatic submatrices. We shall use the above facts to construct a protocol f within communication complexity $r_1 \cdot (r_0 + 2)$.

First, we give some needed denotations and observe some simple facts. Let $\overline{C} = \{C_1, \ldots, C_m\}$, $m \leq 2^{r_0}$, be a set of 0-monochromatic submatrices of $M(f, \Pi)$, and let $\overline{H} = \{H_1, \ldots, H_\ell\}$, $\ell \leq 2^{r_1}$, be a set of 1-monochromatic submatrices of $M(f, \Pi)$ covering all ones in $M(f, \Pi)$. Let A_i denote the submatrix of $M(f, \Pi)$ formed by those rows of $M(f, \Pi)$ that meet C_i, and let B_i denote the submatrix of $M(f, \Pi)$ formed by those columns of $M(f, \Pi)$ that meet C_i. Let $\text{int}(A_i)$ and $\text{int}(B_i)$ respectively denote the number of 1-monochromatic submatrices from $\overline{H}$ having a nonempty intersection with A_i and B_i respectively. Since the intersection of A_i and B_i is exactly C_i, and C_i is a 0-matrix, we get that no matrix $H_j \in \overline{H}$ (for any j) has nonempty intersections with both A_i and B_i, i.e., $\text{int}(A_i) + \text{int}(B_i) \leq \ell = |\overline{H}|$. Let $\overline{C_1} = \{C_k \in \overline{C} \mid \text{int}(A_k) \leq \lceil \ell/2 \rceil\}$, and let $\overline{C_2} = \overline{C} - \overline{C_1} \subseteq \{C_s \in \overline{C} \mid \text{int}(B_s) \leq \ell/2\}$.

Now, we describe the first two rounds of a protocol $D = \langle \Pi, \Phi \rangle$ computing f. Let α be an input. The first computer having α_L looks at the row of $M(f, \Pi)$ corresponding to α_L to see whether it intersects any of the 1-monochromatic

submatrices in $\overline{C_1}$. If so, it sends the message 1γ, where γ is the binary code of the smallest index j such that $C_j \in \overline{C_1}$ and C_j intersects $\text{row}_{\text{BIN}(\alpha_L)+1}(M(f,\Pi))$. If not, the first computer sends 0.

If the second computer receives 0, it looks at the column corresponding to its input α_R to see whether it intersects any of the 1-monochromatic submatrices in $\overline{C_2}$. If so, it sends the message 1β, where β is the binary name of such a 1-monochromatic submatrix (if there are several matrices with this property, one can again to choose the one with the "smallest" name). If not, the second computer sends 0. If the second computer receives 1γ, it sends 1 to the first computer.

Now, let us discuss all three possible situations after the first two rounds.

Case 1: The current communication is "00", i.e., both computers failed to find an appropriate 0-submatrix. Since $\overline{C_1} \cup \overline{C_2} = \overline{C}$ and the set $\overline{C}$ covers all zeros in $M(f,\Pi)$ we get that $f(\alpha) = f(\Pi^{-1}(\alpha_L, \alpha_R)) = 1$.

Case 2: Let the first message be $1\gamma = 1\text{BIN}_z^{-1}(k)$ for $z = \lceil \log_2 m \rceil$. After receiving 1γ the second computer knows that $f(\alpha)$ belongs to the submatrix A_k. Since $C_k \in \overline{C_1}$ we obtain $\text{int}(A_k) \leq \lceil \ell/2 \rceil \leq 2^{r_1-1}$. This means that all ones in A_k can be covered by at most 2^{r_1-1} 1-monochromatic submatrices of A_k. (Note that these 1-submatrices are all intersections of A_k with 1-monochromatic submatrices $H_1, \ldots, H_\ell$ of $M(f,\Pi)$.)

Case 3: Let the first communication message be "0" and the second one be $1\beta = 1\text{BIN}_z^{-1}(d)$. After receiving 1β the first computer knows that $f(\alpha)$ belongs to the submatrix B_d. Since $C_d \in \overline{C_2}$ we have $\text{int}(B_d) \leq \lceil \ell/2 \rceil \leq 2^{r_1-1}$, i.e., all ones in B_d can be covered by at most 2^{r_1-1} 1-monochromatic submatrices.

Thus after the first two rounds either both computers know $f(\alpha)$ or both computers know $f(\alpha)$ lies in a matrix $M_1 (\in \{A_k, B_m\})$ whose ones can be covered by at most 2^{r_1-1} 1-monochromatic submatrices of M_1 (and by at most 2^{r_0} 0-monochromatic submatrices). Following the same construction (i.e., defining 1-monochromatic submatrices and 0-monochromatic submatrices covering ones and zeros in M_1, and defining new actual sets $\overline{C_1}$ and $\overline{C_2}$) and the same kind of information exchange in the next two rounds for the matrix M_1 as described above for $M(f,\Pi)$, we get either the result $f(\alpha)$ or a matrix M_2 whose ones can be covered by 2^{r_1-2} 1-monochromatic submatrices of M_2. Continuing in this way both computers know the result $f(\alpha)$ after at most r_1 rounds. Since each information exchange in rounds $2i$, $2i+1$ has the length at most $2 + z = 2 + \lceil \log_2 \ell \rceil \leq 2 + r_0$ we obtain that the length of the communication is bounded by $r_1(2 + r_0)$. □

A direct consequence of Theorem 2.5.4.6 is the following result.

Theorem 2.5.4.7 *For every Boolean function f*

$$\mathrm{cc}(f) \leq \mathrm{ncc}(f) \cdot (\mathrm{ncc}(f^C) + 2).$$

We call attention to the fact that Theorem 2.5.4.6 cannot be essentially improved. Later, in Lemma 2.5.5.9 we give a function $f_n \in B_2^n$ for any $n \in \mathbb{N}, n \geq m^2$, such that $\mathrm{cc}(f_n^C, \overline{\Pi}) = n$, $\mathrm{ncc}(f_n, \overline{\Pi}) = \sqrt{n} + \log_2 n$, and $\mathrm{ncc}(f_n^C, \overline{\Pi}) \leq \sqrt{n}(\log_2 \sqrt{n} + 1)$.

2.5.5 Randomized Protocols

In this subsection and in the next one we shall study two kinds of randomization of protocols (probabilistic protocols) – Las Vegas randomization and Monte Carlo randomization. This section gives the basic definitions of randomized protocols and presents some examples illustrating the computational power of probabilistic (randomized) computations (communications). Section 2.5.6 proves results establishing some relations between determinism and nondeterminism on one side, and randomness on the other.

Generally, Las Vegas probabilistic algorithms are nondeterministic algorithms (after each step the algorithm may randomly decide what to do in the next step) whose each of possibly many computations on a given input must lead to the correct result. Note that nondeterministic algorithms for the language recognition require only the existence of at least one accepting computation if the given input is in the given language. Another difference between nondeterministic algorithms and Las Vegas algorithms is that the complexity of a nondeterministic algorithm A working on an input α is the complexity of the shortest correct computation of A on α while the complexity of a Las Vegas algorithm B working on an input α is the "average" of complexities of all computations of B on α. More precisely, each computation C of B has assigned its probability (weight for computing the "average" value) which is the product of probablities of all random decisions made in C. In what follows we shall study only the basic model in which all computations have the same probability to be executed.

To introduce Monte Carlo probabilistic algorithms (protocols) one can again start from nondeterministic computing models. Again, the first difference is that instead of guessing the next action (communication) from the allowed set of steps (communications), we several times "toss the coin" and make the move as a function of the outcome. This makes a difference in the definition of accepting: while in nondeterminism the input is accepted iff there is at least one accepting computation on the input, in the Monte Carlo algorithms we consider the probability of getting an accepting computation. With each given computation one associates a probability (in the same way as for Las Vegas algorithms) which is the product of the probabilities at the coin-tossing steps of the computation. The probability of accepting an input is the sum of probabilities associated with the accepting computations of the algorithm on the given input.

Two-sided-error Monte Carlo randomization means that the input is accepted iff the probability of its accepting is greater than $\frac{2}{3}$, and the input is rejected iff the probability of its accepting is smaller than $\frac{1}{3}$. If the probability of accepting lies between $\frac{1}{3}$ and $\frac{2}{3}$ for some input, then the randomized algorithm is unable to decide whether the input has to be accepted or not. One-sided-error Monte Carlo randomization has a yet harder restriction on the definition of acceptance. The input is rejected iff all computations on this input are unaccepting (the probability of accepting is equal to zero), and the input is accepted if the probability of its accepting is greater than $\frac{1}{2}$. The inputs whose probability of accepting is greater than zero and smaller than $\frac{1}{2}$ are not allowed (if they are allowed, then the protocol is not able to decide whether they are accepted or not).

Obviously, the main practical interest is in the design of Las Vegas probabilistic algorithms because they give always the correct result, and if they are quicker than the best known deterministic algorithms for a given problem, then they provide the most effective reliable solution of the problem. But Monte Carlo probabilistic algorithms are also suitable for practical applications because (especially in the one-side-error case) repeating the work of such algorithm on a given input several times one can get a result whose probability to be correct is as large as one wishes. For instance, performing three independent computations of a one-sided-error Monte Carlo algorithm A on an input word α one get three outputs. If at least one output is 1, then the right result is surely 1 (acceptance). If all three outputs are zeros, then the result 0 (rejection) is correct with the probability $1 - (\frac{1}{2})^3 = \frac{7}{8}$. So, after performing k computations on the given input α we know either surely the fact that the input α is accepted or the fact that the probability of the correctness of the rejection of α is at least $1 - (\frac{1}{2})^k$.

One can get the definitions of Monte Carlo protocols and Las Vegas protocols by taking the nondeterministic protocols and changing the definition of acceptance in the appropriate ways. But we have introduced two kinds of nondeterminism in Section 2.5.2 – the standart (private) one and the public one. Here, we prefer to use public nondeterministic protocols as the base for defining randomized protocols because of the simplicity of such definition. One need not to take too much care of the difference between the public source of random bits and the private sources of random bits because this difference is at most a logarithm of the input length. (Note that this contrasts to the case of nondeterministic protocols, where public nondeterminism is extremely powerful). This public model will enable us to directly measure the amount of randomness used. For a given number of bits m, we shall define the m-randomized protocol as a set of 2^m deterministic protocols, where each one of these deterministic protocols realizes the computations corresponding to one concrete choice of the values for the "tossing the coin" m times before their own computation of the protocol. Thus, our interpretation is that both computers get the whole result of "tossing the coin" and depending on these m random bits they start to work deterministically on the given input.

Definition 2.5.5.1 *Let n, m be positive integers. Let $X = \{x_1, x_2, \ldots, x_n\}$ and $U = \{u_1, \ldots, u_m\}$ be sets of Boolean variables, and let Π be a partition of X. A* **randomized protocol $\overline{D}$ over X and U** *is any sequence of 2^m protocols*

$$D^1 = \langle \Pi, \Phi^1 \rangle, D^2 = \langle \Pi, \Phi^2 \rangle, \ldots, D^{2^m} = \langle \Pi, \Phi^{2^m} \rangle$$

(each D^i corresponds to the random sequence $\mathrm{BIN}_m(i-1)$). A randomized protocol over X and U is also called a **$|U|$-randomized protocol over X**. *The set U is called* **the set of random variables**. *For every input $\alpha \in \{0,1\}^n$ the* **probability of acceptance of α by $\overline{D}$** *is*

$$Pb(\overline{D}, \alpha) = | \{i \in \{1, \ldots, 2^m\} \mid D^i(\alpha) = 1\} | / 2^m.$$

□

Definition 2.5.5.2 *Let n, m be positive integers, and let $\overline{D}$ be an m-randomized protocol over a set of n variables. Let $f \in B_2^n$. We say that $\overline{D}$ is a* **one-sided-error Monte Carlo m-protocol computing f** *if*

(i) $\forall \alpha \in N^1(f) : \mathrm{Pb}(\overline{D}, \alpha) \geq \frac{1}{2}$

(ii) $\forall \beta \in N^0(f) : \mathrm{Pb}(\overline{D}, \beta) = 0.$

We say that $\overline{D}$ is a **two-sided-error Monte Carlo m-protocol computing f** *if*

(iii) $\forall \alpha \in N^1(f) : \mathrm{Pb}(\overline{D}, \alpha) > \frac{2}{3}$

(iv) $\forall \beta \in N^0(f) : \mathrm{Pb}(\overline{D}, \beta) < \frac{1}{3}.$

We say that $\overline{D}$ is a **Las Vegas m-protocol computing f** *if*

(v) $\forall \alpha \in N^1(f) : \mathrm{Pb}(\overline{D}, \alpha) = 1$

(vi) $\forall \beta \in N^0(f) : \mathrm{Pb}(\overline{D}, \beta) = 0.$ □

Observation 2.5.5.3 *Each one-sided-error Monte Carlo m-protocol computing a Boolean function f is a public nondeterministic protocol computing f.*

Proof. Of course, each public nondeterministic protocol D computing f may be viewed as a m-randomized protocol for some m, where $\forall \alpha \in N^1(f) : \mathrm{Pb}(\overline{D}, \alpha) > 0$ (for each α with $f(\alpha) = 1$ there exists at least one accepting computation of D on α), and $\forall \beta \in N^0(f) : \mathrm{Pb}(\overline{D}, \beta) = 0$ (for each β such that $f(\beta) = 0$ all computations of D are unaccepting). Thus, the condition (i) of Definition 2.5.5.2 is a harder restriction than the above restriction for public nondeterministic protocols. □

Definition 2.5.5.4 *Let n, m be positive integers, and let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. Let $\overline{D}$ be an m-randomized protocol $D^1, D^2, \ldots, D^{2^m}$ over X. The* **communication complexity of $\overline{D}$** *is*

$$\mathbf{cc}(\overline{\boldsymbol{D}}) = \max\{\mathrm{cc}(D^i) \mid i = 1, \ldots, 2^m\}.$$

Let f be a Boolean function over X. We define the **one-sided-error m-Monte Carlo communication complexity of f according to a partition Π** $\in \mathrm{Abal}(X)$ *as*

$$\boldsymbol{m}\text{-}\mathbf{1MCcc}(\boldsymbol{f}, \boldsymbol{\Pi}) = \min\{\mathrm{cc}(\overline{D}) \mid \overline{D} = \langle \Pi, \Phi^1 \rangle, \ldots, \langle \Pi, \Phi^{2^m} \rangle \text{ for arbitrary } \Phi^1, \ldots, \Phi^{2^m}, \text{ and } \overline{D} \text{ is a one-sided-error Monte Carlo } m\text{-protocol computing } f\}.$$

We define the **two-sided-error m-Monte Carlo communication complexity of f according to a partition Π** $\in \mathrm{Abal}(X)$ *as*

$$\boldsymbol{m}\text{-}\mathbf{2MCcc}(\boldsymbol{f}, \boldsymbol{\Pi}) = \min\{\mathrm{cc}(\overline{D}) \mid \overline{D} = \langle \Pi, \Phi^1 \rangle, \ldots, \langle \Pi, \Phi^{2^m} \rangle \text{ for arbitrary } \Phi^1, \ldots, \Phi^{2^m} \text{ such that } \overline{D} \text{ is a Monte Carlo } m\text{-protocol computing } f\}.$$

The **one-sided-error m-Monte Carlo communication complexity of f** *is*

$$\boldsymbol{m}\text{-}\mathbf{1MCcc}(\boldsymbol{f}) = \min\{m\text{-1MCcc}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\},$$

and the **(two-sided-error) m-Monte Carlo communication complexity of f** *is*

$$\boldsymbol{m}\text{-}\mathbf{2MCcc}(\boldsymbol{f}) = \min\{m\text{-2MCcc}(f, \Pi) \mid \Pi \in \mathrm{Bal}(X)\}.$$

□

Definition 2.5.5.5 *Let n, m be positive integers, and let $X = \{x_1, \ldots, x_n\}$. Let $\overline{D}$ be a Las Vegas m-protocol computing a Boolean function f over X. The* **Las Vegas communication complexity of $\overline{D}$ on input $\alpha \in \{0,1\}^n$** *is*

$$\mathbf{pcc}(\overline{\boldsymbol{D}}, \boldsymbol{\alpha}) = 2^{-m} \sum_{i=1}^{2m} \mathrm{cc}(D^i, \alpha),$$

where $\mathrm{cc}(D^i, \alpha)$ is the length of the communication of D^i on α.
The **Las Vegas communication complexity of $\overline{D}$** *is*

$$\mathbf{pcc}(\overline{\mathbf{D}}) = \max\{\mathrm{pcc}(\overline{D}, \alpha) \mid \alpha \in \{0,1\}^n\}.$$

The **m-Las Vegas communication complexity of f according to a partition Π of X** *is*

$$\boldsymbol{m}\text{-LVcc}(\boldsymbol{f},\boldsymbol{\Pi}) = \min\{\text{pcc}(\overline{D}) \mid \overline{D} = \langle \Pi, \Phi^1\rangle, \ldots, \langle \Pi, \Phi^{2^m}\rangle \text{ for arbitrary } \Phi^1, \ldots, \Phi^{2^m}, \text{ and } \overline{D} \text{ is a Las Vegas } m\text{-protocol computing } f\}.$$

The ***m*-Las Vegas communication complexity of *f*** *is*

$$\boldsymbol{m}\text{-LVcc}(\boldsymbol{f}) = \min\{m\text{-LVcc}(f,\Pi) \mid \Pi \in \text{Bal}(X)\}.$$

□

Now we show an example of the work of a randomized protocol.

Example 2.5.5.6 We consider the language $Un = \{xy \mid x \neq y, |x| = |y|, x, y \in \{0,1\}^+\}$. Let $X_{2n} = \{x_1, \ldots, x_{2n}\}$ be the set of input variables of $h_{2n}(Un)$, and let $\overline{\Pi}_{2n} \in \text{Bal}(X_{2n})$ be the partition defined by $\overline{\Pi}_{2n}(x_i) = 1$ for $i = 1, \ldots, n$. Since the matrix $\text{M}(h_{2n}(Un), \overline{\Pi}_{2n})$ is the 0-diagonal matrix we know that $\text{cc}(h_{2n}(Un)), \overline{\Pi}_{2n}) = n$. Let $m = \lceil 2\log_2 n \rceil$.

Now, we show that $m\text{-1MCcc}(h_{2n}(Un), \overline{\Pi}_{2n}) \leq \lceil 2\log_2 n \rceil$. Let $p_1, \ldots, p_r$ be all prime numbers such that $2 \leq p_i \leq n^2$ for every $i = 1, \ldots, r$. For sufficiently large n we know that r is approximately $\frac{n^2}{\log_e n}$. At the beginning a binary code of a number $i \in \{1, \ldots, r\}$ is randomly chosen. Then, the protocol $D^i = \langle \overline{\Pi}_{2n}, \Phi^i \rangle$ works as follows. The first computer computes the remainder x' of its input x modulo p_i and sends x' to the second computer. Receiving x' the second computer computes the remainder y' of its input y modulo p_i, and compares it with x'. If they are distinct, the second computer concludes that $x \neq y$ (i.e., it accepts). If they are the same, it concludes that x=y (i.e., it does not accept).

If $x = y$ (i.e., $xy \notin N^0(h_{2n}(Un))$), then each protocol D^j for every $j \in \{1, \ldots, r\}$ reaches the right conclusion (i.e., $\forall \beta \in N^0(h_{2n}(Un)) : \text{Pb}(\overline{D}, \beta) = 0$). If x and y are different, however, then it could happen that x' and y' are the same and the protocol reaches the wrong conclusion. This happens if p_i divides $\text{BIN}(x) - \text{BIN}(y)$. Since $|\text{BIN}(x) - \text{BIN}(y)| < 2^n$, $\text{BIN}(x) - \text{BIN}(y)$ has fewer than n different prime divisors. On the other hand r is aproximately $n^2/2\log_e n$, and so the probability that one chooses from r primes one of the divisors of $|\text{BIN}(x) - \text{BIN}(y)|$ tends to zero. Thus, for sufficiently large n, $\text{Pb}(\overline{D}, \alpha) \geq 1/2$ holds for every $\alpha \in N^1(h_{2n}(Un))$. □

Following Observation 2.5.5.3 we get immediately the following result.

Theorem 2.5.5.7 *For any Boolean function $f \in B_2^n$, any partition Π of n variables, and any $m \in \mathbb{N}$*

$$\text{ncc}(f,\Pi) - m \leq m\text{-pncc}(f,\Pi) \leq m\text{-1MCcc}(f,\Pi).$$

□

Theorem 2.5.5.7 renders the fact that, for every f and Π, 1-fooling sets for f and Π, and $\mathrm{Cov}(M(f,\Pi))$ provide direct lower bounds on m-1MCcc$(f,\Pi)+m$ (in the same way as they do for $\mathrm{ncc}(f,\Pi)$). Note that the 0-diagonal matrix used to get the linear lower bound on $\mathrm{cc}(h_{2n}(Un),\overline{\Pi}_{2n})$ in Example 2.5.5.6 corresponds to a 0-fooling set, and so we were able to show an exponential gap between communication complexity and one-sided-error $(2\log_2 n)$-Monte Carlo communication complexity. But taking the language $Eq = \{\mathrm{ww} \mid \mathrm{w} \in \{0,1\}^*\}$ (which is a complement of Un in some sense) the matrix $\mathrm{M}(h_{2n}(Eq),\overline{\Pi}_{2n})$ is the usual diagonal matrix corresponding to the 1-fooling set $N^1(h_{2n}(Eq))$ of the cardinality 2^n.

Thus m-1MCcc$(h_{2n}(Eq),\overline{\Pi}_{2n}) \geq n-m = \mathrm{cc}(h_{2n}(Eq),\overline{\Pi}_{2n}) - m$ for every $m \in \mathbb{N}$. This is not only an example where the one-sided-error Monte Carlo randomness does not help. It is also an example showing that two-sided-error Monte Carlo randomness is more powerful than one-sided-error Monte Carlo randomness.

Lemma 2.5.5.8 *Let $X_{2n} = \{x_1,\ldots,x_{2n}\}$ be a set of $2n$ input variables, and let $\overline{\Pi}_{2n} \in \mathrm{Bal}(X_{2n})$ be defined as $\overline{\Pi}_{2n}(x_i) = 1$ for every $i \in \{1,\ldots,n\}$. Then*

(i) *m-1MCcc$(h_{2n}(Eq),\overline{\Pi}_{2n}) \geq n-m$ for every $m \in \mathbb{N}$, and*

(ii) *$(2\log_2 n)$-2MCcc$(h_{2n}(Eq),\overline{\Pi}_{2n}) \leq 2\log_2 n$ for sufficiently large n.*

Proof. The fact (i) is proved above. To see (ii) take the protocol $\overline{D}$ from Example 2.5.5.6. Let $\overline{D}_1$ work exactly as $\overline{D}$ does except that $\overline{D}_1$ accepts iff $\overline{D}$ does not accept ($\overline{D}_1$ does not accept iff $\overline{D}$ does). Then, for each $\beta \in N^1(h_{2n}(Eq)) = \{\mathrm{ww} \mid \mathrm{w} \in \{0,1\}^n\}$, $\mathrm{Pb}(\overline{D}_1,\beta) = 1 > \frac{2}{3}$. For each $\alpha \in N^0(h_{2n}(Eq)) = \{xy \mid x,y \in \{0,1\}^n, x \neq y\}$, $\mathrm{Pb}(\overline{D}_1,\alpha) \leq 1/3$ for sufficiently large n because $\mathrm{Pb}(\overline{D}_1,\alpha)$ tends to zero as shown in Example 2.5.5.6. □

Above, we have presented two examples showing the power of Monte Carlo probabilistic communication algorithms. Obviously, the most realistic and suitable model for practical computations is Las Vegas probabilistic computation. Thus, we give one more example showing that Las Vegas protocols may be much more powerful than deterministic ones.

We consider the language

$$L_{\exists\forall} = \{x_1x_2\ldots x_my_1y_2\ldots y_m \mid m \in \mathbb{N}, \text{for } \forall i \in \{1,\ldots,m\},\ x_i,y_i \in \{0,1\}^m, \text{ and } \exists j \in \{1,\ldots,m\} \text{ that } x_j = y_j\}.$$

Lemma 2.5.5.9 *Let n,m be two positive integers, $n = m^2$. Let $X = (\bigcup_{i=1}^m X_i) \cup (\bigcup_{i=1}^m Y_i)$ be a set of n Boolean variables, where $X_i = \{x_{i1},\ldots,x_{im}\}, Y_i = \{y_{i1},\ldots,y_{im}\}$ for every $i \in \{1,\ldots,m\}$. Let $\overline{\Pi} \in \mathrm{Bal}(X)$ be defined as $\overline{\Pi}_L = \bigcup_{i=1}^m X_i$. Then*

(i) $\mathrm{ncc}(h_{2n}(L_{\exists\forall}),\overline{\Pi}) = m + \lceil \log_2 m \rceil$,

(ii) $\mathrm{cc}(h_{2n}(L_{\exists\forall}), \overline{\Pi}) = n = m^2$,

(iii) $\lceil \log_2 m \rceil^2\text{-LVcc}(h_{2n}(L_{\exists\forall}), \overline{\Pi}) \leq m(\lceil \log_2 m \rceil^2 + 4)$ *for sufficiently large* m*'s.*

Proof.

(i) To prove the lower bound on nondeterministic protocols we use the method foolfix. Consider the sets

$$\mathcal{A}_k = \{0^{m(k-1)}\alpha_k 0^{m(m-k)} 1^{m(k-1)} \alpha_k 1^{m(m-k)} \mid \alpha_k \in \{0,1\}^m - \{0^m, 1^m\}\}$$

for every $k \in \{1, \ldots, m\}$. Obviously, for every $k \in \{1, \ldots, m\}$, $\mathcal{A}_k$ is a 1-fooling set for $h_n(L_{\exists\forall})$ and $\overline{\Pi}$. We claim that also $\mathcal{A} = \bigcup_{k=1}^{m} \mathcal{A}_k$ is a 1-fooling set for $h_n(L_{\exists\forall})$ and $\overline{\Pi}$ (Note that $\mathcal{A}_k$'s are pairwise disjoint). The fact $\forall \gamma \in \mathcal{A} : h_n(L_{\exists\forall})(\gamma) = 1$ is obvious. Since $\mathcal{A}_k$'s are 1-fooling sets it remains to show that $i \neq j$ implies $h_n(L_{\exists\forall})(\overline{\Pi}^{-1}(\gamma_{\overline{\Pi},L}, \delta_{\overline{\Pi},R})) = 0$ for every $\gamma \in \mathcal{A}_i$ and every $\delta \in \mathcal{A}_j$. Let

$$\gamma = 0^{m(i-1)}\alpha_i 0^{m(m-i)} 1^{m(i-1)} \alpha_i 1^{m(m-i)} \in A_i$$

for some $\alpha_i \in \{0, 1^m\} - \{0^m, 1^m\}$, and let

$$\delta = 0^{m(j-1)}\beta_j 0^{m(m-j)} 1^{m(j-1)} \beta_j 1^{m(m-j)} \in A_j$$

for some $\beta_j \in \{0,1\}^m - \{0^m, 1^m\}$. Since $\{\alpha_j, \beta_j\} \cap \{0^m, 1^m\} = \emptyset$ it can be easily seen that

$$\overline{\Pi}^{-1}(\gamma_{\overline{\Pi},L}, \delta_{\overline{\Pi},R}) = 0^{m(i-1)}\alpha_i 0^{m(m-i)} 1^{m(j-1)} \beta_j 1^{m(m-j)} \notin L_{\exists\forall}.$$

Since $|\mathcal{A}| = m|\mathcal{A}_1| = m(2^m - 2)$ we get

$$\mathrm{ncc}(h_n(L_{\exists\forall}), \overline{\Pi}) \geq \lceil \log_2(m(2^m - 2)) \rceil.$$

To see the upper bound it is sufficient to consider a one-way nondeterministic protocol $D_n = \langle \overline{\Pi}, \Phi \rangle$ working as follows. For every input $\alpha = \alpha_1\alpha_2 \ldots \alpha_m\beta_1\beta_2 \ldots \beta_m, \alpha_i, \beta_i \in \{0,1\}^m$, the first computer guesses an integer $k \in \{1, \ldots, m\}$ and submits the message $\mathrm{BIN}^{-1}_{\lceil \log_2 m \rceil}(k)\alpha_k$ of the length $m + \lceil \log_2 m \rceil$. If $\alpha_k = \beta_k$, then the second computer accepts α, elsewhere it rejects α.

(ii) To show that $\mathrm{cc}(h_{2n}(L_{\exists\forall}), \overline{\Pi}) \geq n$ it is sufficient to prove $\mathrm{cc}((h_{2n}(L_{\exists\forall}))^{\mathtt{C}}, \overline{\Pi}) \geq n$. Note that $(L_{\exists\forall})^{\mathtt{C}} = \{x_1x_2 \ldots x_m y_1 y_2 \ldots y_m \mid m \in \mathbb{N}, \forall i \in \{1, \ldots, m\} : x_i \neq y_i; x_i, y_i \in \{0,1\}^m\}$. To do it we use the method rankfix. We define the function

$$g_{2n}(w_1, \ldots, w_m, y_1, \ldots, y_m) =$$

$$\sum_{x_1 \neq w_1} \sum_{x_2 \neq w_2} \cdots \sum_{x_n \neq w_n} h_{2n}(L^{\mathtt{C}}_{\exists\forall})(x_1, \ldots x_n, y_1, \ldots y_n) \mod 2,$$

where $w_i, y_i, x_i \in \{0,1\}^m$ for every $i \in \{1,\ldots,m\}$. We claim that $g_{2n}(\alpha) = h_{2n}(\mathrm{Eq})(\alpha)$, i.e., that $g(w_1,\ldots,w_m,y_1,\ldots,y_m) = 1$ iff $\forall i \in \{1,\ldots,m\}$: $w_i = y_i$. Let us prove it. First consider inputs $\alpha = w_1 \ldots w_m w_1 \ldots w_m$ for any $w_i \in \{0,1\}^m$.

$$g_{2n}(\alpha) = g_{2n}(w_1,\ldots,w_m,w_1,\ldots,w_m) = (2^m-1)^m \bmod 2 = 1$$

because there are exactly $2^m - 1$ words from $\{0,1\}^m$ different from w_i and $h_{2n}(L^{\mathrm{C}}_{\exists\forall})(\beta, w_1,\ldots,w_m) = 1$ for all $\beta \in \{x_1x_2\ldots x_m \in \{0,1\}^n \mid \forall i \in \{1,\ldots,m\} : x_i \in \{0,1\}^m, x_i \neq w_i\}$.

Now we consider inputs $\gamma \in \{w_1w_2\ldots w_m y_1 y_2 \ldots y_m \mid w_i, y_i \in \{0,1\}^m$ for every $i \in \{1,\ldots,m\}$, and $\exists j$ such that $w_j \neq y_j\}$.

Let $\gamma = u_1u_2\ldots u_m v_1 v_2 \ldots v_m$, where $u_i, v_i \in \{0,1\}^m$, $u_k \neq v_k$ for every $k \in \{j_1,\ldots,j_r\} \subseteq \{1,\ldots,m\}$, and $u_p = v_p$ for $\forall p \in \{1,\ldots,m\} - \{j_1,\ldots,j_r\}$. Then

$$g_{2n}(\gamma) = (2^m-2)^r(2^m-1)^{m-r} \bmod 2 = 0.$$

Thus, the matrix $\mathrm{M}(g_{2n}, \overline{\Pi})$ is the diagonal matrix of size $2^n \times 2^n$ and $\mathrm{rank}(M(g_{2n}, \overline{\Pi})) = 2^n$. Following the definition of g_{2n} we see that $\mathrm{M}(g_{2n}, \overline{\Pi})$ is obtained from $\mathrm{M}(h_{2n}(L^{\mathrm{C}}_{\exists\forall}), \overline{\Pi})$ by adding rows [each row of $\mathrm{M}(g_{2n}, \overline{\Pi})$ is a sum mod 2 of rows of $\mathrm{M}(h_{2n}(L^{\mathrm{C}}_{\exists\forall}), \overline{\Pi}]$.

So, $\mathrm{rank}(M(h_{2n}(L_{\exists\forall}), \overline{\Pi})) = \mathrm{rank}(M(h_{2n}(L^{\mathrm{C}}_{\exists\forall}), \overline{\Pi})) \geq \mathrm{rank}(M(g_{2n}), \overline{\Pi}) = 2^n$.

(iii) First, we describe the Las Vegas protocol $D_{2n} = \langle \overline{\Pi}, \Phi \rangle$ computing $h_{2n}(L_{\exists\forall})$, and then we analyse its communication complexity.

Let $S = \{p_1, p_2, \ldots, p_r \mid \forall j = 1,\ldots,r : p_j \leq m$ and p_i is a prime$\}$. (Note that $r \sim \frac{m}{\log m}$). We describe the work of D_n on an input $\alpha = \alpha_1\alpha_2\ldots\alpha_m\beta_1\beta_2\ldots,\beta_m, \alpha_i, \beta_i \in \{0,1\}^m$ for every $i \in \{1,\ldots,m\}$. For each $i = 1,\ldots,m$, D_{2n} uses the following procedure to check whether $x_i = y_i$ or not.

Step 1 $\lceil \log_2 m \rceil = d$ numbers $s_1,\ldots,s_d$ from S are randomly chosen

Step 2 The first computer submits the message "$\mathrm{BIN}_d^{-1}(\mathrm{BIN}(\alpha_i) \bmod p_{s_1})$ $\mathrm{BIN}_d^{-1}(\mathrm{BIN}(\alpha_i) \bmod p_{s_2})\ldots\mathrm{BIN}_d^{-1}(\alpha_i) \bmod p_{s_d})$"

Step 3 If (for $\forall j \in \{1,\ldots,m\}$: $\mathrm{BIN}(\alpha_i) \bmod p_{s_j} = \mathrm{BIN}(\beta_i) \bmod p_{s_j}$), then the second computers submits the message "1". After that the first computer submits the whole α_i to the second computer. If $\alpha_i = \beta_i$ the second computer accepts α (otherwise the second computer submits "0" and the protocol continues to check whether $\alpha_{i+1} = \beta_{i+1}$).

Step 4 If $\exists j \in \{1,\ldots,m\}$ such that $\mathrm{BIN}_d^{-1}(\alpha_i) \bmod p_{s_j} \neq \mathrm{BIN}_d^{-1}(\beta_i) \bmod p_{s_j}$, then the second computer sends "0" to the first one and D_{2n} continues to check whether $\alpha_{i+1} = \beta_{i+1}$

Note, that the numbers $s_1, \ldots, s_d$ randomly chosen from S may be used for the comparison of α_i and β_i for every $i \in \{1, 2, \ldots, m\}$, i.e. we do not need to choose new d numbers for every comparison. So, D_{2n} can be viewed as a sequence of protocols, such that each of these protocols has $\lceil \log_2 m \rceil = d$ prime numbers from S as a random input (d^2 bits) and each one computes $h_{2n}(L_{\exists\forall})$.

To analyse the communication complexity of D_{2n} we use the following fact known from number theory and already proved in Example 2.5.5.6:

Fact. For every two numbers $a, b \in \{0, 1, \ldots, 2^m - 1\}$, $a \neq b$ implies $| \{z \in S \mid a \bmod z \neq b \bmod z\} | \geq \frac{|S|}{2}$.

We compute the communication complexity of the rounds of D_{2n} connected with the comparison of α_i and β_i for an $i \in \{1, \ldots, m\}$. The step 2 takes always d^2 bits. If $\alpha_i \neq \beta_i$, then the fact stated above implies that the probability of the use of step 3 is 2^{-d} (more precisely, 2^{-d} is the ratio of the number of deterministic protocols using step 3 to the number of all protocols in the sequence of protocols D_{2n}). Hence, if $\alpha_i \neq \beta_i$ then the average (expected) number of bits sent is steps 2 and 3 is

$$d^2 + 2^{-d}m + 2 \leq d^2 + 3.$$

If $\alpha_i = \beta_i$, then the steps 2 and 3 are performed and the number of exchanged bits is $d^2 + m + 1$. Note that the equality case $\alpha_i = \beta_i$ occurs only once during the execution of the communication algorithm because after knowing $\alpha_i = \beta_i$ the second computer halts and accepts.

Thus, for every $\alpha \in N^0(h_{2n}(L_{\exists\forall}))$, the average (over all protocols in the sequence D_{2n}) length of the communication is

$$m(d^2 + 3) = m(\lceil \log_2 m \rceil^2 + 3).$$

For every $\gamma \in N^1(h_{2n}(L_{\exists\forall}))$ the average length of the communication is at most (the worst-case input is the input with $\alpha_m = \beta_m$ and $\alpha_i \neq \beta_i$ for all $i = 1, \ldots, m-1$)

$$(m-1)(d^2 + 3) + d^2 + m + 1 \leq m(d^2 + 4) \leq m(\lceil \log_2 m \rceil^2 + 4).$$

□

2.5.6 Randomness Versus Nondeterminism and Determinism

In the previous section we gave some examples showing the power of randomized protocols for one fixed partition. The aim of this section is to compare the randomized communication complexity as minimum over all balanced partitions with communication complexity and nondeterministic communication

complexity. Since we consider the private nondeterminism and public randomness we define the probabilistic complexity measure in a way forcing the exchange of random bits. In this way we obtain a convenient base for the comparison of nondeterministic protocols and randomized ones. Note that this additional $+m$ factor in the following definition of randomized communication complexity is not of a crucial importance because m can be always bounded by $O(\log_2(\text{the number of input variables}))$ (see Exercise 2.5.7.18 for more details).

Definition 2.5.6.1 *Let f be a Boolean function defined on a set X of Boolean variables. The* **Las Vegas communication complexity of f** *is*

$$\mathbf{LVcc}(f) = \min\{m\text{-LVcc}(f) + m \mid m \in \mathbb{N}\},$$

and the **one-sided-error Monte Carlo communication complexity of f** *is*

$$\mathbf{1MCcc}(f) = \min\{m\text{-1MCcc}(f) + m \mid m \in \mathbb{N}\}.$$

The **two-sided-error Monte Carlo communication complexity of f** *is*

$$\mathbf{2MCcc}(f) = \min\{m\text{-2MCcc}(f) + m \mid m \in \mathbb{N}\}.$$

We start with some simple observable relations.

Theorem 2.5.6.2 *For every Boolean function f*

(i) $\text{ncc}(f) \leq \text{1MCcc}(f) \leq \text{cc}(f)$, *and*

(ii) $\text{ncc}(f) \leq \text{LVcc}(f) \leq \text{cc}(f) \leq (\text{LVcc}(f) + 2)^2$.

Proof. (i) is a direct consequence of Theorem 2.5.5.7. $\text{ncc}(f) \leq \text{LVcc}(f)$ follows from the facts that every Las Vegas randomized protocol is also a nondeterministic protocol, and that the communication complexity of the nondeterministic protocol on an input α is the length of the shortest communication while the complexity of Las Vegas randomized protocol on an input α is taken as the average of the length of all computations on α. $\text{LVcc}(f) \leq \text{cc}(f)$ holds because for every $m \in \mathbb{N}$ and every deterministic protocol D one can construct an Las Vegas randomized m-protocol consiting of 2^m copies of D. The last unequality $\text{cc}(f) \leq (\text{LVcc}(f) + 2)^2$ follows from Theorem 2.5.4.6 [$\text{cc}(f) \leq \text{ncc}(f)(\text{ncc}(f^{\mathbb{C}}) + 2)$] and from the fact that Las Vegas communication complexity is closed under complementation [i.e., $\text{ncc}(f^{\mathbb{C}}) \leq \text{LVcc}(f)$]. □

We show now an exponential gap between communication complexity and one-sided error Monte Carlo communication complexity. To do it we consider the language

$$L_{\text{shift}} = \{x_1x_2\dots x_mz_1\dots z_d \mid x_i \in \{0,1\} \text{ for } i = 1,\dots,m, z_j \in \{0,1\} \text{ for } j = 1,\dots,d,\ m \in \mathbb{N}, m \bmod 4 = 0, d = \lceil \log_2 m \rceil, \text{ and for } k = \text{BIN}(z_1\dots z_d): x_1x_2\dots x_{m/4} = x_k x_{(k+1) \bmod m}\dots x_{(k+m/4-1) \bmod m}\}.$$

Theorem 2.5.6.3 *Let $n = m + \lceil \log_2 m \rceil$ for some $m = 4b, b \in \mathbb{N}$. Let $d = \lceil \log_2 m \rceil$. Then*

(i) $\text{ncc}(h_n(L^{\mathbf{C}}_{shift})) \leq \lceil \log_2 m \rceil$ *and* $2d\text{-1MCcc}(h_n(L^{\mathbf{C}}_{shift})) \leq 2\lceil \log_2 m \rceil$,

(ii) $\text{cc}(h_n(L^{\mathbf{C}}_{shift})) = \Omega(n)$.

Proof.

(i) Because of Theorem 2.5.6.2 it is sufficient to show $2d\text{-1MCcc}(h_n(L^{\mathbf{C}}_{shift})) \leq 2d$. Let $X = \{x_1,\dots,x_m,z_1,\dots,z_d\}$ be the set of input variables of $h_n(L^{\mathbf{C}}_{shift})$. To compute $h_n(L^{\mathbf{C}}_{shift})$ we take a Monte Carlo $2d$-randomized protocol $D_n = \langle \Pi, \Phi \rangle$, where Π is choosen in a way such that $\Pi_L \supseteq \{z_1,\dots,z_d,x_1,\dots,x_{m/4}\}$. We describe the work of D_n on an input $\alpha_1,\dots,\alpha_m\beta_1\dots\beta_d$. Let $k = \text{BIN}(\beta_1\dots\beta_d)$, and for every $r \in \mathbb{N}$, $S_r = \{p \in N \mid p \leq r \text{ and p is prime }\} = \{p_1,\dots,p_{a_r}\}$. The first computer knows k and $\gamma_1 = \alpha_1\dots\alpha_{m/4-1}$, and $z \geq 0$ bits from $\gamma_2 = \alpha_k\alpha_{(k+1) \bmod m}\dots\alpha_{k+m/4-1 \bmod m}$. If the knowledge about this z bits implies $\gamma_1 \neq \gamma_2$ the protocol accepts the input. If not, then the protocol has to check whether $\beta_1 \neq \beta_2$ for $\beta_1,\beta_2 \in \{0,1\}^{m-z}$ such that the first computer knows β_1 and nothing about β_2, and the second computer knows β_2 and nothing about β_1. This can be done by using the random input $\delta \in \{0,1\}^{\lceil \log_2(a_{m-z}) \rceil}$. The first computer sends $\text{BIN}_d^{-1}(\text{BIN}(\beta_1) \bmod p_{\text{BIN}(\delta)})$ to the second computer. If $\text{BIN}(\beta_1) \bmod p_{\text{BIN}(\delta)} \neq \text{BIN}(\beta_2) \bmod p_{\text{BIN}(\delta)}$, then the second computer accepts α, otherwise it rejects α. Following the fact $\mid \{p \in S_{m-z} \mid b_1 \bmod p \neq b_2 \bmod p\} \mid \leq \mid S_{m-z} \mid /2$ for any $b_1 \neq b_2$, $b_1, b_2 \in \{0,1\}^{m-z}$, we see that the protocol described above is a one-sided-error Monte Carlo $\log_2(a_{m-z})$-protocol (note that $a_{m-z} \leq (m-z)^2$ which implies $\log_2(a_{m-z}) \leq 2d$).

(ii) Because communication complexity classes are closed under complement, it is sufficient to prove that $\text{cc}(h_n(L_{shift})) \geq n/32$. Let Π be any balanced partition of $X = \{x_1,\dots,x_m,z_1,\dots z_d\}$. Without loss of generality we may assume that $\mid \Pi_L \cap \{x_1,x_2,\dots,x_{m/4}\} \mid \geq m/8$ (if not, the same proof can be made from the point of view of the second computer). Let $\Pi_L \cap \{x_1,\dots,x_{m/4}\} = \{x_{1p},\dots,x_{ip}\}$, $p \geq \frac{m}{8}$. Let, for $j = m/4,\dots,3/4m-1, S_j = \{(i_1+j) \bmod m, \dots, (i_p+j) \bmod m\}$, and $S_j(x) = \{x_r \in X \mid r \in S_j\}$. Let $s_j = |S_j(x) \cap \Pi_R|$ for every $j \in \{m/4+1,\dots,(3/4)m\}$. Now, we want to prove that there exists $u \in \{m/4+1, m/4+2,\dots,(3/4)m\}$ such that $s_u \geq n/32$.

For every $k \in \{1,\dots,p\}$, let

$$R_k = \{(i_k + m/4 + 1), (i_k + m/4 + 1), \ldots, (i_k + 3/4m)\},$$

$$R_k(x) = \{x_b \in X \mid b \in R_k\}, \text{ and } r_k = |R_k(x) \cap \Pi_R|.$$

Obviously, $\bigcup_{k=1}^{p} R_k(x) = \bigcup_{j=m/4+1}^{3m/4} S_j(x)$ and $\sum_{j=m/4+1}^{3m/4} s_j = \sum_{k=1}^{p} r_k$.

(One can see the situation as a matrix whose p rows are $R_k(x)$'s and whose $m/2$ columns are $S_j(x)$'s). Since $\Pi \in \text{Bal}(X)$, $r_k \geq m/2 - ((m \cdot \lceil \log_2 m \rceil)/2 - p)$ for every $k \in \{1, \ldots, p\}$ (otherwise Π_L will contain more than the half of input variables).

Thus,

$$\sum_{j=m/4+1}^{3m/4} s_j = \sum_{k=1}^{p} r_k \geq \sum_{k=1}^{p} [m/2 - ((m\lceil \log_2 m \rceil)/2 - p)] \geq p(p - (\log_2 m)/2) \geq$$

$$p(m/8 - (\log_2 m)/2) \geq p(2n - 5 \log m)/8 \geq pn/8 \geq mn/64$$

for sufficiently large n.

So, among the $m/2$ numbers s_j $(j = m/4 + 1, \ldots, 3m/4)$ there exists a number

$$s_u \geq \sum_{j=m/4+1}^{3m/4} s_j/(m/2) \geq n/32.$$

This means that if the input assignment $\gamma : \{z_1, \ldots, z_d\} \to \{0, 1\}$ is chosen so that $\text{BIN}(\gamma) = u$, then the protocol has to check $x_b = x_{b+u}$, where $x_b \in \Pi_L, x_{b+u} \in \Pi_R$, for $n/32$ different variables x_b. Using either the fooling set method or the rank method the proof can be completed in the standard way. □

We see that the lower bound proof of (ii) of Theorem 2.5.6.3 is based on "shifting", which means that:

1. We change the original computing problem P comparing values $(x_i = x_j)$ of some variables (and so having large communication complexity according to a fixed partition partitioning the compared pairs of variables) to a "relativized" version of P, where the choice of pairs of variables that must be compared depends on the shift given by the values of some new variables.

2. For each balanced partition Π of variables of the relativized problem, there exists a shift (a choice of values for the new variables) such that Π divides a lot of pairs of variables that must be compared according to the given shift.

The above shifting idea can be used also to extend Lemma 2.5.5.9 to show that Las Vegas randomized protocols can be more powerful than deterministic ones (LVcc(f) $\ll$ cc(f) for some f shifting the problem $h_n(L_{\exists\forall})$). Note that the gap between LVcc(f) and cc(f) can be at most quadratic because of (ii) of Theorem 2.5.6.2.

Finally, we deal with the relation between the following powerful protocols: nondeterministic ones and (two-sided-error) Monte Carlo ones. We show an exponential gap between nondeterministic communication complexity and Monte Carlo communication complexity.

Theorem 2.5.6.4 *Let $n = m + \lceil \log_2 m \rceil$ for some $m = 4b, b \in \mathbb{N}$. Let $d = \lceil \log_2 m \rceil$. Then,*

(i) $\mathrm{ncc}(h_n(L_{shift})) = \Omega(n)$, *and*

(ii) $d\text{-2MCcc}(h_n(L_{shift}) = O(\log_2 n)$.

Proof.

(i) In the proof of (ii) of Theorem 2.5.6.3 we have proved (for sufficiently large n) $\mathrm{cc}(h_n(L_{shift})) \geq n/32$ by constructing a 1-fooling set of cardinality at least $2^{n/32}$ for every balanced partition Π. Thus, we get immediately $\mathrm{ncc}(h_n(L_{shift})) \geq \frac{n}{32}$ for sufficiently large n.

(ii) Let $X = \{x_1, \ldots, x_m, z_1, \ldots, z_d\}$ be the input variables of $h_n(L_{shift})$. Obviously, it is sufficient to prove $d\text{-MCcc}(h_n(L_{shift}), \Pi) \leq \log_2 n$ for some $\Pi \in \mathrm{Bal}(x)$ and all sufficiently large n.

We choose a Π such that $\{z_1, \ldots, z_d, x_1, \ldots, x_{m/4}\} \subseteq \Pi_L$. Then , for any input assignment $\gamma : \{z_1, \ldots, z_d\} \to \{0,1\}$, the first computer knows the shift $k = \mathrm{BIN}(\gamma)$, and the task is only to check whether $x_1 \ldots x_{m/4} = x_k x_{(k+1) \bmod m} \cdots x_{(k+m/4-1) \bmod m}$ or not. Anyway, independently of which variables from $\{x_k, x_{(k+1) \bmod m}, \ldots, x_{(k+m/4-1) \bmod m}\}$ are in Π_L (known for the first computer), the task is only to check whether $\alpha = \beta$ for $\alpha, \beta \in \{0,1\}^r$, $r \leq m/4 - 1$. But this can be done by using the $\log_2 r$-Monte Carlo protocol $\overline{D}_1$ from the proof of Lemma 2.5.5.8 (see also Example 2.5.5.6 working within $\log_2 n$ communication complexity).

□

We note that Theorem 2.5.6.4 shows not only that Monte Carlo can be more powerful that nondeterminism, but also that Monte Carlo can be much more powerful than Las Vegas and one-sided-error Monte Carlo.

2.5.7 Exercises

Exercise 2.5.7.1 *Let n, m be two positive integers, $m \leq n/2$. Let $f_1, f_2 \in B_2^n$ be two Boolean functions defined on a set X of input variables. Let $\Pi \in \mathrm{Abal}(X)$. Prove that* $\mathrm{ncc}(f_1, \Pi) \leq m$ *and* $\mathrm{ncc}(f_2, \Pi) \leq m$ *imply* $\mathrm{ncc}(f_1 \vee f_2, \Pi) \leq m + 1$.

Exercise 2.5.7.2 *Prove that there exist two languages* L_1 *and* L_2 *such that* $\mathrm{ncc}(h_n(L_1)) \leq 1$, $\mathrm{ncc}(h_n(L_2)) \leq 1$, *and* $\mathrm{ncc}(h_n(L_1 \cup L_2)) = \Omega(n)$.

Exercise 2.5.7.3 *Let* $f_1, f_2 \in B_2^n$ *for some positive integer* n, *and let* Π *be a balanced partition of their input variables. Prove that* $\mathrm{ncc}(f_1 \wedge f_2, \Pi) \leq \mathrm{ncc}(f_1, \Pi) + \mathrm{ncc}(f_2, \Pi) + 1$ *is always true.*

Exercise 2.5.7.4 * *Prove that there exist two languages* L_1 *and* L_2 *such that* $\mathrm{ncc}(h_n(L_1)) \leq 1$, $\mathrm{ncc}(h_n(L_2)) \leq 1$, *and* $\mathrm{ncc}(h_n(L_1 \cap L_2)) = \Omega(n)$.

Exercise 2.5.7.5 * *Find some further languages* L *such that* $\mathrm{cc}(h_n(L))$ *is large and* $\mathrm{ncc}(h_n(L))$ *is small.*

Exercise 2.5.7.6 * *Find some further languages* L *with the property*

$$\mathrm{cc}(h_n(L_n), \overline{\Pi}) = \Omega(\mathrm{ncc}(h_n(L), \overline{\Pi})\mathrm{ncc}(h_n(L^{\mathbf{C}}, \overline{\Pi}))/(\log n)^i)$$

for some $i \in \mathbb{N}$.

Exercise 2.5.7.7 *Define Las Vegas communication complexity classes and prove that they are closed under complement.*

Exercise 2.5.7.8 *Prove that Las Vegas communication complexity language (function) classes are not closed under union and intersection (disjunction and conjunction). Note that it is possible to prove that there exist languages* L_1 *and* L_2 *such that* $\mathrm{LVcc}(h_n(L_1)) \leq 1$, $\mathrm{LVcc}(h_n(L_2)) \leq 1$ *and* $\mathrm{LVcc}(h_n(L_1 \cup L_2)) = \Omega(n)$.

Exercise 2.5.7.9 *Define one-sided-error Monte Carlo communication complexity classes and two-sided-error Monte Carlo communication complexity classes. Prove that the first ones are not closed under complementation while the second ones are.*

Exercise 2.5.7.10 *Prove that there exist languages* L_1 *and* L_2 *such that* $\mathrm{1MCcc}(h_n(L_1)) \leq 1$, $\mathrm{1MCcc}(h_n(L_2)) \leq 1$, *and* $\mathrm{1MCcc}(h_n(L_1 \cup L_2)) = \Omega(n)$.

Exercise 2.5.7.11 * *Define a shifting version* $L_{sh\exists\forall}$ *of the language* $L_{\exists\forall}$ *such that* $\mathrm{LVcc}(h_n(L_{sh\exists\forall})) = O(\sqrt{n} \cdot (\log_2 n)^2)$ *and* $\mathrm{cc}(h_n(L_{sh\exists\forall})) = \Omega(n)$.

Exercise 2.5.7.12 * *Prove or disprove:*

"There exists a language L *such that* $\mathrm{1MCcc}(h_n(L))$ *grows essentially (exponentially, if possible) more quickly than* $\mathrm{ncc}(h_n(L))$*."*

Exercise 2.5.7.13 *Improve the nondeterministic protocol* D_n *of Example 2.5.2.4 in such a way that* $\mathrm{ncc}_1(D_n) \leq 2 \cdot \log_2 n + 4$.

Exercise 2.5.7.14 *Find a language L such that for any n, $r \in \mathbb{N}$, $0 \leq r \leq \log_2 n$,*

$$\lceil \frac{n}{2^r} \rceil \leq r\text{-pncc}(h_n(L)) \leq \lceil \frac{n}{2^r} \rceil + r.$$

Exercise 2.5.7.15 * *Define, for all positive integers m and k, and for every Boolean function f, the m-public nondeterministic k-rounds communication complexity of f, and denote it by $m\text{-pncc}_k(f)$. We known that the numbers of rounds is not essential for nondeterministic protocols. This changes if one fixes the degree of nondeterminism. Find, for every $k \in \mathbb{N}$, a language L_k such that*

(i) $\text{cc}_k(h_n(L_k)) \leq k \cdot \log_2 n$, *and*

(ii) for every c, $0 < c < 1$, $(c \cdot \log_2 n)\text{-pncc}_{k-1}(h_n(L_k)) = \Omega(n^{1-c})$.

Exercise 2.5.7.16 *Prove that, for every positive integers n and $r \leq n/2$, there exists a Boolean function $f \in B_2^n$ such that*

$$r\text{-pncc}(f) \geq \lfloor n/2 \rfloor - r.$$

Exercise 2.5.7.17 *Prove that there is a language L such that*

(i) $\text{cc}(h_n(L)) = n$, *and*

(ii) $\text{ncc}(h_n(L)) \leq \lceil \log_2 n \rceil + 1$.

Exercise 2.5.7.18 * *Prove that there exists a positive integer c such that, for every language L and every function $g : \mathbb{N} \to \mathbb{N}$,*

(i) $g(n)\text{-1MCcc}(h_n(L)) \leq (c \cdot \log_2 n)\text{-1MC}(h_n(L))$,

(ii) $g(n)\text{-2MCcc}(h_n(L)) \leq (c \cdot \log_2 n)\text{-2MC}(h_n(L))$, *and*

(iii) $g(n)\text{-LVcc}(h_n(L)) \leq (c \cdot \log_2 n)\text{-LVcc}(h_n(L))$.

Exercise 2.5.7.19 * *Prove that, for every Boolean function f, every positive integer m, and every partition Π of the set of the input variables of f*

$$\text{ncc}(f, \Pi) \leq m\text{-LVcc}(f, \Pi).$$

Exercise 2.5.7.20 * *Improve the result of Lemma 2.5.5.9 by showing*

$$O(\log_2 m)\text{-LVcc}(h_{2n}(L_{\exists\forall}), \overline{\Pi}) \leq m(\lceil \log_2 m \rceil^2 + 4).$$

Exercise 2.5.7.21 * *Find a language L such that*

(i) $\mathrm{ncc}(h_n(L)) = O(\sqrt{n})$,

(ii) $\mathrm{cc}(h_n(L)) = \Omega(n)$, *and*

(iii) $\mathrm{LVcc}(h_n(L)) = O(\sqrt{n} \cdot (\log_2 n)^2)$.

Exercise 2.5.7.22 ** *Consider the language* $DISJ = \{x_1x_2 \dots x_ny_1y_2 \dots y_n \mid x_i, y_i \in \{0,1\}$ *for* $i \in \{1, \dots, n\}$, $n \in \mathbb{N}$, *and* $\sum_{i=1}^{n} x_iy_i = 0\}$. *Prove that*

(i) $\mathrm{ncc}(h_{2n}(DISJ^{\complement}, \overline{\Pi})) = O(\log_2 n)$, *and*

(ii) $2\mathrm{MCcc}(h_{2n}(DISJ^{\complement})) = \Omega(\sqrt{n})$.

Exercise 2.5.7.23 ** *Find, for any function* $s : \mathbb{N} \to \mathbb{N}$, *a language* L_s *such that*

(i) $s(n)\text{-pncc}(h_{2n}(L_s)) = O(s(n))$, *and*

(ii) $o(s(n)/\log_2 n)\text{-pncc}(h_{2n}(L_s)) = \Omega(n/\log_2(n/s(n)))$.

This means that a loss of a logarithmic factor of advice bits has to be payed for with an almost maximal increase in communication, even if the original number of advice bits is large (i.e., superlogarithmic).

Exercise 2.5.7.24 * *Define one-way Las Vegas communication complexity and prove that it has a linear relation to one-way communication complexity. Find a language* L *such that the one-way Las Vegas communication complexity of* $h_n(L)$ *is equal to* $\mathrm{cc}_1(h_n(L))/2$.

2.5.8 Research problems

Problem 2.5.8.1 * *Find, for some functions* $g(n) : \mathbb{N} \to \mathbb{N}$, $g(n) > \log_2 n$, *a concrete language* L_g *such that* $L_g \in \mathrm{NCOMM}(g(n)) - \mathrm{NCOMM}(g(n) - 1)$. *Note that* $\mathrm{NCOMM}(g(n)) - \mathrm{NCOMM}(g(n) - 1) \neq 0$ *has been proved in Theorem 2.5.4.5, but for* $g(n) > \log_2 n$ *the proof is existential. Probably, this task is too hard, and so results of the kind* $L_g \in \mathrm{NCOMM}(g(n)) - \mathrm{NCOMM}(g(n)/2)$ *are of interest too.*

Problem 2.5.8.2 *Prove that there is a language* L *such that* $\mathrm{cc}(h_n(L)) = \Omega(\mathrm{ncc}(h_n(L))\mathrm{ncc}(h_n(L^{\complement})))$ *or improve the upper bound given in Theorem 2.5.4.7.*

Problem 2.5.8.3 * *Find, for every* $k \in \mathbb{N}$, *a language* L_k *such that*

(i) $\mathrm{cc}_k(h_n(L_k)) = O(\log_2 n)$, *and*

(ii) $s(n)\text{-pncc}_{k-1}(h_n(L_k)) = \Omega(n^a)$

for a real number $a > 0$ and a superlogarithmic function $s : \mathbb{N} \to \mathbb{N}$ (or prove that such a result is impossible). Note that this assertion extends the claim of Exercise 2.5.7.15, where the importance of the number of communication rounds for a small number of random bits is given.

Problem 2.5.8.4 ** *Let $s : \mathbb{N} \to \mathbb{N}$ be a superlogarithmic function. Does there exists a language L_s such that $s(n)$-pncc$(h_n(L_s))$ is small ($O(s(n))$ for instance) and $o(s(n))$-pncc$(h_n(L_s))$ is large (almost as cc$(h_n(L_s))$? This would strengthen the assertion of Exercise 2.5.7.23 by showing a very strong threshold on the number of advice bits for some computing problems.*

Problem 2.5.8.5 ** *Consider unbounded-error randomized protocols (communication complexity). Is there possible to restrict the number of random bits for these probabilistic protocols in a similar way as we did it for Las Vegas and Monte Carlo protocols in Exercise 2.5.7.18 ?*

2.6 An Improved Model of Communication Protocols

2.6.1 Introduction

The communication complexity measure introduced and studied in the previous sections has the following two drawbacks:

(1) There are computing problems requiring a lot of communication to be solved in parallel, but our communication complexity is small (i.e., our communication complexity is unable to provide a lower bound close to upper bounds given by designed parallel algorithms).

(2) It is mostly very hard to prove nontrivial lower bounds on communication complexity of specific problems because communication complexity is a minimum of communication complexities according to all (almost) balanced partitions.

While the drawback (2) is obvious, the drawback (1) needs a more careful explanation. Let us consider a problem consisting of a constant number of subproblems defined over disjoint sets of input variables. Let some of these subproblems require linear (maximal) communication complexity, and let small (constant, for example) communication complexity suffice to obtain the solution of the problem in the case that the solutions of the subproblems are known.

Now, if we take a partition of input bits that gives the input variables of some subproblems to the first computer and the input bits of additional subproblems to the second computer, the problem can be solved with small (constant) communication complexity. On the other hand the complexity of parallel algorithms solving the problem is large because the subproblems are difficult (they have a large communication complexity). An extreme example of the drawback (1)

is a problem $\{f_1, f_2\}$, where $cc(f_1) = cc(f_2) = n$, f_1 is defined over the set of input variables $X = \{x_1, \ldots, x_n\}$, f_2 is defined over the set of input variables $Z = \{z_1, \ldots, z_n\}$, and $X \cap Z = \emptyset$. Obviously $cc(\{f_1, f_2\}) = 1$ (choose a partition Π such that $\Pi_L = X$) but the real communication complexity of parallel processing is large.

To partially overcome the drawbacks mentioned above we redefine the communication complexity as follows. Let P be a problem with a set of input variables X, and let Z be a subset of X. We define the communication complexity according to Z as the minimum over all partitions of X which divide Z into two (almost) equal-sided parts (this means that input variables from $X - Z$ are distributed arbitrarily). Then, we define the new s(subset)-communication complexity of P as the maximum (over all $Z \subseteq X$) of communication complexities according to Z.

Let us now look at our s-communication complexity. It is defined in such a way that we may choose some "kernel" Z of the set of input variables and then we can measure the communication complexity as minimum over partitions dividing Z into two (almost) equal-sided parts. Thus, there is no problem which has a small s-communication complexity and simultaneously contains a subproblem with a large s-communication complexity. As we shall show in the next chapters, s-communication complexity provides lower bounds on parallel complexity measures almost always in the same way as communication complexity does. This means that it is suitable to use s-communication complexity because it is always at least as large as communication complexity (the kernel Z can be chosen as the whole set of input variables), i.e., it provides lower bounds on parallel complexity measures that are at least as high as that provided by communication complexity.

Let us now return to the drawback (2). The possibility to choose a suitable $Z \subseteq X$ can also help to make some lower bound proofs easier. The reason for this assertion is that choosing a suitable $Z \subseteq X$ we do not need to deal with some partitions (dividing X in a balanced way) for which the lower bound proof is especially hard. We shall show a few such examples in the following subsections.

This section is organized as follows. Section 2.6.2 contains the formal definition of s-communication complexity, and Section 2.6.3 is devoted to lower bound proofs for s-communication complexity. Communication complexity and s-communication complexity are compared in Section 2.6.4, and some theoretical properties of s-communication complexity are presented in Section 2.6.5.

2.6.2 Definitions

To define s-communication complexity we do not need to change the definition of protocols; we have only to deal with some new sets of partitions.

Definition 2.6.2.1 *Let X be a set of input variables, and let Z be a subset of X. Let Π be a partition of X. We say that Π is* **Z-balanced** *if*

$$|| \Pi_{L,X} \cap Z | - | \Pi_{R,X} \cap Z || \leq 1.$$

We say that Π *is* **almost Z-balanced** *if*

$$|Z|/3 \leq |\Pi_{L,X} \cap Z| \leq 2|Z|/3.$$

Let $\mathbf{Bal_Z(X)} = \{\Pi \mid \Pi$ *is a* Z*-balanced partition of* $X\}$, *and let* $\mathbf{Abal_Z(X)} = \{\Pi \mid \Pi$ *is an almost* Z*-balanced partition of* $X\}$.

Definition 2.6.2.2 *Let* X *be a set of input variables, and let* Y *be a set of output variables. Let* Z *be a subset of* X. *A partition* Π *of* X *and* Y *is* **Z-balanced** *if* $|| \Pi_{L,X} \cap Z | - | \Pi_{R,X} \cap Z || \leq 1$. *A partition* Π *of* X *and* Y *is called* **almost Z-balanced** *if* $|Z|/3 \leq | \Pi_{L,X} \cap Z | \leq 2|Z|/3$. *Let* $\mathbf{Bal_Z(X,Y)} = \{\Pi \mid \Pi$ *is a* Z*-balanced partition of* X *and* $Y\}$, *and let* $\mathbf{Abal_Z(X)} = \{\Pi \mid \Pi$ *is an almost* Z*-balanced partition of* X *and* $Y\}$.

Note that there is no requirement on the partitioning of $X - Z$ and of Y in the above definitions. Now we define the s-communication complexity as informally described in the previous subsection.

Definition 2.6.2.3 *Let* P_n^r *be a computing problem with a set* X *of input variables* ($|X| = n$) *and a set* Y *of output variables* ($|Y| = r$). *Let* Z *be a subset of* X. **The communication complexity of P_n^r according to Z** *is*

$$\mathbf{cc}(P_n^r, \langle Z \rangle) = \min\{\mathrm{cc}(P_n^r, \Pi) \mid \Pi \in \mathrm{Bal}_Z(X,Y)\},$$

and the **a-communication complexity of P_n^r** *is*

$$\mathbf{acc}(P_n^r, \langle Z \rangle) = \min\{\mathrm{cc}(P_n^r, \Pi) \mid \Pi \in \mathrm{Abal}_Z(X,Y)\}.$$

The **one-way communication complexity of P_n^r according to Z** *is*

$$\mathbf{cc_1}(P_n^r, \langle Z \rangle) = \min\{\mathrm{cc}_1(P_n^r, \Pi) \mid \Pi \in \mathrm{Bal}_Z(X,Y)\},$$

and the **one-way a-communication complexity of P_n^r according to Z** *is*

$$\mathbf{acc_1}(P_n^r, \langle Z \rangle) = \min\{\mathrm{cc}_1(P_n^r, \Pi) \mid \Pi \in \mathrm{Abal}_Z(X,Y)\}.$$

We define the **s-communication complexity of P_n^r** *as*

$$\mathbf{scc}(P_n^r) = \max\{\mathrm{cc}(P_n^r), \langle Z \rangle) \mid Z \subseteq X\},$$

the **s-a-communication complexity of P_n^r** *as*

$$\mathbf{sacc}(P_n^r) = \max\{\mathrm{cc}_1(P_n^r, \langle Z \rangle) \mid Z \subseteq X\},$$

the **one-way s-communication complexity of P_n^r** *as*

$$\mathbf{scc_1}(\boldsymbol{P_n^r}) = \max\{\mathrm{acc}_1(P_n^r, \langle Z\rangle) \mid Z \subseteq X\},$$

and the **one-way s-a-communication complexity of** $\boldsymbol{P_n^r}$ *as*

$$\mathbf{sacc_1}(\boldsymbol{P_n^r}) = \max\{\mathrm{acc}_1(P_n^r), \langle Z\rangle) \mid\subseteq X\}.$$

The simplified version of the previous definition for one-output problems is as follows.

Definition 2.6.2.4 *Let f be a Boolean function defined on a set X of Boolean variables. Let Z be a subset of X. The* **communication complexity of f according to Z** *is*

$$\mathbf{cc}(\boldsymbol{f}, \langle \boldsymbol{Z}\rangle) = \min\{\mathrm{cc}(f, \Pi) \mid \Pi \in \mathrm{Bal}_Z(X)\},$$

and the **a-communication complexity** *of f is*

$$\mathbf{acc}(\boldsymbol{f}, \langle \boldsymbol{Z}\rangle) = \min\{\mathrm{acc}(f, \Pi) \mid \Pi \in \mathrm{Abal}_Z(X)\}.$$

The **one-way communication complexity of f according to Z** *is*

$$\mathbf{cc_1}(\boldsymbol{f}, \langle \boldsymbol{Z}\rangle) = \min\{\mathrm{cc}_1(f, \Pi) \mid \Pi \in \mathrm{Bal}_Z(X)\},$$

and the **one-way a-communication complexity of f according to Z** *is*

$$\mathbf{acc_1}(\boldsymbol{f}, \langle \boldsymbol{Z}\rangle) = \min\{\mathrm{cc}_1(f, \Pi) \mid \Pi \in \mathrm{Abal}_Z(X)\}.$$

We define the **s-communication complexity of f** *as*

$$\mathbf{scc}(\boldsymbol{f}) = \max\{\mathrm{cc}(f, \langle Z\rangle) \mid Z \subseteq X\},$$

the **s-a-communication complexity of f** *as*

$$\mathbf{sacc}(\boldsymbol{f}) = \max\{\mathrm{acc}(f, \langle Z\rangle) \mid Z \subseteq X\},$$

the **one-way s-communication complexity of f** *as*

$$\mathbf{scc_1}(\boldsymbol{f}) = \max\{\mathrm{cc}_1(f, \langle Z\rangle) \mid Z \subseteq X\},$$

and the **one-way s-a-communication complexity of f** *as*

$$\mathbf{sacc_1}(\boldsymbol{f}) = \max\{\mathrm{acc}_1(f, \langle Z\rangle) \mid Z \subseteq X\}.$$

The following facts follow directly from the definitions stated above.

Observation 2.6.2.5 *For any problem P*

(i) $\mathrm{cc}(P) \leq \mathrm{scc}(P)$,

(ii) $\mathrm{cc}_1(P) \leq \mathrm{scc}_1(P)$, *and*

(iii) $\mathrm{scc}(P) \leq \mathrm{scc}_1(P)$.

We close this subsection by defining the s-communication complexity classes.

Definition 2.6.2.6 *Let n, m be positive integers, $m \leq n/2$. We define*

$$\mathbf{SCOMM_n(m)} = \{f \in B_2^n \mid \mathrm{scc}(f) \leq m\},$$

and

$$\mathbf{SCOMM_n^1(m)} = \{f \in B_2^n \mid \mathrm{scc}_1(f) \leq m\}.$$

Definition 2.6.2.7 *Let g be a function from $\mathbb{N}$ to $\mathbb{N}$. We define*

$$\mathbf{SCOMM(g(n))} = \{L \subseteq \{0,1\}^* \mid \mathrm{scc}(h_n(L)) \leq g(n) \text{ for every } n \in \mathbb{N}\},$$

and

$$\mathbf{SCOMM^1(g(n))} = \{L \subseteq \{0,1\}^* \mid \mathrm{scc}_1(h_n(L)) \leq g(n) \text{ for every } n \in \mathbb{N}\}.$$

2.6.3 Lower Bound Methods

All three lower bound methods developed for communication complexity can be simply transformed to lower bound methods for s-communication complexity. We present these methods without proving their correctness. But the proofs are obvious and we leave them to the readers.

Method subset-fool

Input: A problem P_n^r with a set X of input variables and a set Y of output variables.

Step 1: Choose a suitable $Z \subseteq X$.

Step 2: For every $\Pi \in \mathrm{Bal}_Z(X,Y)$ $[\Pi \in \mathrm{Abal}_Z(X,Y)]$ find a fooling set $\mathcal{A}(P_n^r, \Pi)$.

Step 3: Compute $d = \min\{|\mathcal{A}(P_n^r, \Pi)| \mid \Pi \in \mathrm{Bal}_Z(X,Y)\}$
$[d = \min\{|\mathcal{A}(P_n^r, \Pi)| \mid\mid \Pi \in \mathrm{Abal}_Z(X,Y)\}]$.

Output: "$\mathrm{scc}(P_n^r) \geq \lceil \log_2 d \rceil$" ["$\mathrm{sacc}(P_n^r) \geq \lceil \log_2 d \rceil$"].

Method subset-rank

Input: A Boolean function f defined on a set X of input variables.

Step 1: Choose a suitable $Z \subseteq X$.

Step 2: For every $\Pi \in \mathrm{Bal}_Z(X)$ $[\Pi \in \mathrm{Abal}_Z(X)]$ construct the matrix $M(f, \Pi)$.

Step 3: For some positive integer d, prove that

$$d \leq \min\{\text{rank}(M(f,\Pi)) \mid \Pi \in \text{Bal}_Z(X)\}.$$

$$[d \leq \min\{\text{rank}(M(f,\Pi)) \mid \Pi \in \text{Abal}_Z(X)\}].$$

Output: "$\text{scc}(f) \geq \lceil \log_2 d \rceil$" ["$\text{scc}(f) \geq \lceil \log_2 d \rceil$"]

Method subset-tilling

Input: A Boolean function f defined on a set X of input variables.

Step 1: Choose a suitable $Z \subseteq X$.

Step 2: For each $\Pi \in \text{Bal}_Z(X)$ $[\Pi \in \text{Abal}_Z(X)]$ construct the matrix $M(f,\Pi)$.

Step 3: For some positive integer d, prove that

$$d \leq \min\{\text{Til}(M(f,\Pi)) \mid \Pi \in \text{Bal}_Z(X)\}$$

$$[d \leq \min\{\text{Til}(M(f,\Pi)) \mid \Pi \in \text{Abal}_Z(X)\}.]$$

Output: "$\text{scc}(f) \geq \lceil \log_2 d \rceil - 1$" ["$\text{sacc}(f) \geq \lceil \log_2 d \rceil - 1$"].

Analogously, the methods for proving lower bounds on one-way communication complexity can be transformed to methods providing lower bounds on one-way s-communication complexity.

Method subset-1fool

Input: A problem P_n^r with a set X of input variables and a set Y of output variables.

Step 1: Choose a suitable $Z \subseteq X$.

Step 2: For each $\Pi \in \text{Bal}_Z(X,Y)$ $[\Pi \in \text{Abal}_Z(X,Y)]$ find a one-way fooling set $\mathcal{A}_1(P_n^r,\Pi)$.

Step 3: Compute $d = \min\{|\mathcal{A}_1(P_n^r,\Pi)| \mid \Pi \in \text{Bal}_Z(X,Y)\}$
$[d = \min\{|\mathcal{A}_1(P_n^r,\Pi)| \mid \Pi \in \text{Abal}_Z(X,Y)\}$.

Output: "$\text{scc}_1(P_n^r) \geq \lceil \log_2 d \rceil$" ["$\text{sacc}(P_n^r) \geq \lceil \log_2 d \rceil$"].

Method subset-mrow

Input: A Boolean function f defined on a set X of input variables.

Step 1: Choose a suitable $Z \subseteq X$.

Step 2: For each $\Pi \in \text{Bal}_Z(X)$ $[\Pi \in \text{Abal}_Z(X)]$ construct the matrix $M(f,\Pi)$.

Step 3: Compute $d = \min\{|\text{Row}(M(f,\Pi))| \mid \Pi \in \text{Bal}_Z(X)\}$
$[d = \min\{|\text{ Row}(M(f,\Pi))| \mid \Pi \in \text{Abal}_Z(X)\}$.

Output: "$\text{scc}_1(f) \geq \lceil \log_2 d \rceil$" ["$\text{sacc}(f) \geq \lceil \log_2 d \rceil$"].

Next we present three examples of lower bounds on s-communication complexity of specific Boolean functions. All these examples show that the choice of a suitable subset Z of the set X of input variables can help to simplify the lower bound proof in the comparison to the lower bound proof for $Z = X$. Let us first consider the following language

$$\begin{aligned} L_{choice} = \{&wyuv \in \{0,1\}^* \mid |w| = |y| = |v| = |u| = m, m \in \mathbb{N}, \\ &w = w_1, \ldots, w_m, y = y_1, \ldots, y_m; \\ &\text{if } u = z_1 1 z_2,\ v = r_1 1 r_2,\ \#_1(z_1) = \#_1(r_1), \\ &|z_1| = j - 1, \text{and } |r_1| = i - 1 \text{ for some } i, j \in \mathbb{N}, \text{ then } w_j = y_i\}. \end{aligned}$$

Note that L_{choice} is similar to L_{shift}, because the subwords u and v decide which positions of w and y has to be compared.

Theorem 2.6.3.1 *For every $n = 8k$, $k \in \mathbb{N}$, $\mathrm{scc}(h_n(L_{choice})) \geq n/8$.*

Proof. Let $X_n = W_n \cup Y_n \cup U_n \cup V_n$ be the set of input variables of $h_n(L_{choice})$ for $W_n = \{w_1, \ldots, w_m\}$, $Y_n = \{y_1, \ldots, y_m\}$, $U_n = \{u_1, \ldots u_m\}$, and $V_n = \{v_1, \ldots v_m\} (m = n/4 = 2k)$. We show that $\mathrm{cc}(h_n(L_{choice}), \langle W_n \rangle) \geq k = n/8$. Let Π be a partition from $\mathrm{Bal}_{W_n}(X_n)$. Without loss of generality we assume that $| \Pi_R \cap Y_n | \geq k = m/2$. Let $i_1, i_2, \ldots, i_k$, and $j_1, j_2, \ldots j_k$ be such numbers that

$\Pi_L \cap W_n = \{w_{i_1}, w_{i_2}, \ldots, w_{i_k}\}$ and $\Pi_R \cap Y_n \subseteq \{y_{j_1}, y_{j_2}, \ldots, y_{j_k}\}$.

Now, we fix the values of variables in $V_n \cup U_n$ in the following way. For any $d \in \{1, \ldots, k\}$, $u_d = 1$ if $d \in \{i_1, \ldots, i_k\}$, else $u_d = 0$. For any $d \in \{1, \ldots, k\}$, $v_d = 1$ if $d \in \{j_1, \ldots, j_k\}$, else $v_d = 0$. Thus, for every input α fulfilling the partial value assignment described above

$$h_n(L_{choice})(\alpha) = 1 \text{ iff } \bigwedge_{r=1}^{k} (w_{i_r} \equiv y_{j_r}) \text{ is equal to 1.}$$

Since we have shown a suitable assignment of values of variables in $U_n \cup V_n$ for every $\Pi \in \mathrm{Bal}_{w_n}(X_n)$ the proof can be completed in one of the already routine ways (either by constructing a 1-fooling set of cardinality 2^k or by showing that $M(h_n(L_{choice}), \Pi)$ contains a diagonal submatrix of size $2^k \times 2^k$). □

Note that the communication complexity of L_{choice} is also large, but to show it requires an extensive analysis dealing with all possible partitions of $W_n \cup Y_n \cup U_n \cup V_n$ into two equal-sided parts.

The next language $L_{sm} = \{wuwv \in \{0,1\}^* \mid 2|w| = |uv|\}$ is also an example presenting the advantages of s-communication complexity in creating the lower bound proofs.

Theorem 2.6.3.2 *For every $n = 32k$, $k \in \mathbb{N}$,*

$$\mathrm{scc}(h_n(L_{sm})) \geq k.$$

Proof. Let $W_n = \{w_1, w_2, \ldots w_{8k}\}$ and $U_n = \{u_1, u_2, \ldots, u_{24k}\}$ be the input variables of $h_n(L_{sm})$ for any $n \in N$. We show that $\mathrm{scc}(h_n(L_{sm}), \langle W_n \rangle) \geq k$. Let Π be a balanced partition from $\mathrm{Bal}_{W_n}(X_n)$. Without loss of generality we may assume $|\Pi_L| \leq |\Pi_R|$. Let $W_n \cap \Pi_L = \{w_{i_1}, w_{i_2}, \ldots, w_{i_{4k}}\}$ and $U_n \cap \Pi_R = \{u_{j_1}, u_{j_2}, \ldots, u_{j_z}\}$ for some $z \in \{12k, 12k+1, \ldots, 24k\}$. Using the same idea as in the proof of fact (ii) of Theorem 2.5.6.3 it can be shown that there exists $i \in \{8k, \ldots, 24k\}$ such that

$$| \{i + i_1, i + i_2, \ldots, i + i_{4k}\} \bigcap \{j_1, j_2, \ldots, j_z\} | \geq k.$$

Now, the proof can be completed in a way similar to that of Theorem 2.6.3.1. □

The last language

$$\begin{aligned} L_{dcf} = \{&0w_1 0w_2 0 \ldots 0w_{a-1} 0w_a 1^b 0w_a 0w_{a-1} 0 \ldots 0w_2 0w_1 1^c \mid \\ &a, b, c \geq 1, w_i \in \{0,1\} \text{ for } i = 1, 2, \ldots, a\} \end{aligned}$$

presented here is of special importance because L_{dcf} is a deterministic context-free language. So, showing a linear lower bound on $\mathrm{scc}(L_{dcf})$ we get the interesting claim that the parallel recognition of deterministic context-free languages (belonging among the simplest languages of the Chomsky hierarchy) has a non-negligible computational difficulty.

The idea of the lower bound proof for L_{dcf} is again based on a shift (determined by 1^b and 1^c) specifying which positions of the input have to be equal (compared).

Theorem 2.6.3.3 *For all sufficiently large positive integers n*

$$\mathrm{scc}(h_n(L_{dcf})) \geq n/32 - 1/2.$$

Proof. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of input variables of $h_n(L_{dcf})$, and let r be the even one of the numbers $\lfloor n/8 \rfloor$, $\lfloor n/8 - 1 \rfloor$. Let $Z = \{x_2, x_4, x_6, \ldots, x_{2r}\}$, $| Z | = r$. In what follows we say that a pair $\{i, j\}$, $i, j \in \{1, \ldots, n\}$, $i \neq j$, is divided by a partition Π of X iff neither Π_L nor Π_R involves both x_i and x_j.

Now, we prove that, for any partition $\Pi \in \mathrm{Bal}_Z(X)$, there exists a natural positive integer m, $2r < m < 6r$, such that at least $r/4$ of the pairs $\{2i, m + 2r - 2i + 2\}$, $i = 1, 2, \ldots, r$, are divided by Π. Consider the $r \times r$ Boolean matrix $M(\Pi) = [a_{ij}]_{i,j=1,\ldots,r}$ where $a_{ij} = 1(0)$ iff the pair $\{2i, 6r - 2j + 2\}$ is (not) divided by Π. Since exactly $r/2$ elements from $Z = \{x_2, x_4, \ldots, x_{2r}\}$ belong to Π_L, exactly $r^2/2$ elements $[(r/2)d + (r/2)(r - d)$, where d denotes the number of elements from $\{x_{6r}, x_{6r-2}, \ldots, x_{4r+2}\}$ belonging to $\Pi_R]$ of the matrix $M(\Pi)$ are equal to 1. The following $2r - 1$ disjoint sets

$A_q = \{a_{i,i+q} \mid i \in \{1, 2, \ldots r - q\}\}$ for $q = 0, 1, \ldots, r - 1$,

$A_p = \{a_{i,i+p} \mid i \in \{1 - p, 2 - p, \ldots, r\}\}$ for $p = -1, -2, \ldots, -(r - 1)$

are the diagonals of $M(\Pi)$, and so they form the partition of elements of

$M(\Pi)$ into $2r-1$ sets. Obviously, there must exist an integer u, $-(r-1) \leq u \leq r-1$, such that at least $r/4$ elements from A_u are equal to 1. This means we have for every $\Pi \in \mathrm{Bal}_Z(X)$ a set of integers $\{i_1, i_2, \ldots i_{\lfloor r/4 \rfloor}\} \subseteq \{1, 2, \ldots, r\}$ such that all the pairs $\{2i_1, 6r-2u-2i_1+2\}$, $\{2i_2, 6r-2u-2i_2+2\}, \ldots$, $\{2i_{\lfloor r/4 \rfloor}, 6r-2u-2i_{\lfloor r/4 \rfloor}+2\}$ are divided by Π.
Thus, we can see that $\mathcal{A}(h_n(L_{dcf}), \Pi)$ is a 1-fooling set for $h_n(L_{dcf})$ and Π, where

$$\begin{aligned}\mathcal{A}(h_n(L_{dcf}), \Pi) = \{&y_1 y_2 \ldots y_n \mid y_{i_j} = y_{6r-2u-2i_j+2} \in \{0,1\} \text{ for every} \\ &j \in \{1, \ldots, \lfloor r/4 \rfloor\},\ y_t = 1 \text{ for every integer } t \text{ such that} \\ &2r < t \leq 4r-2u \text{ or } 6r-2u < t \leq n, \text{ and } y_v = 0 \\ &\text{for all } v \notin \{i_j, 6r-2u-2_{i_j}+2 \mid j = 1, \ldots, \lfloor r/4 \rfloor\} \cup \\ &\{t \mid 2r < t \leq 4r-2u \text{ or } 6r-2u < t \leq n\}\}.\end{aligned}$$

We observe that the cardinality of $\mathcal{A}(h_n(L_{dcf}), \Pi)$ is $2\lceil r/4 \rceil$. □

2.6.4 Communication Complexity Versus s-communication Complexity

In this subsection we show some differences between s-communication complexity and communication complexity. We start with a concrete example of a language with linear s-communication complexity and constant communication complexity. Let L be a language with $\mathrm{cc}(h_n(L)) \in \Omega(n)$ (for instance, L_Δ, L_{shift} or any other language for which we have proved a linear lower bound on the communication complexity). We consider $\boldsymbol{U_L} = \{xy \in \{0,1\}^* \mid |x| = |y| \text{ and } x \in L\}$.

Theorem 2.6.4.1 *Let L be a language with $\mathrm{cc}(h_n(L)) = \Omega(n)$. Then*

$$\mathrm{cc}(h_n(U_L)) = 0 \text{ and } \mathrm{scc}(h_n(U_L)) = \Omega(n).$$

Proof. Let $X = \{x_1, x_2, \ldots, x_{2m}\}$, $n = 2m$, be the input variables of $h_n(U_L)$. To see that $\mathrm{cc}(h_n(U_L)) = 0$ it is sufficient to consider the partition $\overline{\Pi}$ of input variables with $\overline{\Pi}_L = \{x_1, \ldots, x_m\}$ (note that the variables $x_{m+1}, \ldots, x_{2m}$ assigned to the second computer are dummy variables). To show $\mathrm{scc}(h_n(U_L)) = \Omega(n)$ we consider $Z = \{x_1, x_2, \ldots, x_m\}$. It can be simply observed that $\mathrm{scc}(h_n(U_L), \langle Z \rangle) \geq \mathrm{cc}(h_n(L)) = \Omega(n)$. □

Theorem 2.6.4.1 is based on using dummy variables. But taking $\widetilde{U_L} = \{xy \in \{0,1\}^* \mid |x| = |y| \text{ and } x, y \in L\}$ instead of U_L we get also $\mathrm{cc}(h_n(\widetilde{U_L})) \leq 1$ and $\mathrm{scc}(h_n(\widetilde{U_L})) \geq \mathrm{cc}(h_n(L))$ and no dummy variable has been used. But the idea with dummy variables gives us the possibility to show that there is a large quantity of languages with different communication complexity and s-communication complexity. In Lemma 2.3.4.5 we have proved that $B_2^{2n}(n+m) \subseteq \mathrm{COMM}_{2n}(m)$ for every $m, n \in \mathbb{N}, m \leq n$. But almost all functions in $B_2^{2n}(n+m)$ have s-communication complexity equal to $\lfloor (n+m)/2 \rfloor$.

Lemma 2.6.4.2 *For every $n \in \mathbb{N}$, and every positive integer $m \leq n$,*

(i) $B_2^n(m) \subseteq \mathrm{SCOMM}_n^1(\lfloor m/2 \rfloor)$, *and*

(ii) $\lim_{n\to\infty} \frac{|B_2^n(m) \cap \mathrm{SCOMM}_n(\lfloor m/2 \rfloor - 1)|}{|B_2^n(m)|} = 0$ *for any m "growing" with n.*

Proof.

(i) Let $f \in B_2^n(m)$ be defined on variables in $X = \{x_1, \ldots, x_m, z_1, z_2, \ldots, z_{n-m}\}$, where $z_1, \ldots, z_{n-m}$ are the dummy variables of f. To prove (i) it is sufficient to show that $\mathrm{cc}(f, \langle Z \rangle) \leq \lfloor m/2 \rfloor$ for every $Z \subseteq X$. But, for every $Z \subseteq X$, there is a partition $\Pi \in \mathrm{Bal}_Z(X)$ such that the first computer has assigned at most $\lfloor m/2 \rfloor$ variables from $\Pi_L \bigcap \{x_1, \ldots, x_m\}$. Thus after receiving the values of all variables from $\Pi_L \cap \{x_1, \ldots x_m\}$, the second computer can immediately compute the result.

(ii) Let $f \in B_2^n(m)$ be defined on variables in $X = \{x_1, \ldots, x_m, z_1, \ldots, z_{n-m}\}$, where $z_1, \ldots, z_{n-m}$ are dummy variables of f. Let Π' be a partition of X, where $\Pi'_L = \{x_1, \ldots, x_m\}$ and $\Pi'_R = \{z_1, \ldots, z_{n-m}\}$. Since all variables in Π'_R are dummy ones we can define a Boolean function $f' \in B_2^m(m)$ as $f'(\alpha) = f(\Pi'^{-1}(\alpha, \beta))$ for every assignment α from $\{x_1, \ldots, x_m\}$ to $\{0,1\}$ and every input assignment β from $\{z_1, \ldots, z_{n-m}\}$to $\{0,1\}$. We observe that $\mathrm{scc}(f) \geq \mathrm{scc}(f, \langle \{x_1, \ldots, x_m\} \rangle) \geq \mathrm{cc}(f')$ because each protocol $\langle \Pi, \Phi \rangle$ computing f' for some $\Pi \in \mathrm{Bal}(\{x_1, \ldots, x_m\})$ can be directly (step by step) simulated by a protocol $\langle \Pi^1, \Phi^1 \rangle$, where $\Pi_L \subseteq \Pi_L^1, \Pi_R \subseteq \Pi_R^1, \Pi^1 \in \mathrm{Bal}_{\{x_1,\ldots,x_m\}}(X)$.

In Lemma 2.3.4.7 and in Theorem 2.4.4.4 we have proved that almost all Boolean functions $f' \in B_2^n(m)$ have $\mathrm{cc}(f') = \lfloor m/2 \rfloor$. So, following the consideration stated above we get that almost all $f \in B_2^n(m)$ have $\mathrm{scc}(f) \geq \lfloor m/2 \rfloor$. □

2.6.5 Some Properties of s-communication Complexity

All the theoretical questions related to communication complexity may be investigated also for s-communication complexity. Since almost all results can be achieved by using modifications of proof techniques used above for communication complexity we restrict our attention only on the following three fundamentals points:

1. Hierarchy of s-communication complexity classes,
2. Relation between one-way s-communication complexity and s-communication complexity,
3. Gap between nondeterministic s-communication complexity and deterministic one.

We start by dealing with point 1.

Theorem 2.6.5.1 *For every sufficiently large $n \in \mathbb{N}$, and every $m \in \mathbb{N}$, $n/2 \geq m \geq 1$,*

$$\mathrm{SCOMM}_n^1(m) - \mathrm{SCOMM}_n(m-1) \neq \emptyset.$$

Proof. The fact (i) of Lemma 2.6.4.2 claims that $B_2^n(2m) \subseteq \mathrm{SCOMM}_n^1(m)$. For sufficiently large n and $m \geq \log_2 n$, $B_2^n(2m) \not\subseteq \mathrm{SCOMM}_n(m-1)$ follows directly from the fact (ii) of Lemma 2.6.4.2. For $m \leq \log_2 n$, let us consider the language $L_{\mathrm{mod}\ m} = \{\alpha \in \{0,1\}^* \mid \#_0(\alpha) \bmod 2^m = 0\}$. Obviously $h_n(L_{\mathrm{mod}\ m}) \in \mathrm{SCOMM}_n^1(m)$. In Theorem 2.5.4.5 we have proved that $\mathrm{ncc}(h_n(L_{\mathrm{mod}\ m})) \geq m$, and so $\mathrm{scc}(h_n(L_{\mathrm{mod}\ m})) \geq \mathrm{cc}(h_n(L_{\mathrm{mod}\ m})) \geq \mathrm{ncc}(h_n(L_{\mathrm{mod}\ m})) \geq m$. □

So, we have again the property that one additional communication bit can bring more computational power than the extension of the one-way communication mode to the two-way communication mode. The following hierarchies are direct consequences of Theorem 2.6.5.1

Theorem 2.6.5.2 *For every sufficiently large positive integer n, and every $m \in \{1, 2, \ldots, n/2\}$,*

$$\mathrm{SCOMM}_n(m-1) \subsetneq \mathrm{SCOMM}_n(m),$$

and

$$\mathrm{SCOMM}_n^1(m-1) \subsetneq \mathrm{SCOMM}_n^1(m).$$

Theorem 2.6.5.3 *Let $g : \mathbb{N} \to \mathbb{N} - \{0\}$ be any unbounded function. Then:*

$$\mathrm{SCOMM}(g(n)-1) \subsetneq \mathrm{SCOMM}(g(n)),$$

and

$$\mathrm{SCOMM}^1(g(n))-1) \subsetneq \mathrm{SCOMM}^1(g(n)).$$

Now, we present the exponential gap between s-communication complexity and one-way s-communication complexity.

Theorem 2.6.5.4 *For any positive integers $n, m, m \leq n/2$*

$$\mathrm{SCOMM}_n(m) \subseteq \mathrm{SCOMM}_n^1(2^{m+1}).$$

Proof. To prove Theorem 2.6.5.4 it is sufficient to show that, for each protocol $D = \langle \Pi, \Phi \rangle$ with $\mathrm{cc}(D) = m$, there exists an equivalent one-way protocol $D' = \langle \Pi, \Phi' \rangle$ with $\mathrm{cc}(D) = 2^{m+1}$. But this fact is already proved in the proof of Theorem 2.4.4.1. □

Corollary 2.6.5.5 *For any function* $g : \mathbb{N} \to \mathbb{N}$, $g(n) \leq n/2$,

$$\mathrm{SCOMM}(g(n)) \subseteq \mathrm{SCOMM}^1(2^{g(n)+1}).$$

□

To show that the difference between $\mathrm{scc}_1(f)$ and $\mathrm{scc}(f)$ can be very large for some f, we consider the language $\overline{L}$ introduced in Theorem 2.4.3.4.

Theorem 2.6.5.6 *For any integer* $n = r2^r$, $r \in \mathbb{N}, r \geq 4$

(i) $\mathrm{scc}(h_n(\overline{L})) \geq (n/\log_2 n)^{1/2}$, *and*

(ii) $\mathrm{scc}(h_n(\overline{L})) \leq 2\log_2 n + 1$.

Proof. Since $\mathrm{scc}_1(f) \geq \mathrm{cc}_1(f)$ for every Boolean function f, and $\mathrm{cc}_1(h_n(\overline{L})) \geq (n/\log_2 n)^{1/2}$ has been proved in Theorem 2.4.3.4 , the fact (i) is proved. The fact (ii) is obvious (if not, see the proof of Theorem 2.4.4.3). □

Finally we shall deal with nondeterminism. To do it we need the following definitions.

Definition 2.6.5.7 *Let f be a Boolean function defined on a set X of Boolean variables. Let Z be a subset of X. The* **nondeterministic communication complexity of f according to Z** *is*

$$\mathbf{ncc}(f, \langle Z \rangle) = \min\{\mathrm{ncc}(f, \Pi) \mid \Pi \in \mathrm{Bal}_Z(X)\},$$

and the **nondeterministic s-communication complexity of f** *is*

$$\mathbf{sncc}(f) = \max\{\mathrm{ncc}(f, \langle Z \rangle) \mid Z \subseteq X\}.$$

Definition 2.6.5.8 *Let n, m be positive integers, $m \leq n/2$. We define*

$$\mathbf{SNCOMM}_n(m) = \{f \in B_2^n \mid \mathrm{sncc}(f) \leq m\}.$$

For any function $g : \mathbb{N} \to \mathbb{N}$, $g(n) \leq n/2$, we define

$$\mathbf{SNCOMM}(g(n)) = \{L \subseteq \{0,1\}^* \mid \mathrm{scc}(h_n(L)) \leq g(n) \text{ for every } n \in N\}.$$

Theorem 2.6.5.9 *Let n, m be positve integers, $m \leq n/2$. Then*

$$\mathrm{SNCOMM}_n(m) \subseteq \mathrm{SCOMM}^1(2^m).$$

Proof. The proof is the same as the proof of Theorem 2.5.4.1. □

Corollary 2.6.5.10 *For any function* $g : \mathbb{N} \to \mathbb{N}$, $g(n) \leq n/2$,

$$\mathrm{SNCOMM}(g(n)) \subseteq \mathrm{SCOMM}^1(2^{g(n)}).$$

Theorem 2.6.5.11 *For every* $n = \binom{m}{2}$, $m \in \mathbb{N}$,

(i) $\mathrm{scc}(h_n(L_\Delta)) = \Omega(n)$, *and*

(ii) $\mathrm{sncc}(h_n(L_\Delta)) \leq 2 + \lceil \log_2 n \rceil$.

Proof. The fact (i) follows from Theorem 2.3.3.2 claiming $\mathrm{cc}(h_n(L_\Delta)) \in \Omega(n)$. The fact (ii) is obvious. □

2.6.6 Exercises

Exercise 2.6.6.1 *Give a formal proof of the fact that the method A provides lower bounds on s-communication complexity for* $A \in \{$*subset-fool, subset-rank, subset-tiling*$\}$.

Exercise 2.6.6.2 *Give a formal proof of the fact that the method B provides lower bounds on one-way s-communication complexity for* $B \in \{$*subset-1fool, subset-mrow*$\}$.

Exercise 2.6.6.3 * *Prove the lower bound of Theorem 2.6.3.1 by using the subset-tiling method.*

Exercise 2.6.6.4 *Define the one-way (one-round) nondeterministic s-communication complexity of a Boolean function, and prove that it is always equal to unrestricted nondeterministic s-communication complexity.*

Exercise 2.6.6.5 *Prove that for every sufficiently large n and every positive integer* $m \leq n/2$,

$$\mathrm{SCOMM}^1_n(m) - \mathrm{SNCOMM}_n(m-1) \neq \emptyset.$$

Exercise 2.6.6.6 * *Define Las Vegas s-communication complexity and Monte Carlo s-communication complexities. Prove results about relations between randomized s-communication complexity measures, nondeterministic s-communication complexity, and deterministic complexity similar to the relations between different kinds of communication complexity measures presented in Section 2.5.*

Exercise 2.6.6.7 *Specify the lower bounds proof methods on nondeterministic s-communication complexity based on* 1*-fooling sets and matrix covers.*

Exercise 2.6.6.8 * *Compare the lower bound techniques subset-fool, subset-rank, subset-tiling each with each other. Find some specific languages for which different techniques render essentially different lower bounds.*

2.6.7 Problems

Problem 2.6.7.1 * *We know that for every even positive integer n, there exists an $f \in B_2^n$ such that $\mathrm{scc}(f) = n/2$, but the lower bound $\mathrm{scc}(h_n(L_{choice})) \geq n/8$ of Theorem 2.6.3.1 is one of the highest known lower bounds for explicitly defined functions.*

(i) Prove $\mathrm{scc}(h_n(L)) \geq n/d$ for some specific language L and $d < 4$.

(ii) ** *Prove (or find at least a candidate L) for some specific language L with the property*

$$\mathrm{scc}(h_n(L)) \geq n/2 - o(n).$$

Problem 2.6.7.2 *Find a smallest possible $k \in \mathbb{N}$ such that for all $n \geq k$, and all $m \in \{1, \ldots, n/2\}$*

$$\mathrm{SCOMM}_n^1(m) - \mathrm{SCOMM}_n(m-1) \neq \emptyset.$$

Note that we have proved this fact in Theorem 2.6.5.1 for "sufficiently large n".

Problem 2.6.7.3 * *Theorem 2.6.5.6 shows the largest known gap between s-communication complexity and one-way s-communication complexity for a specific language (existentially it is proved that there exists L with $\mathrm{cc}_1(h_n(L)) = \Omega(n)$ and $\mathrm{cc}(h_n(L)) = O(\log_2 n)$). Find a specific language S for which the gap between $\mathrm{scc}_1(h_n(S))$ and $\mathrm{scc}(h_n(S))$ is larger than the gap for the language $\overline{L}$ in Theorem 2.6.5.6.*

2.7 Bibliographical Remarks

The communication complexity of computing problems according to a fixed partition has been introduced by Abelson [Ab78, Ab80] and Yao [Ya79] in order to get a lower bound proof technique for distributive computing. Communication complexity as the minimum over all (almost) balanced partitions and its connection to lower bounds on VLSI computations have been explicitly introduced by Lipton and Sedgewick [LS81] and Yao [Ya81]. (Note that a lower bound proof technique based on information transfer was implicitly included in several previous work on VLSI lower bounds.) The formal definitions of protocols and communication complexity used here are based on those in Papadimitriou and Sipser [PS82, PS84a]. Another approach for defining communication protocols

was given by Lovász in [Lo89] (see Exercise 2.2.4.3), where a survey of communication complexity according to a fixed partition is given. Several authors interested only in a rough, asymptotical measurement of the communication complexity of concrete problems do not use any formal model of protocols. The formal models mentioned above are mainly used by researchers investigating also theoretical properties of communication complexity as an abstract complexity measure.

The lower bound method based on fooling sets is a version of the "crossing sequence argument" which is the most widely used lower bound argument in the classical (sequential) complexity theory (see, for instance, Hopcroft and Ullman [HU79]). The fooling method for proving lower bounds on VLSI computations was described by Brent and Kung [BK80, BK81], Abelson and Andreae [AA80], Thompson [Th79, Th80], Lipton and Sedgewick [LS81], Yao [Ya81], and Ullman [Ul84] (not as a lower bound method on communication complexity). The fooling (foolfix) method as a method providing lower bounds on communication complexity is first formulated by Aho, Ullman and Yannakakis in [AUY83]. The study of matrices $M(f, \Pi)$, and the introduction of the tiling (tilingfix) method has been introduced by Yao [Ya81]. The mrank (rankfix) method has been introduced by Mehlhorn and Schmidt [MS82]. Aho, Ullman, and Yannakakis [AUY83] were the first researchers dealing with the comparison of these lower bound proof methods for communication complexity according to a fixed partition (no paper dealing with the comparison of these methods as methods for general communication complexity is known to the author of this book). Most of the relationships between these methods and their relations to communication complexity according to a fixed partition are presented in Section 2.2.2. Theorems 2.2.2.15, 2.2.2.16, 2.2.2.17 were established in [AUY83]. The rest of the comparison results of Section 2.2.2 follows the work of Dietzfelbinger, Hromkovič and Schnitger [DHS94]. An inprovement of Theorem 2.2.2.28 has been proved by Hühne [Hue93]. The generalized fooling set method introduced in Exercise 2.2.4.15 is due to [DHS94]. A new lower bound method on communication complexity (not presented here) has been introduced by Kushilevitz, Linial, and Ostrovsky [KLO96]. Another survey on the comparison of lower bound proof methods for communication complexity can be found by Orlitsky and El Gamal [OG88]. The main open problem in the comparison study is the question whether the lower bounds provided by the rank method are polynomially close to communication complexity (Problem 2.2.5.1). Some nontrivial considerations connected with Problem 2.2.5.1 and the first examples of computing problems with at least a small difference between their communication complexity and their ranks are presented by Nisan and Wigderson [NW94], and by Raz and Spiker [RS93]. The assertion of Exercise 2.1.5.22 about the rank of random Boolean matrices has been proved by Komlós [Ko65, Ko68].

Theorem 2.2.3.3 showing the existence of a Boolean function f of $2n$ variables with $\mathrm{cc}(f, \overline{\Pi}) = n$ for a balanced partition $\overline{\Pi}$ was proved by Papadimitriou and Sipser [PS84a]. The rest of Section 2.2.3 devoted to the unclosure proper-

ties of communication complexity classes according to fixed partitions is due to Gubáš and Waczulík [GW86].

The strong communication complexity hierarchy of Theorems 2.3.4.8 and, 2.3.4.9 were established by Papadimitriou and Sipser [PS82, PS84a]. Theorems 2.3.4.13, 2.3.4.15, and 2.3.4.16 showing the extreme unclosure of communication complexity classes in a non-constructive way are chosen from [GW86]. A constructive proof of this fact is given by Gubáš and Waczulík in [GW87]. Theorem 2.3.3.2 giving a linear lower bound on the communication complexity of L_{Δ} has been established in [PS82]. The relation between the Chomsky hierarchy and the communication complexity hierarchy has been investigated by Hromkovič in [Hr86c], where Theorems 2.3.5.1, 2.3.5.2, and 2.3.5.5 were proved. Theorem 2.3.5.4 showing a context-free language of linear communication complexity was proved here in order to complete the comparison. The proof of Theorem 2.3.5.4 is based on an idea developed by Kumičáková [Ku89]. The idea to obtain hard functions (languages) for every balanced partition by taking new input variables "shifting" the original problem is due Vuillemin [Vu80].

One-way (one-round) protocols were first investigated by Papadimitriou and Sipser [PS82], where also the exponential gap between communication complexity and one-way communication complexity is established (Theorems 2.4.4.1 and 2.4.4.3). One-way fooling sets representing the classical crossing sequence argument are used as defined in [Hr89a]. Ullman [Ul84] uses a little bit different notion of one-way fooling sets based on a combination of one-way communication complexity of Yao [Ya81] and fooling sets of Lipton and Sedgewick [LS81]. The one-way fooling sets are related to one-way communication complexity in [Hr89b]. The extreme unclosure of one-way communication complexity under disjunction and conjunction (Theorem 2.4.4.5) is due to Gubáš and Waczulík [GW86].

Nondeterministic protocols were already considered by Lipton and Sedgewick [LS81] who pointed out that the fooling method provides lower bounds on nondeterministic communication, too. Nondeterministic communication was formally introduced by Papadimitriou and Sipser [PS82], where the exponential gap between communication complexity and nondeterministic communication complexity (Theorem 2.5.4.1 and 2.5.4.3) was proved. This result is of special interest for complexity theory because we have only a few examples of complexity measures for which one can prove that nondeterminism is much more powerful than determinism. The comparison of nondeterminism and determinism for sequential time is perhaps the central open problem of theoretical computer science. The lower bound method on nondeterministic communication complexity based on $\mathrm{Cov}(M(f, \Pi))$ has been investigated by Aho, Ullman, and Yannakakis [AUY83], who used it to show that $\mathrm{ncc}(f) \cdot \mathrm{ncc}(f^{\mathbf{C}})$ is an upper bound on $\mathrm{cc}(f)$ (Theorems 2.5.4.5 and 2.5.4.7). Theorem 2.5.4.5 showing that $m + 1$ bits deterministicly communicated can be more powerful than m bits used by nondeterministic protocols is due to Ďuriš, Galil, and Schnitger [DGS84]. The public nondeterministic communication complexity was introduced and investigated by Hromkovič and Schnitger [HrS96]. The assertion of Exercise 2.5.7.15 show-

ing the existence of superlogarithmic thresholds on the number of advice bits was proved in [HrS96]. Note that one does not have any proof of the existence of such thresholds on the degree of nondeterminism for other fundamental complexity measures. In this paper the notion of self-verifying nondeterminism was introduced too. A self-verifying nondeterministic protocol computing a function f must have at least one accepting computation for every α with $f(\alpha) = 1$, and at least one rejecting computation for every β with $f(\beta) = 0$. There may exist computations finishing with the answer "I do not know". Because of Theorems 2.5.4.5 and 2.5.4.7 self-verifying nondeterministic communication complexity is polynomially related to deterministic one. In [HrS96] the power of the degree of self-verifying nondeterminism for communication protocols is investigated too.

The power of randomization for communication protocols was investigated in several papers (see, for instance, Ablayev [Abl96], Chor and Goldreich [CG85], Fürer [Fue87], Halstenberg and Reischuk [HR88], Ja'Ja, Prassanna Kamar, and Simon [JPS84], King Pang and Abhasel Gamal [KA86], Kalyanasundaram and Schnitger [KS87], Mehlhorn and Schmidt [MS82], Meinel and Waack [MW92], Nisan and Wigderson [NW91], Newman [New91], Paturi and Simon [PaS84], Razborov [Ra90], and Yao [Ya83]) including some kinds of randomization. Here, we have restricted ourselves to Las Vegas randomization and Monte Carlo randomization with uniform probability distribution over the random variables, because these fundamental models of probabilistic computations suffice to show the advantages of randomized communications over deterministic. The results of Section 2.5.5 are selected from results established by Mehlhorn and Schmidt [MS82] and Ja'Ja, Prassanna Kamar and Simon [JPS84], except Example 2.5.5.6 based on the idea of Freivalds [Fr77]. Lemma 2.5.5.9 and Exercise 2.5.7.11 claiming that Las Vegas protocols are more powerful than the deterministic ones is due to Mehlhorn and Schmidt [MS82]. The assertions of Theorems 2.5.6.3 and 2.5.6.4 showing the power of Monte-Carlo probabilistic computations were obtained by Ja'Ja, Prassanna Kamar and Simon in [JPS84]. One of the main results on randomized protocols not presented here is the lower bound $\Omega(n^{1/2})$ on the two-sided-error Monte Carlo communication complexity of the disjointness given by Kalyanasundaram and Schnitger [KS87]. A simplified proof has been given later by Razborov [Ra90]. The linear relation between Las Vegas and determinism for one-way communication complexity was proved by Hromkovič and Schnitger [HrS95]. The fact that there is at most a logarithmic difference between the communication complexity of randomized protocols with public random bits and the communication complexity of randomized protocols with private random source [Exercise 2.5.7.18] was proved by Newman [New91]. A nice survey on randomized protocols was given by Schmetzer and Schnitger in [SS96].

Aho, Ullman, and Yannakakis [AUY83] have proposed to revise the definition of communication complexity for the reasons described in Section 2.6. The revised version of communication complexity called s-communication complexity was investigated in [Hr86a, Hr88b, Hr88c, Hr89a, Ku89]. All results of Section 2.6 are taken from Hromkovič [Hr89a], except the linear lower bound on

the s-communication complexity of the recognition of the deterministic context-free languages established by Kumičáková [Ku89].

We call attention to the fact that Chapter 2 announced the presentation only of some fundamental results devoted to theoretical aspects of the communication complexity measure. Next, we mention some further research directions and results connected with our model of communication protocols. Papadimitriou and Sipser [PS82] have introduced the k-round protocols (as defined in Definition 2.4.2.1) and k-round communication complexity as the minimum of communication complexities over k-round protocols. They have formulated a conjecture about the exponential gap between $(k+1)$-round communication complexity and k-round communication complexity. Ďuriš, Galil, and Schnitger [DGS84] have proved, for every positive integer k, the existence of a language L_k with logarithmic $(k+1)$-round communication complexity but linear k-round communication complexity. Furthermore, they have constructed a language L'_k with $(k+1)$-round communication complexity in $O(\log_2 n)$ and k-round communication complexity in $\Omega(n^{1/2}/(\log n)^3)$. The extension of the above results for some randomized protocols has been achieved by Halstenberg and Reischuk [HR88] in a non-constructive way, and by Nisan and Wigderson [NW91] in a constructive way. Obviously we have no hierarchy on the number of rounds of nondeterministic protocols because $\mathrm{ncc}(f) = \mathrm{ncc}_1(f)$ for every Boolean function f. Hromkovič and Schnitger [HrS96] proved that this equality must be payed for the increase of the number of advice bits. They showed that $(k+1)$-round (deterministic) communication complexity may be even much more powerful that k-round public nondeterministic communication complexity with a restricted number of advice variables.

One of the most intensively investigated directions in communication complexity is the comparison of different communication complexity classes. This is connected with the central studies in complexity theory dealing with the comparison of the power of determinism, nondeterminism, randomness, etc. (consider, for instance, P versus NP, DLOG versus NLOG, P versus R, etc.). Most of these principal questions have been open for a long period of time. Papadimitriou and Sipser [PS82] proposed to consider the classes $\mathrm{COMM}(O(\log_2 n))$ and $\mathrm{NCOMM}(O(\log_2 n))$ as analogs of P and NP and to investigate the power of nondeterminism, randomness, etc., according to communication complexity. So, they solved the analog of P versus NP problem by showing $\mathrm{COMM}(O(\log_2 n)) \subsetneq \mathrm{NCOMM}(O(\log_2 n))$ in [PS82] ($L_\Delta \in \mathrm{NCOMM}(O(\log_2 n))$ but $\mathrm{cc}(h_n(L_\Delta))$ is linear). A lot of further questions, open for time complexity classes, were solved for communication complexity classes by Babai, Frankl, and Simon [BFS86a, BFS86b], Damm, Krause, Meinel and Waack [DKMW92], Lam and Ruzzo [LaR81], and Lovász [Lo89].

Another important direction in the study of communication complexity is the investigation of the comunication complexity of concrete computing problems. Some results of this kind can be found for instance by Lovász [Lo89], Lovász and Saks [LS88], Meinel [Me92], and Papadimitriou and Sipser [PS84a].

An interesting fundamental question in the theory of computation is the direct-sum question: Can the cost of solving l independent instances of a problem simultaneously be smaller than the cost of independently solving the l problems, say, sequentially? Related to this question Karchmer, Raz and Wigderson [KRW91] introduced and investigated the amortized communication complexity as the communication complexity of simultaneously computing f on l instances, divided by l. Feder, Kushilevitz, Naor and Nisan [FKNN91], and Karchmer, Kushilevitz, and Nisan [KKN92] showed that the amortized communication complexity of a function can be smaller than its communication complexity for deterministic and randomized protocols.

Most of the communication complexity applications allow an additional restriction on communication protocols. Instead of the prefix-freeness property one requires that all messages submitted in the i-th round have the same length r_i for any i. The theoretical properties of this model have been investigated by Hromkovič in [Hr85, Hr86b].

The protocol model presented here is often called the "two-party protocol" because we have only two communicating computers. There are several other approaches considering more complicated models consisting of a few computers communicating via given communication links creating a so-called "communication structure". There are a lot of papers dealing with such models and we are not able to give a complete overview of publications in this research direction. For some approaches that are mostly close to our protocol model the papers by Chandra, Furst, and Lipton [CFL83], Dolev and Feder [DF92], and Tiwari [Ti87] can be consulted.

Further interesting versions of communication complexity were introduced and investigated in [LTT92, Ya82, GMW87, Ya86, Kus92]. Lam, Tiwari, and Tompa [LTT92] have started to investigate communication protocols where both computers have limited work space. The communication complexity of concrete problems is measured according to some space bounds on protocols. In [Ya82] Yao introduced so-called private communication complexity", where a protocol $\langle \Pi, \Phi \rangle$ has to compute a Boolean function f in a way such that no computer learns any additional information about the part of the input assigned to the other computer (in the information-theoretic sense), i.e., any other information than what follows from its part of the input and the function value $f(\alpha)$. Which Boolean functions can be computed by private protocols and with which communication complexity were investigated by Goldreich, Micali and Wigderson [GMW87], Kushilevitz [Kus92], and Yao [Ya86].

A completely different task is the investigation of the relation between communication complexity and other complexity measures of parallel and sequential computing. The rest of this book is devoted to this task. A nice survey on communication complexity and their applications was also written by Schmetzer and Schnitger [SS96].

3. Boolean Circuits

3.1 Introduction

This is the first chapter of this book to use the theoretical investigation of communication complexity in Chapter 2 in order to give lower bounds for some realistic parallel computations. This chapter is devoted to the relation between communication complexity and the complexity measures of Boolean circuits. More precisely, we show which lower bounds can be derived on the complexity measures of Boolean circuits computing a Boolean function f if one knows a lower bound on the communication complexity of f. The complexity measures considered are the layout area of Boolean circuits, the combinational complexity (number of gates (processors)) of Boolean circuits, and the depth (computing time) of Boolean circuits. To derive lower bounds on layout area and combinational complexity we use a standard application of communication complexity based on cutting a circuit into two parts and claiming that the circuit has to be large because of the necessary amount of information which must flow between these two circuit parts. To get a lower bound on the depth of Boolean circuits computing a specific function we need to introduce communication complexity of relations, which slightly differs from the communication complexity model investigated in Chapter 2.

Chapter 3 is organized as follows. Section 3.2 introduces the standard models of Boolean circuits and defines their complexity measures. It also provides some classical, fundamental results about these complexity measures. Section 3.3 presents the rules of the planar layout of Boolean circuits and shows that, for every Boolean function f, $(cc(f))^2$ provides a direct lower bound on the area of all Boolean circuits computing f. Some nonlinear lower bounds on the volume of the three-dimensional layout of Boolean circuits are also derived. Section 3.4 shows that we are able to obtain nonlinear lower bounds on Boolean circuits which have some "nice" recursive structures, but that communication complexity is unable to provide nonlinear lower bounds on general combinational complexity. In Section 3.5 we show that communication complexity provides nontrivial lower bounds on the size of unbounded fan-in Boolean circuits, which are a powerful generalization of standard Boolean circuits. Finally, Section 3.6 introduces a new communication game on our protocol model in order to provide a *method* for proving lower bounds on the depth of Boolean circuits.

3.2 Definitions and Fundamental Properties

3.2.1 Introduction

The aim of this section is to give the basic definitions connected with Boolean circuit computations and with the related complexity measures, and to provide some fundamental knowledge about Boolean circuit complexities. The section is organized as follows. Section 3.2.2 contains the formal definitions of the basic Boolean circuit models, and the definitions of complexity measures of Boolean circuit computations. Section 3.2.3 is devoted to some fundamental observations concerning the complexity measures defined in the previous section. Finally, Section 3.2.4 contains exercises presenting further basic results about Boolean circuits.

3.2.2 Boolean Circuit Models

Let us start with the basic Boolean circuit model, which is one of the oldest computing models of computer science. Informally, a Boolean circuit computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be viewed as a directed, acyclic "labeled" graph with the following properties:

(i) there are exactly n **input nodes** (sources) with indegree zero, each one of them labeled by one of the input variables (there is one-to-one correspondence between these n nodes and the n input variables),

(ii) each node of a non-zero indegree is called a **gate** (processor), it has indegree either one or two, and it is labeled by a Boolean function of one or two variables,

(iii) there is exactly one **output node** with outdegree zero.

A Boolean circuit computes as follows. Each node with a non-zero degree is considered to be a processor which is active only once during the whole computation on given input. The directed edges (u, v) of the circuit are considered to be communication links transferring the output Boolean value of the processor u as an input value to the processor (successor) v. Each processor v labeled by Boolean function h waits until all its inputs (one or two) are available, and then it computes the Boolean value h(the inputs) $= \alpha$, and sends this value α via all edges outcoming from v to all its successors. Obviously, at the beginning only the nodes (processors) are active whose incoming edges are outcoming from the input nodes. Later, each gate having its input values computes immediately the output which is submitted to all successors. Since the graph is acyclic, each processor works exactly once during the whole computation, and so the output processor computes unambiguously one output value, which is considered to be the output of the circuit for a given input.

Obviously, if one wants to have a Boolean circuit computing a problem P_n^m of n input variables and m output variables, then it is sufficient to consider m output nodes instead of one in the above stated informal definition.

Now we give a formal definition of Boolean circuits.

Definition 3.2.2.1 *Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables, and let* Bas $= \{h_1, h_2, \ldots, h_b\} \subseteq B_2^2 \cup B_2^1$ *be a set of Boolean functions of at most two variables. A* **straight-line Boolean program over the basis Bas and the variable set X** *is any sequence $g_1, g_2, \ldots, g_k$ in which each $g_j (j = 1, \ldots, k)$ is either*

(i) a Boolean variable from X, or

(ii) a Boolean constant 0 or 1, or

(iii) a pair (f, g_m) for an $f \in B_2^1 \cap$ Bas and a g_m with $m < j$, or

(iv) a triple (f, g_r, g_s) for an $f \in B_2^2 \cap$ Bas and some g_r, g_s with $1 \leq r, s < j$.

The elements of the sequence $g_1, g_2, \ldots, g_k$ over Bas *and X which consists of a variable or of a Boolean constant are called* **source elements**. *The remaining elements are called* **gates (processors)**. *For any gate $g_j = (f, g_r, g_s)$ [$= (f, g_m)$] the elements g_r, g_s [the element g_m] are [is] called the* **inputs [input]** *of g_j. Each gate which is no input for any gate is called the* **output of the straight-line Boolean program**.

For each element g_i of $g_1, \ldots, g_k$ we define a Boolean function over X as

$$\begin{aligned} \textbf{Result}(\mathbf{g_i}) &= g_i \text{ if } g_i \text{ is a source element,} \\ &= f(\text{Result}(g_m)) \text{ if } g_i = (f, g_m), \\ &= f(\text{Result}(g_r), \text{Result}(g_s)) \text{ if } g_i = (f, g_r, g_s). \end{aligned}$$

The set of Boolean functions computed by $g_1, \ldots g_k$ is

$$\begin{aligned} \boldsymbol{Fun(g_1, \ldots, g_k)} = \{&\text{Result}(g_m) \mid \text{for any } m \\ &\text{such that } g_m \text{ is an output of } g_1, \ldots, g_k\}. \end{aligned}$$

If $|Fun(g_1, \ldots, g_k)| = 1$ we say that $\boldsymbol{g_1, \ldots, g_k}$ **computes the Boolean function Result($\boldsymbol{g_k}$)**.

A straight-line Boolean program $g_1, \ldots, g_k$ over Bas *and X is called* **semilective** *if for every $x \in X$ there exists at most one $j \in \{1, \ldots, k\}$ such that $g_j = x$. Let $d \geq 2$ be an integer. We say that $g_1, \ldots, g_k$ is* **d-multilective** *if for every $x \in X$ there exist $k - d$ integers $1 \leq i_1, i_2, \ldots, i_{k-d} \leq k$ such that $x \neq g_{i_r}$ for every $r \in \{1, 2, \ldots, k - d\}$.*

In what follows, straight-line Boolean programs denote always semilective straight-line Boolean programs.

We illustrate the above definition by the following example. Consider the Boolean function f over $\{x_1, x_2, x_3, x_4\}$ defined by the following Boolean formula:

$$f(x_1, x_2, x_3, x_4) = (\Gamma(x_1 \vee x_2) \wedge (1 \oplus x_3)) \vee (\Gamma(x_4) \wedge (x_1 \vee x_2)).$$

The following straight-line Boolean program over $\{\vee, \wedge, \oplus, \Gamma\}$ computes $f(x_1, x_2, x_3, x_4)$:

$$\begin{array}{lll}
g_1 = & x_1 & \text{Result}(g_1) = x_1 \\
g_2 = & x_4 & \text{Result}(g_2) = x_4 \\
g_3 = & (\Gamma, g_2) & \text{Result}(g_3) = \Gamma(x_4) \\
g_4 = & x_3 & \text{Result}(g_4) = x_3 \\
g_5 = & x_2 & \text{Result}(g_5) = x_2 \\
g_6 = & 1 & \text{Result}(g_6) = 1 \\
g_7 = & (\vee, g_1, g_5) & \text{Result}(g_7) = x_1 \vee x_2 \\
g_8 = & (\Gamma, g_7) & \text{Result}(g_8) = \Gamma(x_1 \vee x_2) \\
g_9 = & (\oplus, g_6, g_4) & \text{Result}(g_9) = 1 \oplus x_3 \\
g_{10} = & (\wedge, g_8, g_9) & \text{Result}(g_{10}) = (\Gamma(x_1 \vee x_2)) \wedge (1 \oplus x_3) \\
g_{11} = & (\wedge, g_3, g_7) & \text{Result}(g_{11}) = \Gamma(x_4) \wedge (x_1 \vee x_2) \\
g_{12} = & (\vee, g_{10}, g_{11}) & \text{Result}(g_{12}) = f(x_1, x_2, x_3, x_4).
\end{array}$$

Definition 3.2.2.2 *Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables, $n \in \mathbb{N}$, and let* Bas $= \{h_1, h_2 \ldots, h_b\} \subseteq B_2^2 \cup B_2^1$ *be a set of Boolean functions. Let $g_1, g_2, \ldots, g_k$ be a straight-line Boolean program over the basis* Bas *and the set X of variables. A* **Boolean circuit over Bas and X corresponding to $A = g_1, \ldots, g_k$** *is an acyclic graph representation $BC_A = (V, E)$ of $g_1, \ldots, g_k$ described as follows:*

(i) $V = \{g_1, \ldots, g_k\}$,

(ii) For every gate g_i and for every input g_j of g_i, E contains the edge (g_j, g_i) leading from g_j to g_i.

We note that the Boolean circuit representation of straight-line Boolean programs defined above contains some repetition of information. According to Definition 3.2.2.2 each gate is labeled by a $g_j = (f, g_r, g_s)$ $[g_j = (f, g_r)]$ but it is sufficient to consider only the label f because the edges of BC_A determine unambiguously which are the inputs of g_j. Thus, in what follows we consider that each gate is labeled by one Boolean function from $B_2^2 \cup B_2^1$.

We observe that each straight-line Boolean program unambiguously determines one Boolean circuit. On the other hand there may exist several straight-line Boolean programs corresponding to the same Boolean circuit. This is because we can permute the gates of a straight-line Boolean program without changing its meaning as long as we preserve the property that the inputs of

each gate g_j are in the order before g_j. For instance, the above straight-line Boolean program computing the function $f(x_1, x_2, x_3, x_4)$ can be permuted to the straight-line Boolean program $q_1, q_2, \ldots, q_{12}$

$$\begin{aligned}
q_1 &= x_1 \\
q_2 &= x_2 \\
q_3 &= x_3 \\
q_4 &= x_4 \\
q_5 &= (\vee, q_1, q_2) \\
q_6 &= 1 \\
q_7 &= (\oplus, q_6, q_3) \\
q_8 &= (\Gamma, q_4) \\
q_9 &= (\Gamma, q_5) \\
q_{10} &= (\wedge, q_8, q_5) \\
q_{11} &= (\wedge, q_9, q_7) \\
q_{12} &= (\vee, q_{10}, q_{11}),
\end{aligned}$$

which corresponds to the same Boolean circuit (see Figure 3.1).

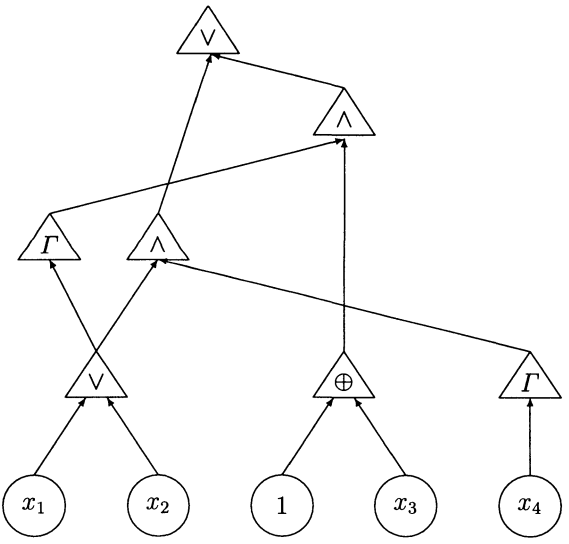

Fig. 3.1. A Boolean circuit computing the following Boolean function of four variables $f(x_1, x_2, x_3, x_4) = (\Gamma(x_1 \vee x_2) \wedge (1 \oplus x_3)) \vee (\Gamma(x_4) \wedge (x_1 \vee x_2))$

Thus, a straight-line Boolean program is unambiguously given by a Boolean circuit and some ordering of the nodes of the circuit. Obviously, this ordering

$v_1, v_2, \ldots, v_m$ must preserve the property that if there is a directed edge (v_i, v_j) in the circuit, then $i < j$. Any order of nodes of a Boolean circuit having this properties is called **topological** in what follows.

Usually, the combinational complexity of a Boolean circuit is considered to be its number of gates (i.e., the number of operations executed). Here, we shall prefer to define combinational complexity of a Boolean circuit as its number of nodes because this will simplify some of our later considerations. We may do so because for any Boolean function f of n variables the difference between the number of nodes of a Boolean circuit B computing f and the number of gates of B can be at most $n+2$, and we shall mostly deal with the asymptotic complexity of f.

Definition 3.2.2.3 *Let* $X = \{x_1, \ldots, x_n\}$ *be a set of* n *Boolean variables,* $n \in \mathbb{N}$, *and let* $\text{Bas} = \{h_1, h_2, \ldots, h_b\} \subseteq B_2^2 \cup B_2^1$ *be a set of Boolean functions. Let* B *be a Boolean circuit corresponding to a straight-line Boolean program* $A = g_1, g_2, \ldots, g_k$ *over* X *and* Bas. *The* **combinational complexity of** $\boldsymbol{B}$, *denoted* $\mathbf{CC}(\boldsymbol{B})$, *is* k *(the length of* A*). The* **depth of** $\boldsymbol{B}$, *denoted* $\mathbf{D}(\boldsymbol{B})$, *is the length (the number of edges) of the longest path in* B.
The **combinational complexity of a Boolean function** $\boldsymbol{f} \in \boldsymbol{B_2^n}$ **according to a basis Bas** *is*

$$\mathbf{CC_{Bas}}(\boldsymbol{f}) = \min\{\text{CC}(B) \mid B \text{ is a Boolean circuit over Bas and } X, \text{ and B computes } f\}.$$

The **combinational complexity of** $\boldsymbol{f}$ *is*

$$\mathbf{CC}(\boldsymbol{f}) = \text{CC}_H(f)$$

for $H = B_2^2 \cup B_2^1$. *The* **depth complexity of** $\boldsymbol{f}$ **according to a basis Bas** *is*

$$\mathbf{D_{Bas}}(\boldsymbol{f}) = \min\{D(B) \mid B \text{ is a Boolean circuit over Bas and } X \text{ and } B \text{ computes } f\}.$$

The **depth complexity of** $\boldsymbol{f}$ *is*

$$\mathbf{D}(\boldsymbol{f}) = \text{D}_H(f)$$

for $H = B_2^2 \cup B_2^1$.

These definitions of complexity measures of Boolean circuits can be easily extended to work for computing problems (sets of Boolean functions) instead for one-output problems (Boolean functions). Because this extension is straightforward (each straight-line program has exactly the same number of outputs as there are computed Boolean functions), and we shall mostly deal with one-output problems in what follows, we omit the formal definitions of complexity measures for many-output problems here.

We observe that Boolean circuits viewed as acyclic graphs have their indegree bounded by 2, and the outdegree is bounded by no constant independent of n (a gate g_i may be the input for all gates g_j with $j > i$). This is because each gate is a very simple processor computing a Boolean function of at most two variables, and each partial result is available for all following gates. But we are still interested in Boolean circuits corresponding to graphs with unbounded indegree. Obviously, allowing gates computing arbitrary Boolean functions is a nonsense because in this case the combinational complexity of every Boolean function would be 1. So, only some restricted sets of functions are considered as candidates for bases. Here, we shall mostly consider B_1^1 and the set of commutative and associative Boolean functions of two variables as the kernel for gate operations. The reason for this is the fact that commutative and associative functions like $\vee, \wedge, \oplus, \equiv$ are well understood independently of the number of variables over which they are applied. So, we define the unbounded fan-in Boolean circuits as follows.

Definition 3.2.2.4 *Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables , and let* Bas *be a (possibly infinite) set of Boolean functions. An* **unbounded fan-in straight-line Boolean program over the basis Bas and the variable set X** *is any sequence $g_1, g_2, \ldots, g_k$ in which each g_j $(j = 1, \ldots, k)$ is either*

(i) a Boolean variable from X, or

(ii) a Boolean constant 0 or 1, or

(iii) a tuple $(f, g_{i_1}, \ldots, g_{i_r})$ for some r-ary $f \in$ Bas, and $\{i_1, \ldots, i_r\} \subseteq \{1, 2, \ldots, j-1\}$, $i_r \neq i_s$ for $r \neq s$.

The elements of the sequence $g_1, g_2, \ldots, g_k$ consisting of a variable or of a Boolean constant are called **source elements**. *The remaining elements are called* **gates**. *If $g_j = (f, g_{i_1}, \ldots, g_{i_r})$, then g_{i_m} is called an* **input** *of g_j for every $m \in \{1, \ldots, r\}$. Each gate which is no input for any gate is called an* **output**. *For each element g_i of $g_1, \ldots, g_k$ we define a Boolean function over X as*

$$\begin{aligned} \mathbf{Result(g_i)} &= g_i \text{ if } g_i \text{ is a source element,} \\ &= f(\mathrm{Result}(g_{i_1}), \mathrm{Result}(g_{i_2}), \ldots, \mathrm{Result}(g_{i_r})) \\ &\quad \text{if } g_i = (f, g_{i_1}, \ldots, g_{i_r}). \end{aligned}$$

Let $g_{j_1}, \ldots, g_{j_m}$ be all outputs of the unbounded fan-in straiht-line Boolean program $g_1, g_2, \ldots, g_k$. Then

$$\boldsymbol{Fun(g_1, \ldots, g_k)} = \{\mathrm{Result}(g_{j_1}), \ldots, \mathrm{Result}(g_{j_m})\}$$

is the **set of Boolean functions computed by $g_1, g_2, \ldots, g_k$** .
If $|Fun(g_1, \ldots, g_k)| = 1$ then we say **$g_1, \ldots, g_k$ computes the Boolean function Result(g_k)**.

Definition 3.2.2.5 *Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables, $n \in \mathbb{N}$, and let* Bas *be a set of Boolean functions. Let $g_1, g_2, \ldots, g_k$ be an unbounded fan-in straight-line Boolean program over the basis* Bas *and the variable set X. An* **unbounded fan-in Boolean circuit over Bas and X corresponding to $A = g_1, \ldots, g_k$** *is an acyclic graph representation $BC_A = (V, E)$ of $g_1, \ldots, g_k$ described as follows:*

(i) $V = \{g_1, \ldots, g_k\}$

(ii) $E = \{(g_i, g_j) \in V \times V \mid g_i \text{ is an input of } g_j\}$.

The **combinational complexity of BC_A**, *denoted* $\mathbf{CC}(BC_A)$, *is the number of gates of $g_1, \ldots, g_k$. The* **depth of BC_A**, *denoted* $\mathbf{D}(BC_A)$, *is the length of the longest directed path in BC_A.*

In what follows we shall mainly consider the following infinite bases. For every $\triangle \in \{\vee, \wedge, \oplus, \equiv\} \subseteq B_2^2$ and every $r \in \mathbb{N}$, we define $g_r^{\triangle}(x_1, x_2, \ldots, x_r) = x_1 \triangle x_2 \triangle \ldots \triangle x_r$. We set $\boldsymbol{F} = B_1^2 \cup \{g_r^{\triangle} \mid \text{for every } \triangle \in \{\vee, \wedge, \oplus, \equiv\} \text{ and every positive integer } r\}$.

Definition 3.2.2.6 *Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables, $n \in \mathbb{N}$, and let* Bas *be a set of Boolean functions. The* **unbounded fan-in combinational complexity of a Boolean function $f \in B_2^n$ according to a basis** Bas *is*

$$\textbf{unfi-CC}_{\textbf{Bas}}(\boldsymbol{f}) = \min\{\mathrm{CC}(B) \mid B \text{ is an unbounded fan-in Boolean circuit over Bas and } X \text{ computing } f\}.$$

The **unbounded fan-in combinational complexity of f** *is*

$$\textbf{unfi-CC}(\boldsymbol{f}) = \textbf{unfi-CC}_{\boldsymbol{F}}(\boldsymbol{f}).$$

The **unbounded fan-in depth complexity of f according to a basis Bas** *is*

$$\textbf{unfi-D}_{\textbf{Bas}}(\boldsymbol{f}) = \min\{\mathrm{D}(B) \mid B \text{ is an unbounded fan-in Boolean circuit over Bas and } X \text{ which computes } f\}.$$

The **unbounded fan-in depth complexity of f** *is*

$$\textbf{unfi-D}(\boldsymbol{f}) = \text{unfi-D}_{\mathrm{F}}(f).$$

Let B_1 and B_2 be two Boolean circuits. In what follows we say that **B_1 and B_2 are equivalent** if they compute the same function.

3.2.3 Fundamental Observations

In the previous section we have introduced basic complexity measures of Boolean circuits. In some sense we have a lot of them because we have parameterized these measures according to different bases. The first fundamental observation is that, studying asymptotic behavior of these measures for Boolean circuits, it is not essential which complete base is considered.

Observation 3.2.3.1 *Let S and R be two complete bases, $S, R \subseteq B_2^2 \cup B_2^1$. Then there exists a constant $\mathrm{const}(S,R)$ depending only on S and R such that*

$$\mathrm{CC}_R(f) \leq \mathrm{const}(S,R) \cdot \mathrm{CC}_S(f)$$

for any Boolean function $f \in \bigcup_{i=0}^{\infty} B_2^i$.

Proof. Let $f \in B_2^n$ for some $n \in \mathbb{N}$, and X be the set of input variables of f. Let $B_1 = g_1, \ldots, g_m$ be a straight-line Boolean program over S and X computing f. Since R is a complete base, each function of S can be computed by a straight-line Boolean program over R and X. Let $\mathrm{const}(S,R)$ be the maximal length of these straight-line Boolean programs over R and X. Thus, if $g_r = (h, g_i, g_j)$ for some $h \in S, i, j < r$, then we can exchange g_r for the straight-line Boolean program over R and X computing h on the arguments g_i and g_j. Obviously, in this way we get a straight-line Boolean program B_2 over R and X computing f, and the length of B_2 is at most $\mathrm{const}(S,R) \cdot m = \mathrm{const}(S,R) \cdot \mathrm{CC}_S(B_1)$. □

The next observation deals with the question what happens if one adds a natural restriction on Boolean circuits requiring the outdegree bounded by 2 (and so the degree bounded by 4). We show that this restriction is not essential for the study of the asymptotic combinational complexity of concrete functions.

Observation 3.2.3.2 *Let X be a set of Boolean variables. Let S be a base containing the identity function. For each Boolean circuit B over S and X, there exists an equivalent Boolean circuit B' over S and X with outdegree bounded by 2 and with*

$$\mathrm{CC}_S(B') \leq 6 \cdot \mathrm{CC}_S(B).$$

Proof. Let $g_i = (h, g_r, g_s)$ be a gate of B with outdegree $k \geq 3$. Then we exchange the edges outcoming from g_i by a a binary tree with k leaves. The internal nodes (gates) of the tree realize the identity function, and the root of the tree is g_i, and the edges are directed from the root to the leaves (see Figure 3.2).

Obviously, if we exchange all gates with degree at least 3 by the trees with the identity gates described above, we get a new Boolean circuit B' equivalent to B. To estimate $\mathrm{CC}(B')$ we first observe that the number of edges in B is at most $2 \cdot \mathrm{CC}(B)$ because each gate of B has indegree bounded by 2. Secondly, we observe that the binary tree replacing the k edges leading from g_i has at most

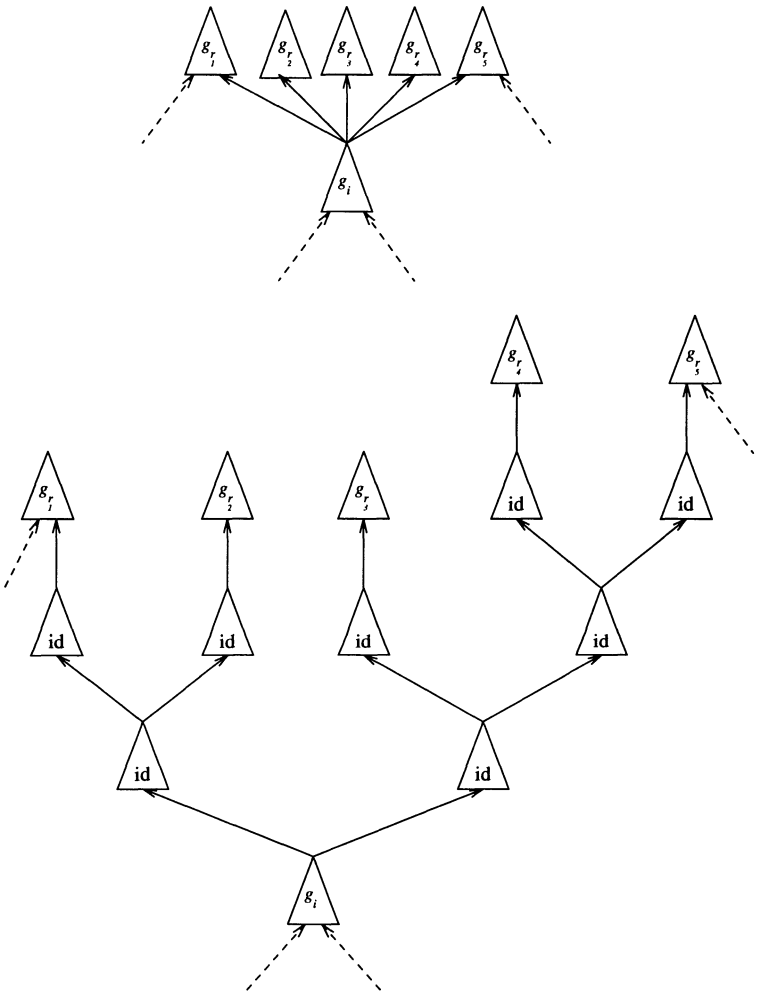

Fig. 3.2. The simulation of an unbounded fan-out gate by a circuit with outdegree bounded by 2.

$2k$ internal edges leading to the identity gates. Thus, B' has at most $6 \cdot \mathrm{CC}(B)$ edges, i.e., $\mathrm{CC}(B') \leq 6 \cdot \mathrm{CC}(B)$. □

In Section 2.1.1 we have defined Boolean formulas as formulas over the base $\overline{\mathrm{Bas}} = \{\Gamma, \vee, \wedge, \oplus, \equiv, \Rightarrow\}$. We define, for any Boolean formula F, the **complexity (length) of F** as a nonnegative integer $\boldsymbol{L(F)}$ equal to the number of the occurrences of Boolean operations in F. For instance, $L(F_1) = 5$ for $f_1 = (\Gamma(x_1) \vee x_2) \Rightarrow (x_3 \vee \Gamma(x_2))$. The **formula complexity of a Boolean function f**, is

$$\boldsymbol{L(f)} = \min\{L(F) \mid F(\alpha) = f(\alpha) \text{ for every input } \alpha\}.$$

The following fact is obvious.

Observation 3.2.3.3 *For every Boolean function $f \in B_2^n$*

$$n + L(f) \geq \mathrm{CC}_{\overline{\mathrm{Bas}}}(f) \geq \mathrm{CC}(f).$$

The rest of this subsection is devoted to estimating the value

$$\mathbf{ShCC(}\boldsymbol{n}\mathbf{)} = \max\{\mathrm{CC}(f) \mid f \in B_2^n\},$$

called **Shannon's function of combinational complexity**. Obviously, the number $\mathrm{ShCC}(n)$ is the combinational complexity of the hardest function of B_2^n, i.e., the minimal combinational complexity sufficient to compute any Boolean function of B_2^n. Despite the fact that there exist very fine estimates of $\mathrm{ShCC}(n)$ in the literature, we give only some rough estimate of $\mathrm{ShCC}(n)$ which are sufficient for our purposes and do not require any hard technical consideration.

We start with an upper bound on $\mathrm{ShCC}(n)$.

Definition 3.2.3.4 *Let $X_n = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. We denote by $\boldsymbol{x_i^1}$ the variable x_i and by $\boldsymbol{x_i^0}$ the formula x_i^c for any $i \in \{1, \ldots, n\}$. Let $\{i_1, \ldots, i_r\}$ be a subset of $\{1, \ldots, n\}$. For each vector $(\alpha_1 \ldots, \alpha_r) \in \{0,1\}^r$ the formula $x_{i_1}^{\alpha_1} \wedge x_{i_2}^{\alpha_2} \cdots \wedge x_{i_r}^{\alpha_r}$ is an* **elementary conjunction over $\boldsymbol{X_n}$**, *and the formula $x_{i_1}^{\alpha_1} \vee x_{i_2}^{\alpha_2} \vee \cdots \vee x_{i_r}^{\alpha_r}$ is an* **elementary disjunction over $\boldsymbol{X_n}$**.

Lemma 3.2.3.5 *For any $n \in \mathbb{N}$,*

$$\mathrm{ShCC}(n) \leq n \cdot 2^{n+1}.$$

Proof. Let f be a Boolean function of B_2^n and $X_n = \{x_1, \ldots, x_n\}$ be the Boolean variables over which the functions in B_2^n are defined. Let $N^1(f)$ be the set of inputs which satisfies f. We observe that, for each $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{0,1\}^n, F_\alpha = \bigwedge_{i=1}^n x_i^{\alpha_i}$ corresponds to a Boolean function h over X_n with $N^1(h) = \{\alpha\}$. Thus the following formula

$$F_f = \bigvee_{\alpha \in N^1(f)} F_\alpha$$

corresponds to f ($F_f(\beta) = f(\beta)$ for every $\beta \in \{0,1\}^n$). Since $L(F_f) \leq |N^1(f)| \cdot L(x_1^c \wedge x_2^c \wedge \cdots \wedge x_n^c) \leq 2^n \cdot (2n-1)$ we get $\mathrm{CC}(f) \leq n \cdot 2^{n+1}$. □

Now we give a lower bound on Shannon's function of combinational complexity by using a simple "counting argument". The idea of the counting argument is based on the fact that for two different Boolean functions one must have two different straight-line Boolean programs. Since the number of different Boolean functions of n variables is 2^{2^n}, we need 2^{2^n} different straight-line Boolean programs (circuits) to compute all functions of B_2^n. What remains is to show how complex straight-line programs must be taken to have the possibility to get 2^{2^n} different ones.

Lemma 3.2.3.6 *For any* $n \in \mathbb{N}$,

$$\mathrm{ShCC}(n) \geq 2^{n-2}/n.$$

Proof. We binary code each straight-line Boolean program in the following way. Let $B = g_1, \ldots, g_k$ be a straight-line Boolean program over $X = \{x_1, \ldots, x_n\}$ and $B_2^2 \cup B_2^1$. Without loss of generality we assume $k \geq n$. We shall code each g_i as a $\mathrm{Cod}(g_i) \in \{0,1\}^l$ for $l = 2d+6$, $d = \lceil \log_2 k \rceil$ as follows. If g_i ($i \in \{1, \ldots, k\}$) is a Boolean variable for some $j \in \{1, \ldots, n\}$, then $\mathrm{Cod}(g_i) = 00\mathrm{BIN}_d^{-1}(j)0^{d+4}$. If g_i is a Boolean constant $\alpha \in \{0,1\}$, then $\mathrm{Cod}(g_i) = 11\alpha 0^{2d+3}$. In what follows, for each $h \in B_2^1$, let $\mathrm{Cod}(h)$ be a code of h from $\{0,1\}$. If $g_i = (h, g_r)$ for some $h \in B_2^1$ and $1 \leq r < i$, then $\mathrm{Cod}(g_i) = 01\mathrm{Cod}(h)000\mathrm{BIN}_d^{-1}(r)0^d$. Obviously, we can also find a coding of functions of B_2^2, such that $\mathrm{Cod}(h) \in \{0,1\}^4$ for each $h \in B_2^2$. Thus, if $g_i = (h, g_r, g_s)$ for some $h \in B_2^2$, $1 \leq r, s < i$, then $\mathrm{Cod}(g_i) = 10\mathrm{Cod}(h)\mathrm{BIN}_d^{-1}(r)\mathrm{BIN}_d^{-1}(s)$. Finally, we set $\mathrm{Cod}(B) = 0^k 1\mathrm{Cod}(g_1)\mathrm{Cod}(g_2)\cdots\mathrm{Cod}(g_k)$. So, we have unambiguously assigned a $\mathrm{Cod}(B)$ of length $k+1+k \cdot l = k+1+k(2\lceil \log_2 k \rceil + 6) \leq k(2\lceil \log_2 k \rceil + 8)$ to each straight-line Boolean program of length k. Since two different straight-line programs have two different codes we get that one must have at least 2^{2^n} different codes to be able to code all straight-line Boolean programs computing all functions of B_2^n. Since the codes are binary words we obtain that this is possible only if the number of different words of length at most $k \cdot (\lceil \log_2 k \rceil + 8)$ is greater than 2^{2^n}. Following this we have

$$2^{k(2\lceil \log_2 k \rceil + 8)+1} \geq 2^{2^n},$$

which implies $k \geq 2^{n-2}/n$. □

3.2.4 Exercises

Exercise 3.2.4.1 *Write a straight-line Boolean program corresponding to the Boolean circuit in Fig. 3.1 and differing from the two straight-line programs for this circuit presented in Section 3.2.2.*

Exercise 3.2.4.2 *Design a Boolean circuit equivalent to the Boolean circuit in Fig. 3.1, but having fewer gates than the circuit in Fig. 3.1.*

Exercise 3.2.4.3 *Prove that for each Boolean function f, there exists an unbounded fan-in Boolean circuit B_f over F computing f with $D(B_f) \leq 3$.*

Exercise 3.2.4.4 *Prove, that for each Boolean function $f \in B_2^n$ there exists a Boolean circuit B'_f over $B_2^1 \cup B_2^2$ computing f with $D(B'_f) \leq n + \log_2 n$.*

Exercise 3.2.4.5 *Find the smallest constant k such that*

$$\mathrm{CC}(f) \leq k \cdot \mathrm{CC}_{\{\wedge,\vee,\neg\}}(f)$$

for any Boolean function f.

Exercise 3.2.4.6 * *Search for the smallest possible constant t such that*

$$\mathrm{CC}_{B_1}(f) \leq t \cdot \mathrm{CC}_{B_2}(f)$$

for any complete bases B_1 and B_2.

Exercise 3.2.4.7 * *Prove*

$$\mathrm{ShCC}_{\{\wedge,\vee,\neg\}}(n) = \frac{2^n}{n} \cdot (1 + o(1)).$$

Exercise 3.2.4.8 * *Let $f \in B_2^n$ be a linear Boolean function. Then*

$$\mathrm{CC}_{\{\wedge,\vee,\neg\}}(f) \leq 4n - 4.$$

Exercise 3.2.4.9 * *Prove that, for any sufficiently large n, there exists a linear Boolean function $h_n \in B_2^n$ such that*

$$\mathrm{CC}_{\{\wedge,\vee,\neg\}}(h_n) = 4n - 4$$

Exercise 3.2.4.10 *Design a Boolean circuit computing the sum of two integers a and b, binary coded on the length n (i.e., the input is of length $2n$ and the output of length $n + 1$).*

3.3 Lower Bounds on the Area of Boolean Circuits

3.3.1 Introduction

The main aim of this section is to show that communication complexity can be used to get quadratic lower bounds on the layout area of Boolean circuits. Note that no other technique before has brought any nonlinear lower bound on the area complexity of Boolean functions. This lower bound technique is the starting point for us to attack the fundamental problem concerning the proof of a nonlinear lower bound on combinational complexity in Section 3.4. It is one of the simplest standard applications of communication complexity based on a division of a circuit into two parts in such a way that each part contains approximately half the inputs. Then the communication complexity $cc(f)$ of a Boolean function f gives a lower bound on the number of edges leading between these two parts. This fact can be used to get a lower bound on the area of any Boolean circuit computing f.

This section is organized as follows. Section 3.3.2 defines the layout of Boolean circuits as well as the Boolean circuit area complexity of Boolean functions. Section 3.3.3 presents the main assertion of the section — the relation between communication complexity and Boolean circuit area. Section 3.3.4 compares two kinds of layouts of Boolean circuits in the plane. Section 3.3.5 is devoted to the three-dimensional layout of Boolean circuits and its relation to communication complexity. After that, as usual, exercises and open problems are formulated.

3.3.2 Definitions

Considering the approaches in the study of circuits, we see that layout area of the circuit is a more important complexity measure from the practical point of view than combinational complexity. The main reason for the above claim is that the layout area exactly corresponds to the size of the chip produced, and the fact that the cost of production and the unreliability of the chip grow exponentially with the size (layout area) of the chip. [To see this, consider a technological process producing reliable chips of size A with probability p, $0 < p < 1$ (for instance, if $p = 1/2$ then approximately every second chip produced by this technology can be used and the rest must be scrapped). Then to produce a chip of the same kind but twice as large as before the probability of reliable production is decreased to p^2 (if $p = 1/2$, then only every 4th chip produced is reliable). Thus, if one wants to produce a chip of size $k \cdot A$, then the probability of producing a reliable chip is decreased to p^k.]

Above we have seen the reasons to investigate the area complexity of circuits but before starting to do so we need a precise definition of area complexity, which strongly depends on the layout rules for Boolean circuits. The layout rules described in the following definitions follow the technological (electrical) requirements that:

(i) the distance between any two physical links (switches) laid in parallel is at least λ (a physical constant assuring that the electrical stream in one switch has no essential influence on the voltage of another switch)

(ii) the distance between any two processors in the layout is at least λ.

Note that we can restrict our study to Boolean circuits with the degree bounded by 4 as already mentioned in Section 3.2.2.

Definition 3.3.2.1 *Let $G = (V, E)$ be a directed graph of degree at most 4. A* **grid-graph** $\overline{G}$ *of $G = (V, E)$ is a layout of G into the two-dimensional lattice with the following properties:*

1. *Each square of the lattice has some of the following contents:*
   - *(a) a vertex of V* [see Figure 3.3a]
   - *(b) a straightforward part of a line going in the horizontal or in the vertical direction (this line is a part of the layout of an edge of the graph)* [see Figure 3.3b]
   - *(c) a broken line coming in the lattice square in one of two vertical (horizontal) directions and coming out in one of two horizontal (vertical) directions* [see Figure 3.3c]
   - *(d) a crossing of two lines, one going in the horizontal direction, the other one in the vertical directions (this depicts the crossing of two edges of E in the layout)* [see Figure 3.3d]
   - *(e) the empty content.*
2. *If one square of the lattice contains a node of V, then none of the 8 neighboring squares of this square contains a node of V.*

The **area complexity of $\overline{G}$, $\mathbf{A}(\overline{G})$,** *is the area of the minimal rectangle* ***Rect***$(\overline{G})$ *comprising all nonempty squares of the lattice.*

An example of the *Rect*$(\overline{G})$ of a grid-graph $\overline{G}$ of the complete binary tree of depth 4 with edge direction from the leaves to the root is depicted in Figure 3.4. The area complexity of this grid-graph is $12 \cdot 12 = 144$.

Definition 3.3.2.2 *Let $G = (V, E)$ be a directed graph of degree at most 4. The* **layout area of G** *is* $\mathbf{A}(G) = \min\{\mathrm{A}(\overline{G}) \mid \overline{G}$ *is a grid-graph of* $G\}$. *Let B be a Boolean circuit with both indegree and outdegree bounded by 2. The* **area complexity of B** *is* $\mathbf{AC}(B) = \mathrm{A}(B)$.

For any Boolean function f, the **area complexity of f** *is*

$$\mathbf{AC}(f) = \min\{\mathrm{AC}(B) \mid B \text{ computes } f\}.$$

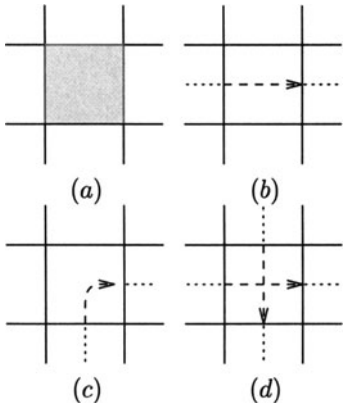

Fig. 3.3. Possible contents of the lattice squares of a grid-graph

Observation 3.3.2.3 *For every Boolean function* $f \in B_2^n, n \in \mathbb{N}$,

$$\mathrm{AC}(f) \leq 16 \cdot (\mathrm{CC}(f))^2 .$$

Proof. Let B be a Boolean circuit of m nodes and e edges. We consider an $(e+2) \times 4m$ lattice. We lay all nodes of B into the second row of the lattice in such a way that, for each square X containing a node of B, X does not lie on the border of the lattice and, for any two squares Y and Z containing nodes of B, the distance between X and Z is at least 3 (see Figure 3.5). For each square of the lattice containing a node v of B, we define the country of v, $C(v)$, as the part of the second row of the lattice involving the square X containing v, the left neighbor of X, and the two next squares laying to the right from X. Now, the edges are laid as follows. A number $n(e)$ of $\{3, 4, \ldots, e+2\}$ is assigned to each edge e of B in such a way that two different edges have two different numbers. An edge (u, v) is laid as a line first going from u vertically via one of the four columns containing an element of $C(v)$ to the $n((u,v))$-th row, then going horizontally via the $n((u,v))$-th row of the lattice to a column Z containing an element of $C(u)$, and finally going to v via the column Z. Since $e \leq 2m$ in Boolean circuits of degree 4, we get the assertion of Observation 3.3.2.3. □

Figure 3.5 illustrates the layout of the Boolean circuit computing the function $(x_1 \vee x_2) \wedge x_1$. We see that one row and two columns are reserved for the layout of each edge.

A typical requirement on circuits in practice is that the inputs and outputs (input ports and output ports) lie on the border of the circuit layout. Since the area complexity of circuits with inputs and outputs on the border of the layout

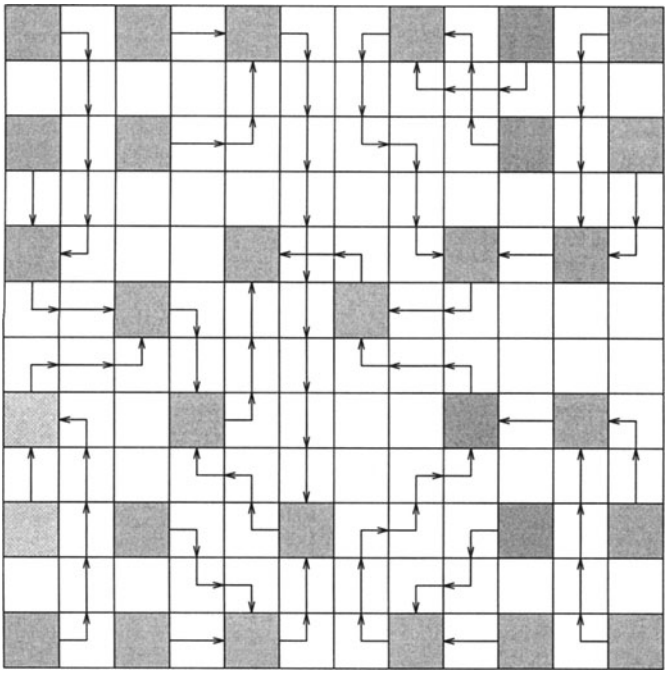

Fig. 3.4. A grid-graph

may essentially differ from the area complexity defined above, we also define this special area complexity measure.

Definition 3.3.2.4 *Let $G = (V, E)$ be a directed graph of degree at most 4, and let U be a subset of V.* **The b-layout area of G according to U** *is*

$$\mathbf{bA}(G, U) = \min\{\mathrm{A}(\overline{G}) \mid \overline{G} \text{ is a grid-graph of } G \text{ containing all nodes of } U \text{ on the border of the minimal rectangle comprising } \overline{G}\}.$$

Let B be a Boolean circuit with both indegree and outdegree bounded by 2, with the set of input nodes X, and with the set of output nodes Y. The **b-area complexity of B** *is*

$$\mathbf{bAC}(B) = \mathrm{bA}(B, X \cup Y).$$

For any Boolean function f, the **b-area complexity of f** *is*

$$\mathbf{bAC}(f) = \min\{\mathrm{bAC}(B) \mid B \text{ computes } f\}.$$

3.3.3 Lower Bounds on the Area Complexity Measures

The aim of this subsection is to show which lower bounds on the area complexity measures are provided by communication complexity. In this case we consider

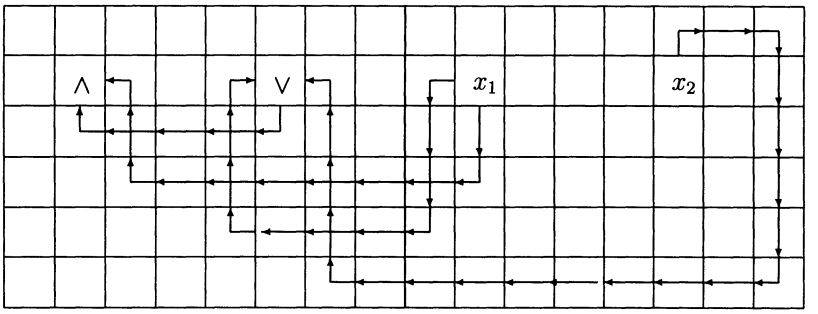

Fig. 3.5. A grid-graph of a Boolean circuit computing the Boolean function $f(x_1, x_2) = (x_1 \vee x_2) \wedge x_1$

a typical use of communication complexity based on the fact that the number of communication bits flowing via the edges dividing a circuit into two parts must be at least as large as the communication complexity according to the partition of input variables corresponding to the cut of the circuit. Since each edge of a Boolean circuit transfers only one bit during the whole computation on an input assignment, the communication complexity provides a lower bound on the number of edges of the above mentioned circuit division. Formalizing this idea in what follows we show how this results in lower bounds on the area complexities of Boolean circuits.

Definition 3.3.3.1 *Let $G = (V, E)$ be a directed graph (Boolean circuit). (E', V_1, V_2) is called a* **cut of G** *if $E' \subseteq E$, $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ and $E' = E \cap (V_1 \times V_2 \cup V_2 \times V_1)$. A* **vertex-cut of G** *is any triple (V', U_1, U_2) such that $V' \subseteq V$, $V' \cap U_1 = V' \cap U_2 = U_1 \cap U_2 = \emptyset$, $V = V' \cup U_1 \cup U_2$ and $E \cap ((U_1 \times U_2) \cup (U_2 \times U_1)) = \emptyset$.*

Note that is some cases an edge-cut (E', V_1, V_2) of a graph G can be given by the set E' (or by the sets V_1, V_2) only. Similarly to describe a vertex-cut (V', U_1, U_2) it may be sufficient to give V' only.

Definition 3.3.3.2 *Let $G = (V, E)$ be a directed graph with a degree bounded by 4. Let $\overline{G}$ be a grid-graph of G, and let $Rect(\overline{G})$ be the minimal lattice rectangle comprising $\overline{G}$. A* **cut-line of $Rect(\overline{G})$** *is any continuous line ω laying on the edges of the lattice with disjoint endpoints laid on the border of $Rect(\overline{G})$* [see Figure 3.6 for an example]. *The* **length of ω** *is the number of lattice edges of ω.* ***E*(ω)** *is the set of all edges of E crossing the line ω (i.e., all edges of E crossing the edges of the lattice which are parts of ω). Let V_1, V_2 be subsets of V such that $E(\omega) = E \cap (V_1 \times V_2 \cup V_2 \times V_1)$, $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$. If there are several possibilities to choose V_1 and V_2 with the above properties we*

fix them unambiguosly as sets of vertices corresponding to the two components of the grid-graph G' given by the line ω. Then $(E(\omega), V_1, V_2)$ is called the **cut of G induced by the line ω.**

Lemma 3.3.3.3 *Let $G = (V, E)$ be a Boolean circuit (a directed graph of a degree at most 4), and let V' be a nonempty subset of V. Let $\overline{G}$ be a grid-graph of G, and let $Rect(\overline{G})$ have size $a \times b$, $a \geq b$. Then there exists a cut-line ω of $Rect(\overline{G})$ such that*

(i) the length of ω is at most $b + 1$, and

(ii) the cut $(E(\omega), V_1, V_2)$ of G induced by ω fulfills

$$-1 \leq |V_1 \cap V'| - |V_2 \cap V'| \leq 1.$$

Proof. We give an algorithm constructing a cut-line with the properties (i) and (ii). The line will be unambiguously defined by the sets S_1 and S_2 of lattice squares produced by the algorithm. At the beginning $S_1 = \emptyset$ and S_2 contains all squares of $Rect(\overline{G})$. In the first step the algorithm puts the square of the top-left corner of $Rect(\overline{G})$ from S_2 to S_1. If S_1 does not contain at least $\lceil |V'|/2 \rceil$ nodes of V', then the algorithm continues by moving one square of S_2 to S_1. Obviously, this can be done in such a way that the line between S_1 and S_2 involves at most one single jog and so its length is at most $b + 1$. (The squares are first chosen from the first column top down, then from the second column top down, etc.) The algorithm halts when S_1 contains $\lceil |V'|/2 \rceil$ nodes of V'. □

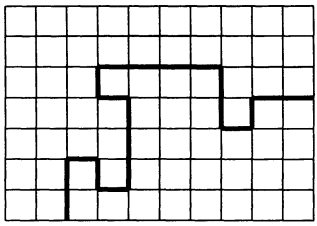

Fig. 3.6. A cut-line of the 10×7 lattice of length 20

Now we are prepared to formulate the lower bounds on area complexities. To do it, we use the general form based on s-communication complexity.

Theorem 3.3.3.4 *Let f be a Boolean function. Then*

$$AC(f) \geq (scc(f) - 1)^2.$$

Proof. Let X be the set of input variables of f, and let $Z \subseteq X$ be such that $\mathrm{scc}(f) = \mathrm{cc}(f, \langle Z \rangle)$. We have to prove that every circuit B computing f has $\mathrm{AC}(B) \geq (\mathrm{cc}(f, \langle Z \rangle) - 1)^2$. We do it by contradiction. Let $S = (V, E)$ be a circuit computing f with $\mathrm{AC}(S) < (\mathrm{cc}(f, \langle Z \rangle) - 1)^2$. This means that there exists a grid-graph $\overline{S}$ of S with $Rect(\overline{S})$ of size $a \times b < (\mathrm{cc}(f, \langle Z \rangle) - 1)^2$. Without loss of generality we assume $a \geq b$, which implies $b < \mathrm{cc}(f, \langle Z \rangle) - 1$. Let $Z' \subseteq V$ denote the set of input nodes corresponding to the variables from Z. Applying Lemma 3.3.3.3 we find a cut-line ω of $Rect(\overline{S})$ such that

(i) the length of ω is at most $b + 1 < \mathrm{cc}(f, \langle Z \rangle)$, and

(ii) the cut $(E(\omega), V_1, V_2)$ of S induced by ω fulfills

$$-1 \leq |V_1 \cap Z'| - |V_2 \cap Z'| \leq 1.$$

Now we describe a communication protocol D computing f with communication complexity at most $b+1$, which will be a contradiction. First, we observe that the partition of input nodes of Z' into $V_1 \cap Z'$ and $V_2 \cap Z'$ corresponds to a balanced partition of the set of input variables X according to Z. Thus, following Figure 3.7 we can consider the part of S left from ω as the first (left) computer of D and the additional part of S (the part right from ω) as the second computer of D. Secondly, the number d of edges of S crossing the cut-line ω is at most $b + 1$, and each edge of S transfers exactly one value during the whole computation on one input assignment.

It is now sufficient to realize that one can unambiguously assign a time-value $t \in \mathbb{N}$ to each edge e of S corresponding to the time in which e transfers a Boolean value. (If this is not clear, consider the following assignment on the nodes of S. Each input node obtains the time-value 1, and each gate obtains the time-value 1+maximum of the values of its input nodes. Each directed edge (u, v) of S gets the time-value of the node u.) Thus, D sends in the first round the Boolean values of all edges leading from the left part of S to the right part of S and having time-values equal to 1. In the second round D does it for the edges crossing ω from right to left and having time-value 1. Generally, the $(2k-1)$-th round and the $2k$-th round of D is used to transfer the Boolean values of edges with the time-value k. The rounds corresponding to an empty set of edges are omitted. The exact formal description of D is left to the reader as an exercise.

So we have constructed a protocol D with $\mathrm{cc}(D) = d \leq b + 1 < \mathrm{cc}(f, \langle Z \rangle)$ which contradicts the existence of S. □

Corollary 3.3.3.5 *Let f be a Boolean function. Then*

$$\mathrm{AC}(f) \geq (\mathrm{cc}(f) - 1)^2.$$

Theorem 3.3.3.6 *Let f be a Boolean function on n variables. Then*

$$\mathrm{bAC}(f) \geq (n + 1) \cdot (\mathrm{scc}(f) - 1)/2.$$

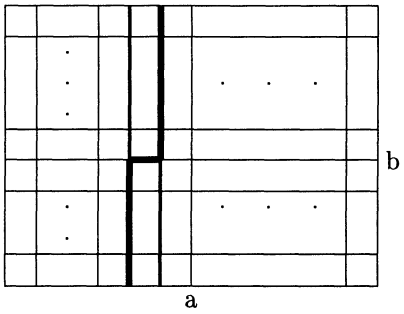

Fig. 3.7. A cut-line of the length b + 1

Proof. Let $X = \{x_1, \ldots, x_n\}$ be the set of input variables of f, and let $Z \subseteq X$ be such that $\mathrm{scc}(f) = \mathrm{cc}(f, \langle Z \rangle)$. Let S be an arbitrary Boolean circuit computing f, and let the size of $Rect(\overline{S})$ be $a \times b$ for some grid-graph $\overline{S}$ of S. Let Z' denote the set of input nodes corresponding to the variables from Z. Without loss of generality, we can assume that $a \geq (n+1)/2$ (at least $(n+1)/4$ input and output nodes are laid on one of the four borders of $Rect(\overline{S})$). Using Lemma 3.3.3.3 we can find a cut-line ω of $Rect(\overline{S})$ with the following properties:

(i) ω is perpendicular to the side of the length a,

(ii) the length of ω is $b+1$, and

(iii) the cut $(E(\omega), V_1, V_2)$ of S induced by ω fulfills

$$-1 \leq |V_1 \cap Z'| - |V_2 - Z'| \leq 1.$$

Similarly as in the proof of Theorem 3.3.3.4 one can prove $b \geq \mathrm{scc}(f) - 1$ which completes the proof because $a \times b \geq (n+1) \cdot (\mathrm{scc}(f) - 1)/2$. □

Corollary 3.3.3.7 *Let f be a Boolean function of n variables. Then*

$$\mathrm{bAC}(f) \geq (n+1) \cdot (\mathrm{cc}(f) - 1)/2.$$

Above we have shown that communication complexity can provide quadratic lower bounds on the area complexity of Boolean circuits. So all functions with linear communication complexity have at least quadratic area complexity. We gave several examples of such Boolean functions in Chapter 1. Here, we present an example of a function having quadratic area complexity and linear communication complexity. So the lower bound provided by communication complexity is asymptotically optimal in this case.

Example 3.3.3.8 We consider the language

$$\begin{aligned} L_{choice} = \{ & wzuv \in \{0,1\}^* \mid |w| = |z| = |v| = |u| = 2m,\ m \in \mathbb{N}, \\ & w = w_1 w_2 \dots w_m,\ z = z_1 z_2 \dots z_m; \\ & \textit{if } u = u_1 1 u_2,\ v = r_1 1 r_2,\ \#_1(u_1) = \#_1(r_1), \\ & |u_1| = j-1 \textit{ and } |r_1| = i-1 \textit{ for some } i,j \in \mathbb{N}, \\ & \textit{then } w_j = z_i \}. \end{aligned}$$

In this language the subwords u and v decide which positions of subwords w and z have to be equal. In Theorem 2.6.3.1 we have shown $\mathrm{scc}(h_n(L_{choice})) \geq n/8$. Thus $\mathrm{AC}(h_n(L_{choice})) = \Omega(n^2)$.

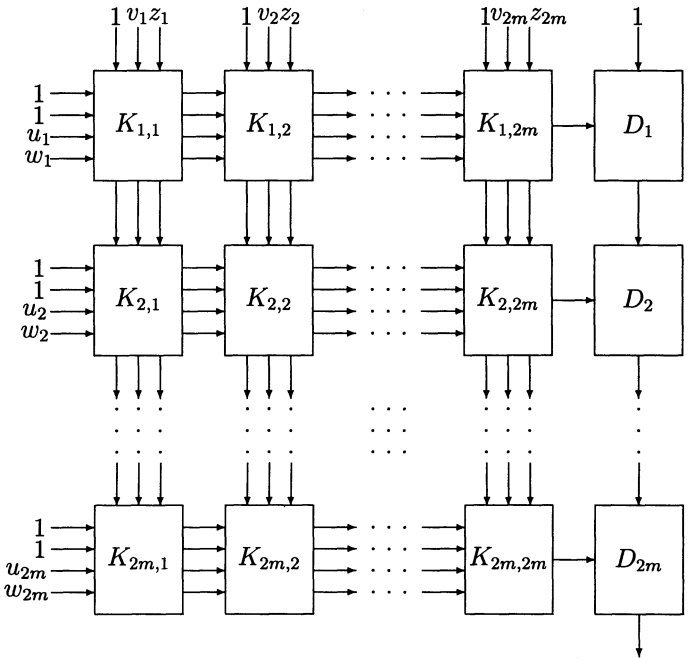

Fig. 3.8. A scheme for the construction of a Boolean circuit computing the Boolean function $h_n(L_{choice})$

Now we show that this quadratic lower bound is asymptotically optimal. Let $n = 8m$. We construct a Boolean circuit S computing $h_n(L_{choice})$ in the following way. S consists of components (small circuits of constant size independent on n), K_{ij} and D_j, for $i,j \in \{1,2,\dots,2m\}$, as depicted in Figure 3.8.

Each component $|K_{ij}|$ has 7 inputs $q_1, q_2, q_3, q_4, q_5, q_6, q_7$ and 7 outputs $p_1, p_2, p_3, p_4, p_5, p_6, p_7$ (see Figure 3.9) that are defined as follows:

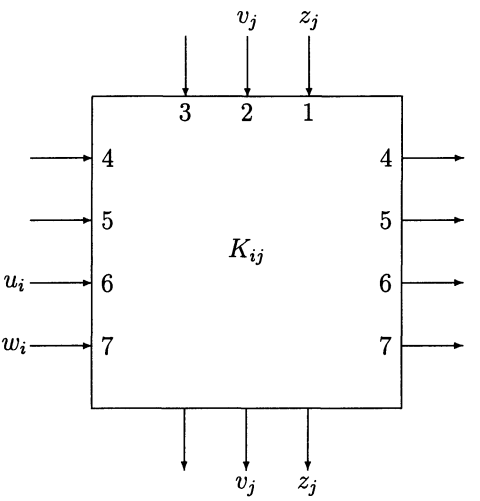

Fig. 3.9. The component K_{ij} of the circuit of Figure 3.8 responsible for the potential comparison of w_i and z_j

$$\begin{aligned} p_i &= q_i \text{ for } i \in \{1,2,6,7\} \\ p_j &= (\overline{q_2} \vee \overline{q_3} \vee \overline{q_4} \vee \overline{q_6}) \wedge q_j \text{ for } j \in \{3,4\}, \\ p_5 &= ((q_1 \equiv q_7) \vee \overline{q_2} \vee \overline{q_3} \vee \overline{q_4} \vee \overline{q_6}) \wedge q_5 \end{aligned}$$

Informally, K_{ij} copies the values $q_1 = z_j$, $q_2 = v_j$, $q_3 = 1$, $q_4 = 1$, $q_5 = 1$, $q_6 = u_i$, $q_7 = w_i$ on the outputs $p_1, p_2, p_3, p_4, p_5, p_6, p_7$, respectively, until the situation does appear in which we have to compare w_i and z_j. This situation appears iff $1 = q_2 = q_3 = q_4 = q_6$. In this case $p_5 = (q_1 \equiv q_7) = (z_j \equiv w_i)$ and $p_3 = p_4 = 0$.

The component D_i, for $i = 1, 2, \ldots, 2n$ computes the conjunction of its inputs. The left input of D_i (see Figure 3.9) is p_5 of the component $K_{i,2m}$.

It can easily be seen that S computes $h_n(L_{choice})$ for $n = 8m$. Since each component $K_{ij}(D_i)$ can be realized by using a constant (independent of n) number of gates, the circuit S has $\text{bAC}(S) \leq d \cdot n^2$ for a constant d. □

3.3.4 A Comparison of two Area Complexity Measures

In the previous subsections the area complexity measures of Boolean circuits ($\text{AC}(f)$ and $\text{bAC}(f)$) were defined and some relations between communication complexity and these measures were established. We observe that for a Boolean function f with a sublinear communication complexity we obtain larger lower bounds for $\text{bAC}(f)$ (namely $n \cdot \text{cc}(f)$) than for $\text{AC}(f)$. A natural question arises: How much may $\text{bAC}(f)$ differ from $\text{AC}(f)$ for a specific function? In this section

we show that for every Boolean function f

$$\mathrm{bAC}(f) \leq 2\mathrm{AC}(f) + 12 \cdot n \cdot \sqrt{\mathrm{AC}(f)}.$$

This result is achieved by giving a general construction bringing the input processors onto the border of the layout. Then we show that this construction is optimal in the sense that there exists a sequence of Boolean functions $\{g_n\}_{n=1}^{\infty}$ with

$$\begin{aligned} \mathrm{AC}(g_n) &= O(n) \quad \text{and} \\ \mathrm{bAC}(g_n) &= \Omega(n^{3/2}). \end{aligned}$$

Lemma 3.3.4.1 *For every Boolean function* $f : \{0,1\}^n \to \{0,1\}$,

$$\mathrm{bAC}(f) \leq 2\mathrm{AC}(f) + 12 \cdot n \cdot \sqrt{\mathrm{AC}(f)} = O(\mathrm{AC}(f) + n\sqrt{\mathrm{AC}(f)}).$$

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be an arbitrary Boolean function of n variables, $n \in \mathbb{N}$, and let S be a Boolean circuit computing f. Let $\overline{S}$ be a grid-graph of S with $Rect(\overline{S})$ of a size $a \times b$, $a \geq b$, $a, b \in \mathbb{N}$. It is sufficient to construct a grid-graph S' of a circuit equivalent to S with all inputs and the output on the border of the layout and with the size of $Rect(\overline{S})$ at most $2ab + 12 \cdot n \cdot b$.

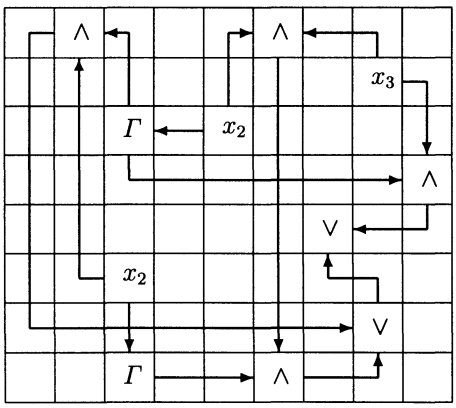

Fig. 3.10. A grid-graph that does not have the inputs on its border

Now we show how to construct S' from $\overline{S}$. Assume $b > 2$ because in the opposite case $\overline{S}$ has already the required properties. Let $r_1, r_2, \ldots, r_a$ denote the rows of $Rect(\overline{S})$ from down to up and $s_1, s_2, \ldots, s_b$ denote the columns of $Rect(\overline{S})$ from the left to the right. We add two new rows r_{-1} and r_0 at the bottom of $Rect(\overline{S})$. The idea is to lay all inputs and the output in the row

r_{-1}. Now we sequentially lay the inputs one after the other in the row r_{-1}. Let x be an input which still does not lie in r_{-1}. We distinguish two possibilities according to the position of x in the lattice. Let x lie on the intersection of a row r_j and a column s_i for some $j \in \{1, \ldots, a\}$, $i \in \{1, \ldots, b\}$.

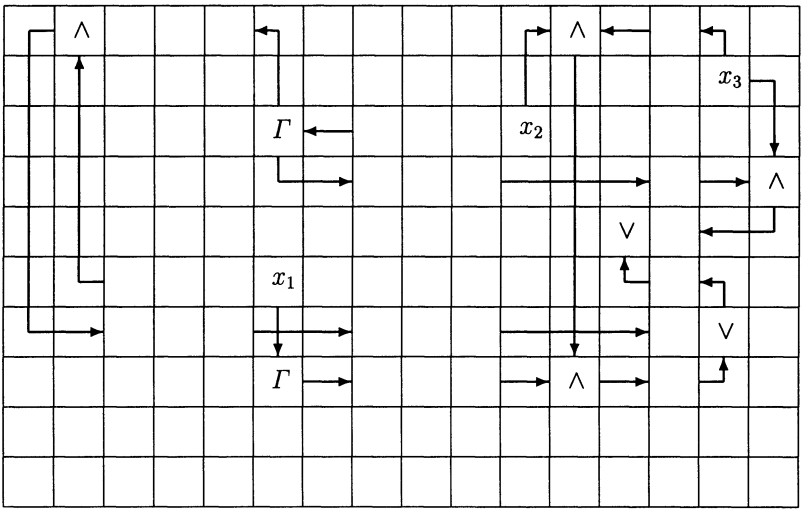

Fig. 3.11. Inserting columns and rows in the grid-graph of Figure 3.10

(i) Let x be the only input lying in the column s_i and let there be no edge horizontally outcoming from x to the left in the row r_j. Then, we insert one new column s_{x1} between the column s_{i-1} and the column s_i into the lattice (see Figures 3.10 and 3.11 for the variable x_3). After this we reorganize the circuit and its layout as follows (see Figure 3.12 for x_3):

a. we lay the input x on the intersection of r_{-1} and s_{x1},

b. we put a gate computing the identity function on the intersection of r_j and s_i instead of x,

c. we lay an edge leading from x on the intersection of r_{-1} and s_{x1} to the identity gate on the intersection of r_j and s_i, and finally

d. we horizontally connect all horizontal links (edges) cut by inserting the column s_{x1} into the lattice.

Obviously, the new circuit computes the same function f as the previous one and the lattice has grown about $a + 2$ squares in this construction.

(ii) Let x be not the only input in the column s_i or let there be an edge horizontally leading from the left side of the square containing x in the

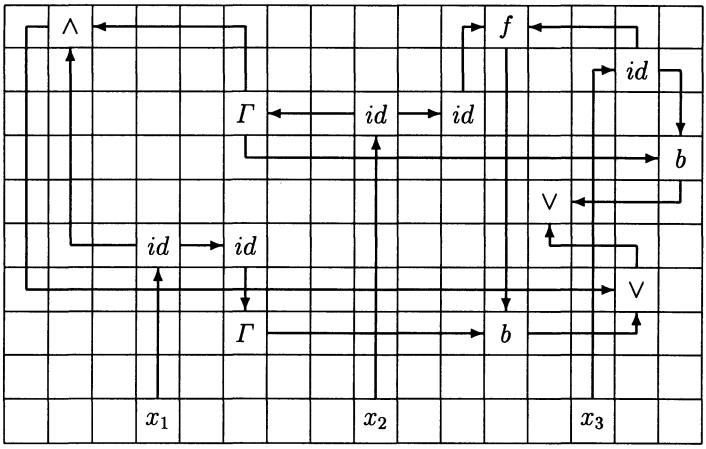

Fig. 3.12. A grid-graph equivalent to the grid-graph of Figure 3.10 and having all inputs on its border

row r_j. Then (for each x in the column) we insert three new columns s_{x1}, s_{x2}, s_{x3} between the column s_{i-1} and the column s_i into the lattice (see Figures 3.10 and 3.11 for the variables x_1 and x_2). After this we make the following changes (see Figure 3.12 for the variables x_1 and x_2):

a. we lay the input x on the intersection of r_{-1} and s_{x2},

b. we put a gate computing the identity function on the interconnection of r_j and s_i instead of x,

c. we put a gate computing the identity function on the interconnection of r_j and s_{x2},

d. we lay an edge vertically leading from x on the intersection of r_{-1} and s_{x2} to the identity gate on the intersection of r_j and s_{x2},

e. we lay an edge horizontally leading from the identity gate on the intersection of r_j and s_{x2} to the identity gate on the intersection of r_j and s_i,

f. if there was an edge horizontally leading from the left side of the square (originally containing x) on the intersection of r_j and s_i, then we add an edge horizontally leading from the right to the left into the square on the intersection of r_j and s_{x1}, and finally

g. we horizontally connect all horizontal links (edges) cut by inserting the columns s_{x1}, s_{x2}, s_{x3} into the lattice.

Obviously, the new circuit computes the same function f as the previous one and the lattice has grown about $3 \cdot (a+2)$ squares in this construction.

Now it remains to bring the output node y into the row x_{-1}. This can be done in almost the same way as for the input in case 2. The only two differences are:

a. we leave the original output gate in position and lay the new output gate computing the identity function on the intersection of the row r_{-1} and the column s_{y2},

b. the edge leading between these two nodes has the same layout as in the input case above but the opposite direction (from the original output gate to the new output gate).

We observe that the new grid-graph S' of the new circuit equivalent to S has $Rect(S')$ of size at most $(a+3(n+1))\cdot(b+2) \leq ab+2a+3(n+1)\cdot(b+2) \leq 2\cdot ab+12nb$. □

To show that the construction of Lemma 3.3.4.1 cannot be asymptotically improved we consider the Boolean functions

$$g_n(x_{1,1}, x_{1,2}, \ldots, x_{1,m}, x_{2,1}, x_{2,2}, \ldots, x_{2,m}, \ldots, x_{m,1}, x_{m,2}, \ldots, x_{m,m}) =$$
$$\left(\bigwedge_{i=1}^{m}(x_{i,1} \equiv x_{i,2} \equiv \ldots \equiv x_{i,m})\right) \vee \left(\bigwedge_{j=1}^{m}(x_{1,j} \equiv x_{2,j} \equiv \ldots \equiv x_{m,j})\right)$$

for any $n = m^2$, $m \in \mathbb{N}$. Note that g_n is very similar to the function $h_n(R_0 \cup C_0)$ considered in Theorem 2.4.4.5.

Lemma 3.3.4.2 *For any $n = m^2$, $m = 2k$, $k \in \mathbb{N}$:*

(i) $AC(g_n) \leq 36n$, *and*

(ii) $bAC(g_n) \geq (n^{3/2} - 2n + n^{1/2} - 2)/8$.

Proof. First we prove the upper bound (i). The scheme in Figure 3.13 shows how to construct a Boolean circuit S computing g_n with $Rect(S)$ of size $6n \times 6n$ (to get a precise layout from this scheme one has to insert one column between any two columns and one row between any two rows of this scheme, and to connect the processors in an appropriate way).

Now we prove the lower bound (ii). First we show that $cc(g_n) \geq m/2$ for any $n = m^2$, $m \in \mathbb{N}$. Let $X = \cup_{i=1}^{m} R_i = \cup_{j=1}^{m} C_j$, where $R_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$ for $j = 1, \ldots, m$ and $C_j = \{x_{1,j}, x_{2,j}, \ldots, x_{m,j}\}$ for $j = 1, \ldots, m$. We call R_i the i-th row of X and we call C_j the j-th column of X for any $i, j \in \{1, \ldots, m\}$. Let Π be an arbitrary balanced partition of X. We distinguish the following three possibilities according to Π:

(1) $\exists r, s \in \{1, \ldots, m\}$ such that $R_r \subseteq \Pi_L$ and $R_s \subseteq \Pi_R$ (each of the two computers contains at least one complete row of X)

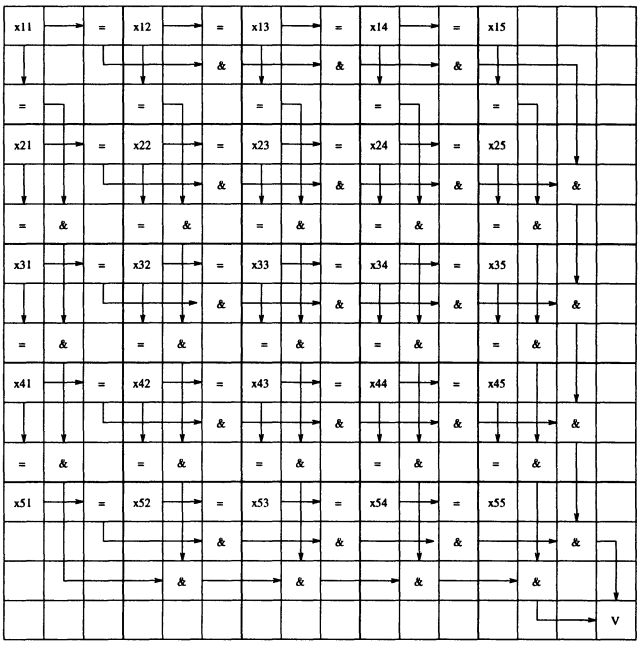

Fig. 3.13. A schema of a grid-graph computing the Boolean function g_n

(2) $\forall i \in \{1, \ldots, m\}$: $R_i \cap \Pi_L \neq \emptyset$ (the first computer has at least one variable from each row of X)

(3) $\forall j \in \{1, \ldots, m\}$: $R_j \cap \Pi_R \neq \emptyset$ (the second computer has at least one variable from each row of X)

Obviously, the cases (1), (2), and (3) cover all possible partitions from Bal(X). Now we deal with (1), (2), (3) separately.

1. (1) implies that, for every $k \in \{1, \ldots, m\}$, $C_k \cap \Pi_L \neq \emptyset$ and $C_k \cap \Pi_R \neq \emptyset$, i.e., the variables of each column are divided between Π_L and Π_R. We define

$$\mathcal{A}(\Pi) = \{\omega : X \to \{0,1\} \mid \omega(x_{1,j}) = \omega(x_{2,j}) = \ldots = \omega(x_{m,j}) \text{ for all } j = 1, \ldots, m, \text{ and } \omega \neq 1^n, \ \omega \neq 0^n\}$$

We observe that $\mathcal{A}(\Pi)$ is a 1-fooling set for g_n and Π because for every $\alpha = \alpha_{1,1}, \alpha_{1,2}, \ldots, \alpha_{1,m}, \ldots, \alpha_{m,1}, \alpha_{m,2}, \ldots, \alpha_{m,m} \in \mathcal{A}(\Pi)$, $\wedge_{i=1}^{m}(\alpha_{i,1} \equiv \alpha_{i,2} \equiv \ldots \equiv \alpha_{i,m}) = 0$, $\wedge_{j=1}^{m}(\alpha_{1,j} \equiv \alpha_{2,j} \equiv \ldots \equiv \alpha_{m,j}) = 1$, and the matrix $M(g_n', \Pi)$ for $g_n'(x_{1,1}, \ldots, x_{m,m}) = \wedge_{j=1}^{m}(x_{1,j} \equiv x_{2,j} \equiv \ldots \equiv x_{m,j})$ contains exactly 2^m ones corresponding to the $2^m - 2$ inputs in $\mathcal{A}(\Pi)$, and to the inputs 1^n and 0^n. Thus for all $\Pi \in$ Bal(X) with the property (1) $cc(g_n, \Pi) \geq \lceil log_2 \mid \mathcal{A}(\Pi) \mid \rceil \geq m$ for $m \geq 3$.

2. (2) implies that Π_L contains at least one variable from each row (i.e., there is no row completely contained in Π_R). Since Π is balanced, there exists a set of positive integers $S_\Pi = \{i_1, i_2, \ldots, i_d\} \subseteq \{1, \ldots, m\}$ such that $d > m/2$, and, for every $l \in S_\Pi$, $R_l \cap \Pi_R \neq \emptyset$. We observe that

$$\begin{aligned}\tilde{\mathcal{A}}(\Pi) = \; & \{\omega \; : \; X \to \{0,1\} \mid \omega(x_{l,1}) = \omega(x_{l,2}) = \ldots = \omega(x_{l,m}) \\ & \text{for every } l \in S_\Pi, \; \omega \neq 1^n, \; \omega \neq 0^n \text{ and} \\ & w(x_{j,1}) = w(x_{j,2}) = \ldots = w(x_{j,m}) = 1 \\ & \text{for every } j \in \{1, 2, \ldots, m\} - S_\Pi\}\end{aligned}$$

is a 1-fooling set for g_n and Π. Since $|\tilde{\mathcal{A}}(\Pi)| \geq 2^d - 2$ we obtain $\mathrm{cc}(g_n, \Pi) \geq \lceil log_2(2^{m/2} - 2) \rceil \geq m/2$ for $m \geq 3$ and $\Pi \in \mathrm{Bal}(X)$ with the property (2).

3. The property (3) is symmetric to the property (2) and so this case can be solved in the same way as the second one.

Thus we have proved $\mathrm{cc}(g_n, \Pi) \geq m/2$ for every $\Pi \in \mathrm{Bal}(X)$ which implies $\mathrm{cc}(g_n) \geq m/2$. Following Theorem 3.3.3.6 we obtain

$$\mathrm{bAC}(g_n) \geq (n+1) \cdot (m/2 - 1)/4 = (n^{3/2} - 2n + n^{1/2} - 2)/8.$$

□

3.3.5 Three-Dimensional Layout

In this section we consider the layout of Boolean circuits in a three-dimensional lattice. The results of this section are more of theoretical than of practical interest because all three-dimensional layouts in practice have the size of the third dimension bounded by a constant. Thus, the assertions above for the two-dimensional layout are more closely related to the three-dimensional layout with a bound on the size of the third dimension than the assertions of this section.

The aim of this section is to show which lower bounds are provided for the space complexity of Boolean circuits by our communication complexity approach. The assertions presented here are straightforward extensions of the planar case. We start with the definitions of the three-dimensional layout and of the space complexity of Boolean functions.

Definition 3.3.5.1 *Let $G = (V, E)$ be a directed graph with degree bounded by 6. A* **three-dimensional grid-graph $\overline{G}$ of G** *is a layout of G in the three-dimensional lattice with the following properties:*

(i) The degree of $\overline{G}$ is bounded by 6.

(ii) Each cube of the three-dimensional lattice has some of the following contents:

(a) a vertex of V;

(b) *one broken line entering the cube through a wall and exiting through one of the neighboring walls;*

(c) *at most three lines, each entering in the cube in a direction that is perpendicular to the input directions of other lines, and exiting through the non-neighboring wall to its input wall;*

(d) *the empty content.*

(iii) *If one cube of the lattice contains a node of V, then none of the 26 neighboring cubes of this cube contains a node of V.*

The **space complexity of $\overline{G}$, $S(\overline{G})$,** *is the area of the minimal rectangular parallelepiped $Rect_3(\overline{G})$ containing all nonempty squares of the three-dimensional lattice.*

Definition 3.3.5.2 *Let $G = (V, E)$ be a directed graph with degree bounded by 4. The* **three-dimensional layout space of G** *is*

$$\mathbf{S}(G) = \min\{S(\overline{G}) \mid \overline{G} \text{ is a three-dimensional grid-graph of } G\}.$$

Let B be a Boolean circuit with both indegree and outdegree bounded by 2. The **space complexity of B** *is* $\mathbf{SC}(B) = S(B)$. *For any Boolean function f, the* **space complexity of f** *is*

$$\mathbf{SC}(f) = \min\{SC(B) \mid B \text{ computes } f\}.$$

Now we give analogous definitions for the three-dimensional layout with the input and output nodes on the outside walls of the layout.

Definition 3.3.5.3 *Let $G = (V, E)$ be a directed graph with degree bounded by 4, and let U be a subset of V. The* **three-dimensional b-layout space of G according to U** *is*

$$\mathbf{bS}(G, U) = \min\{S(\overline{G}) \mid \overline{G} \text{ is a three-dimensional grid-graph of } G \text{ which contains the nodes of } U \text{ on the six outside walls of } Rect_3(\overline{G})\}.$$

Let B be a Boolean circuit with both indegree and outdegree bounded by 2, with the set of input nodes X, and with the set of output nodes Y. The **b-space complexity of B** *is* $\mathbf{bSC}(B) = bS(B)$. *For any Boolean function f, the* **b-space complexity of f** *is*

$$\mathbf{bSC}(f) = \min\{bSC(B) \mid B \text{ computes } f\}.$$

Observation 3.3.5.4 *For any Boolean function f*

(i) $SC(f) \leq bSC(f)$

(ii) $\mathrm{SC}(f) \leq \mathrm{AC}(f)$, *and*

(iii) $\mathrm{bSC}(f) \leq \mathrm{bAC}(f)$.

Now we show some relations between the communication complexity of a Boolean function f and the space complexity measures of f.

Theorem 3.3.5.5 *For any Boolean function f*

$$\mathrm{SC}(f) \geq (\mathrm{scc}(f))^{3/2}/\sqrt{8}.$$

Proof. Let X be the set of input variables of f, and let $Y \subseteq X$ be such that $\mathrm{scc}(f) = \mathrm{cc}(f, \langle Y \rangle)$. Let R be a Boolean circuit computing f, and let $\overline{R}$ be a three-dimensional grid-graph of R with $Rect_3(\overline{R})$ of size $a \times b \times c$. Now it is sufficient to show that

$$(a \cdot b \cdot c)^2 \geq (\mathrm{scc}(f))^3/8.$$

Let W of dimensions $u \times v$, $u, v \geq 2$ be an arbitrary wall of $Rect_3(\overline{R})$. It is easy to find a plane L with at most one stair (in the same way as a line with a single jog was found in Lemma 3.3.3.3 for the two-dimensional case) such that

(i) L is parallel to W,

(ii) the area of L is at most $uv + u + v \leq (u+1) \cdot (v+1) \leq 2uv$ ($u + v$ is for the part of the stair perpendicular to W),

(iii) L defines a cut of R into two parts with the property that at least $\lfloor |Y|/2 \rfloor$ input variables from Y enter each of these two parts.

Consequently, $2uv \geq \mathrm{cc}(f, \langle Y \rangle) = \mathrm{scc}(f)$. Since we have obtained this inequality for arbitrary sizes u, v of P_R, we have

(1) $2ab \geq \mathrm{scc}(f)$,

(2) $2ac \geq \mathrm{scc}(f)$,

(3) $2bc \geq \mathrm{scc}(f)$.

Multiplying expressions (1), (2), and (3), we obtain

$$8a^2b^2c^2 \geq 8(abc)^2 \geq (\mathrm{scc}(f))^3$$

which completes the proof of Theorem 3.3.5.5. □

Theorem 3.3.5.6 *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function of n variables, $n \in \mathbb{N}$. Then*

$$\mathrm{bSC}(f) \geq \sqrt{n+1} \cdot \mathrm{scc}(f)/\sqrt{24}.$$

Proof. Let X be the set of input variables of f, and let $Y \subseteq X$ be such that $\mathrm{scc}(f) = \mathrm{cc}(f, \langle Y \rangle)$. Let R be an arbitrary Boolean circuit computing f, and let the size of $Rect_3(\overline{S})$ be $a \times b \times c$, $a \geq b \geq c$, for some three-dimensional grid-graph $\overline{S}$ of S. It suffices to show $a^2b^2c^2 \geq (n+1) \cdot (\mathrm{scc}(f))^2/24$. Since $\overline{S}$ can have at most $2ab + 2ac + 2bc \leq 6ab$ input and output vertices on its walls, we obtain

(4) $6ab \geq n + 1$.

Obviously, (2) and (3) from the proof of Theorem 3.3.5.5 hold. Multiplying expressions (2), (3), and (4), we obtain

$$24a^2b^2c^2 \geq (n+1) \cdot (\mathrm{cc}(f, \langle Y \rangle))^2 = (n+1) \cdot (\mathrm{scc}(f))^2.$$

□

Corollary 3.3.5.7 *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function of n variables, $n \in \mathbb{N}$. Then*

(i) $\mathrm{SC}(f) \geq (\mathrm{cc}(f))^{3/2}/\sqrt{8}$, *and*

(ii) $\mathrm{bSC}(f) \geq \sqrt{n+1} \cdot \mathrm{cc}(f)/\sqrt{24}$.

Thus, using communication complexity one can obtain $\Omega(n^{3/2})$ lower bounds on the space complexity of Boolean functions.

Corollary 3.3.5.8 *Let $f = \{f_i\}_{i=1}^{\infty}$ be a sequence of Boolean functions, where $f_i : \{0,1\}^i \to \{0,1\}$. If $\mathrm{scc}(f_n) = \Omega(n)$, then*

$$\mathrm{SC}(f_n) = \Omega(n^{3/2}).$$

3.3.6 Exercises

Exercise 3.3.6.1 * *Let T_k^2 be a complete binary tree of depth k. Estimate $\mathrm{A}(T_k^2)$ for any $k \in \mathbb{N}$.*

Exercise 3.3.6.2 * *Let $T_k^2 = (V_k, E_k)$ be a complete binary tree of depth k. Let $U_k \subseteq V_k$ be the set of all leaves of T_k^2. Estimate $\mathrm{bA}(T_k^2, U_k)$ for any $n \in \mathbb{N}$.*

Exercise 3.3.6.3 * *Prove that each planar graph of r nodes with the degree at most 4 can be laid out in a lattice of area $O(r \cdot (\log r)^2)$.*

Exercise 3.3.6.4 *Prove an extended version of Theorem 3.3.3.4 considering computing problems instead of Boolean functions.*

Exercise 3.3.6.5 *Prove the following extension of Theorem 3.3.3.6: "For every computing problem P_n^m of n inputs and m outputs, $n, m \in \mathbb{N}$,*

$$\mathrm{bAC}(P_n^r) \geq (n+m) \cdot (\mathrm{scc}(P_n^r) - 1)/4."$$

Exercise 3.3.6.6 *Improve the constants in the assertion of Lemma 3.3.4.1.*

Exercise 3.3.6.7 *Use the scheme in Figure 3.13 to give a detailed description of the layout of the circuit S computing g_n with $\mathrm{A}(S) \leq 36n^2$.*

Exercise 3.3.6.8 *Give a precise formal description of the protocol D simulating the communication flowing between the two parts of the circuit S given by the cut $(E(\omega), V_1, V_2)$ in the proof of Theorem 3.3.3.4.*

Exercise 3.3.6.9 *Let f be a Boolean function, and let X be the set of input variables of f. Let $S = (V, E)$ be a Boolean circuit of depth k computing f, and let (E', V_1, V_2) be a cut of S with the property $-1 \leq |V_1 \cap X| - |V_2 \cap X| \leq 1$. Prove that there exists a $(k+1)$-rounds protocol D computing f within the communication complexity $|E'|$. Note that the protocol from the proof of Theorem 3.3.3.4 uses $2k$ rounds.*

Exercise 3.3.6.10 *Give a precise description of the Boolean circuits and their grid-graphs corresponding to the components K_{ij} and D_j for $i, j \in \{1, 2, \ldots, 2m\}$ from Example 3.3.3.8.*

Exercise 3.3.6.11 *Let $T_k^2 = (V_k, E_k)$ be a complete binary tree of depth k. Let $U_k \subset V_k$ be the set of all leaves of T_k^2. Estimate $\mathrm{bS}(T_k^2, U_k)$ and $\mathrm{S}(T_k^2)$ for any $k \in \mathbb{N}$.*

Exercise 3.3.6.12 * *We define the Shannon's functions of the area complexity of Boolean functions as* $\mathbf{ShAC(}\boldsymbol{n}\mathbf{)} = \max\{\mathrm{AC}(f) \mid f \in B_2^n\}$ *and* $\mathbf{ShbAC(}\boldsymbol{n}\mathbf{)} = \max\{\mathrm{bAC}(f) \mid f \in B_2^n\}$. *Give some estimate of* $\mathrm{ShAC}(n)$ *and* $\mathrm{ShbAC}(n)$.

Exercise 3.3.6.13 * *We define the Shannon's function of the space complexity of Boolean functions as* $\mathbf{ShSC(}\boldsymbol{n}\mathbf{)} = \max\{\mathrm{SC}(f) \mid f \in B_2^n\}$ *and* $\mathbf{ShbSC(}\boldsymbol{n}\mathbf{)} = \max\{\mathrm{bSC}(f) \mid f \in B_2^n\}$. *Give some estimate of* $\mathrm{ShSC}(n)$ *and* $\mathrm{ShbSC}(n)$.

Exercise 3.3.6.14 * *Prove* $\mathrm{bAC}(P_n^{2^n}) = \Omega(n2^n)$ *and* $\mathrm{bAC}(P_n^{2^n}) = O(n2^n)$, *where $P_n^{2^n}$ is the computing problem containing all 2^n elementary conjunctions of n variables.*

Exercise 3.3.6.15 * *Prove* $\mathrm{bAC}(\overline{P}_n^m) = \Omega(n2^{2^n})$ *and* $\mathrm{bAC}(\overline{P}_n^m) = O(n2^{2^n})$ *for $m = 2^{2^n}$, where $\overline{P}_n^m$ is the computing problem containing all 2^{2^n} Boolean functions of n variables.*

Exercise 3.3.6.16 * *Prove* $\mathrm{bAC}(P_{2n}^{2n+1}) = \Omega(n^2)$ *and* $\mathrm{bAC}(P_{2n}^{2n+1}) = O(n^2)$, *where* P_{2n}^{2n+1} *is the computing problem corresponding to the multiplication of two binary integers of length* n.

Exercise 3.3.6.17 *Prove, for some sequence* $\{f_n\}_{n=1}^{\infty}$ *of symmetric Boolean functions* $f_n : \{0,1\}^n \to \{0,1\}$, $\mathrm{bAC}(f_n) = O(n \log n)$ *and* $\mathrm{bAC}(f_n) = \Omega(n \log n)$.

Exercise 3.3.6.18 ** *Prove, that for any directed graph* $G = (V, E)$ *of degree bounded by 4,* $\mathrm{A}(G) \leq c \cdot (\mathrm{S}(G))^{3/2}$ *for some constant* c *independent on* G. *Show the optimality of this simulation result by finding a sequence* $\{f_n\}_{n=1}^{\infty}$ *of Boolean functions such that* $f_n \in B_2^n$ *and* $\mathrm{bAC}(f_n) = \Omega((\mathrm{bSC}(f_n))^{3/2})$.

Exercise 3.3.6.19 *Prove for any Boolean function* $f : \{0,1\}^n \to \{0,1\}$,

$$\mathrm{SC}(f) \geq (\mathrm{scc}(f) - 1)^{4/3}$$

and

$$\mathrm{bSC}(f) = \Omega((n+1)^{2/3} \cdot (\mathrm{scc}(f))^{2/3}).$$

[Hint: Exercise 3.3.6.18 provides $\mathrm{AC}(f) \leq c(\mathrm{SC}(f))^{3/2}$, *and also* $\mathrm{bAC}(f) \leq c(\mathrm{bSC}(f))^{3/2}$ *for any Boolean function* f*].*

3.3.7 Problems

Problem 3.3.7.1 ** *Prove an asymptotically higher lower bound than* $\Omega(n^2)$ *on* $\mathrm{AC}(f_n)$ *or* $\mathrm{bAC}(f_n)$ *for a sequence of Boolean functions* $\{f_n\}_{n=1}^{\infty}$. *Note that according to the assertion of Observation 3.3.2.3 such a super-quadratic lower bound implies a nonlinear lower bound on* $\mathrm{CC}(f_n)$.

Problem 3.3.7.2 * *Use* k*-rounds communication complexity to find a Boolean function* f *with the following properties:*

(i) the area complexity of any circuit of depth k *computing* f *is 'large', and*

(ii) there exists a circuit of depth $k+1$ *computing* f *in a 'small' area.*

[Hint: Consider Exercise 3.3.6.9 to get a lower bound technique for results of kind (i)].

Problem 3.3.7.3 * *Consider Problem 3.3.7.2 for the space complexity (three-dimensional layout).*

Problem 3.3.7.4 *Either find a Boolean function* f *with a large difference between* $\mathrm{SC}(f)$ *and* $\mathrm{bSC}(f)$ *or prove that such a Boolean function does not exist. A general strategy getting the input vertices on the walls of the three-dimensional*

layout, and a concrete sequence of Boolean functions proving the optimality of this general strategy would give a complete solution of this problem. Note that Section 3.3.4 provides such a solution for the two-dimensional layout.

Problem 3.3.7.5 * *Theorem 3.3.5.6 and Exercise 3.3.6.19 provide*

$$\mathrm{bSC}(f) \geq \max\{\sqrt{n+1} \cdot scc(f)/\sqrt{24},\ (n+1)^{2/3}(\mathrm{scc}(f))^{2/3}/d\}$$

for any Boolean function $f : \{0,1\}^n \to \{0,1\}$, where d is a constant independent of f and n. This means that there may exist functions for which Theorem 3.3.5.6 provides better lower bounds than Exercise 3.3.6.19, and conversively there may exist functions for which Exercise 3.3.6.19 provides better lower bounds than Theorem 3.3.5.6. Try to find a lower bound technique on the complexity measure $\mathrm{bSC}(f)$ *which generalizes the approaches compared above.*

3.4 Topology of Circuits and Lower Bounds

3.4.1 Introduction

In this section we extend the idea of Section 3.3 by proving a nonlinear lower bound on the number of gates of Boolean circuits without any extremely dense connection between two arbitrary parts of the circuit. Section 3.4 is organized as follows. Section 3.4.2 gives the graph-theoretical basis formalizing the notion of "extreme density" involved here. In Section 3.4.3 we show that the communication complexity of f provides a direct lower bound on some tradeoff between the density of the circuit computing f and the number of gates of this circuit. In Section 3.4.4 we show the existence of Boolean circuits with such high density that the communication complexity approach cannot help to prove nonlinear lower bounds on their size. Finally, in Section 3.4.5 we use the lower bound proof technique of Section 3.4.3 to prove $\Omega(n^2)$ lower bounds on the combinational complexity of planar Boolean circuits computing specific Boolean functions.

3.4.2 Separators

In this section we define some graph-theoretical (topological) restrictions on Boolean circuits in order to be able to prove nonlinear lower bounds on the combinational complexity of such restricted circuits. We start with separators enabling some recursive partitioning of graphs. Note that the following definitions have the same meaning independently of whether directed graphs or undirected graphs are considered.

Definition 3.4.2.1 *Let $G = (V, E,)$ be a directed graph of n nodes, $n \in \mathbb{N}$. Let $s : \mathbb{N} \to \mathbb{N}$ be a function. We say that G has an* $\boldsymbol{s(n)}$ **vertex-separator** *if either it has only one node, or one can find a set $V' \subseteq V$, $|V'| \leq s(|V|)$, such that there exist $V_1, V_2 \subseteq V$ with the following properties:*

(i) $V' \cup V_1 \cup V_2 = V,\ V' \cap V_1 = \emptyset,\ V_1 \cap V_2 = \emptyset,\ V' \cap V_2 = \emptyset,$

(ii) $E \subseteq V_1 \times V_1 \cup V_2 \times V_2 \cup V_1 \times V' \cup V' \times V_1 \cup V' \times V' \cup V' \times V_2 \cup V_2 \times V'$ *(i.e., the set of edges adjacent to the nodes in* V' *involves a cut of* G*),*

(iii) $|V' \cup V_1| \geq |V|/3$ *and* $|V' \cup V_2| \geq |V|/3,$

(iv) the directed graphs $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *with* $E_i = E \cap (V_i \times V_i)$ *for* $i = 1, 2$ *have an* $s(n)$ *vertex-separator.*

We say that G *has a* **strong $s(n)$ vertex-separator** *if* G *has only one node, or one can find a set* $V' \subseteq V$ *with the properties (i), (ii), and*

(v) $|V' \cup V_1| \geq \lfloor |V|/2 \rfloor$ *and* $|V' \cup V_2| \geq \lfloor |V|/2 \rfloor,$

(vi) the directed graphs $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *with* $E_i = E \cap (V_i \times V_i)$ *for* $i = 1, 2$ *have an strong* $s(n)$ *vertex-separator.*

We observe that an $s(n)$ vertex-separator of a graph G enables us by an appropriate distribution of V' between V_1 and V_2 to partition G into two graphs G_1' and G_2', each of size at least one third of G. If this vertex-separator is strong, then we can obtain G_1' and G_2' of sizes differing at most by 1.

Now we define edge-separators.

Definition 3.4.2.2 *Let* $G = (V, E)$ *be a directed graph of* n *nodes,* $n \in \mathbb{N}$*. Let* $s : \mathbb{N} \to \mathbb{N}$ *be a function. We say that* G *has an* **$s(n)$ edge-separator** *if either it has only one node, or one can find a cut* (E', V_1, V_2) *of* G *such that*

(i) $|E'| \leq s(|V|),$

(ii) $|V_i| \geq |V|/3$ *for* $i = 1, 2,$ *and*

(iii) the directed graphs $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *with* $E_i = E \cap (V_i \times V_i)$ *for* $i = 1, 2$ *have an* $s(n)$ *edge-separator.*

We say that G *has a* **strong $s(n)$ edge-separator** *if either it has only one node, or one can find a cut* (E', V_1, V_2) *of* G *such that (i), and*

(iv) $|V_i| \geq \lfloor |V|/2 \rfloor$ *for* $i = 1, 2,$

(v) the directed graphs $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *with* $E_i = E \cap (V_i \times V_i)$ *for* $i = 1, 2$ *have a strong* $s(n)$ *edge-separator*

hold.

We say that a cut (E', V_1, V_2) *of* G *is an* **$s(n)$ bisection** *of* G *if (i) and (ii) hold. We say that a cut* (E', V_1, V_2) *of* G *is a* **strong $s(n)$ bisection** *of* G *if (i) and (iv) hold.*

Observation 3.4.2.3 *If* G *has a (strong)* $s(n)$ *edge-separator for some* $s : \mathbb{N} \to \mathbb{N}$*, then* G *has a (strong)* $s(n)$ *bisection.*

Now we give two examples illustrating separators for two special classes of graphs.

Example 3.4.2.4 We consider the family $\{G_{m\times m}\}_{m=1}^{\infty}$ of two-dimensional grids (lattices) $G_{m\times m}$ of size $m \times m$ (see Figure 3.14 for the $G_{10\times 10}$). We observe that $G_{m\times m}$ of $n = m^2$ nodes has a strong $\sqrt{n} = m$ edge-separator if $m = 2^k$ for some $k \in \mathbb{N}$, and a strong $\sqrt{n} + 1$ edge-separator for other m. To see this in Figure 3.14, we first divide $G_{m\times m}$ in the middle in order to get two $G_{m\times m/2}$ grids. $G_{m\times m/2}$ can be divided into two $G_{m/2\times m/2}$ by removing $m/2 \leq \sqrt{m^2/2}$ nodes. So, after two recursive separations we again get squared grids $G_{m/2\times m/2}$ and the recursive separation can continue as described above. One can easily observe that the bisection of $G_{m\times m}$ is at least $\sqrt{n} = m$, and so, following Observation 3.4.2.3, each $s(n)$ edge-separator of $G_{m\times m}$ fulfills $s(n) \geq \sqrt{n}$. □

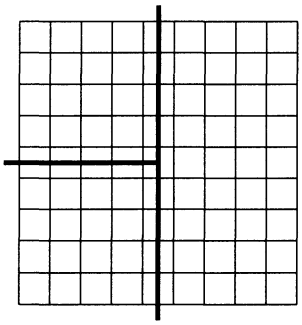

Fig. 3.14. The application of an edge-separator for the grid of size 10×10

Example 3.4.2.5 We show that any binary tree (acyclic graph of degree at most 3) has a 1 edge-separator.

Let $T = (V, E)$ be a binary tree of n nodes, $n \geq 2$. Obviously, each edge of E is a cut of T. The following procedure will find an edge representing a cut fulfilling the property (ii) of Definition 3.4.2.2. At the beginning the procedure picks an arbitrary edge $(u, v) \in E$. This divides T into T_u and T_v, where T_u is the tree with the root u and T_v is the tree with the root v. If both T_u and T_v have at least $n/3$ nodes each, then we are ready. Without loss of generality we assume T_u has more than $\lfloor 2n/3 \rfloor$ nodes. If u has degree one in T_u then the procedure picks the only edge (u, r) adjacent to u in T_u and considers this edge (u, r) instead of (u, v) as a cut of T. If (u, r) and (u, s) are the two edges adjacent to u in T_u and the subtree T_r with the root r has at least as many nodes as the subtree T_s with the root s, then the procedure considers (u, r) instead of (u, v) as a cut of T. T_r has at least $\lfloor n/3 \rfloor$ nodes because T_u has at least $\lceil 2n/3 \rceil$ nodes,

and T_u is the largest subtree of the two subtrees of T_u. Thus, if T_r has fewer than $\lceil 2n/3 \rceil$ nodes, $\{(u,r)\}$ is a cut of T with the required properties. If T_r has more than $2n/3$ nodes the procedure continues as described above.

This procedure must finish with a cut having the required properties because the largest part of the two parts of T is always decreased in one step (the change of the candidating edge) but never decreased by more than $n/3$ nodes (i.e., the largest part is never exchanged by a part with fewer than $\lfloor n/3 \rfloor$ nodes).

We illustrate this procedure on the tree depicted in Figure 3.15. Let us pick the edge (a,b) as the first candidate for the cut. Then the tree T_a contains two nodes and the tree T_b contains 9 nodes. Obviously, this cut $\{(a,b)\}$ is not balanced enough. Since b has only one son d in T_b we consider the edge (d,b) as the candidate for a cut of T. Now, T_d contains 8 nodes, which is still too much. The subtree T_e of T_d has two nodes and the subtree T_f of T_d has five nodes. According to the procedure described above we pick the edge (d,f). Since T_f has 5 nodes we have found a bisection $\{d,f\}$ of T. □

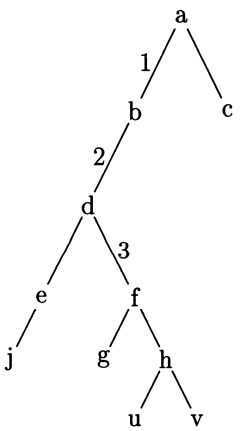

Fig. 3.15. The search for the bisection of a binary tree

Observation 3.4.2.6 *Let G be a directed graph of degree bounded by a constant $d \in \mathbb{N}$. Let $s : \mathbb{N} \to \mathbb{N}$ be a function. If G has a (strong) $s(n)$ vertex-separator, then G has a (strong) $d \cdot s(n)$ edge-separator.*

Following Observation 3.4.2.6 we see that if we are interested in the asymptotic behavior of separators of some Boolean circuits (whose degree is bounded by 4), then it does not matter whether we consider vertex-separators or edge-

separators. In the case of unbounded fan-in Boolean circuits we have to take account of the kind of separators we consider.

Next we deal with the question, how much strong separators may differ from separators of the same graph G.

Let, for any function $s : \mathbb{N} \to \mathbb{N}$,

$$\Gamma_s(n) = \sum_{i=0}^{\lceil \log_{3/2} n \rceil} s((2/3)^i \cdot n).$$

We want to show that if G has a $s(n)$ edge-separator then it has a strong $\Gamma_s(n)$ edge-separator. To do this we need the following definition and technical lemma.

Definition 3.4.2.7 *Let G be a graph having an $s(n)$ edge-separator for some $s : \mathbb{N} \to \mathbb{N}$. Let the recursive partitioning of G according to s divide G into G_0 and G_1. Let G_i be partitioned into $G_{i,0}$ and $G_{i,1}$, etc., $G_{i_1,i_2,\ldots,i_k}$ be partitioned into $G_{i_1,i_2,\ldots,i_k,0}$ and $G_{i_1,i_2,\ldots,i_k,1}$, until we get graphs of one node. The* **partition tree** *of this recursive partition is a binary tree $(\overline{V}, \overline{E})$, where*

$$\overline{V} = \{G, G_0, G_1, G_{00}, G_{01}, G_{10}, G_{11}, G_{001}, \ldots\}$$

and

$$\overline{E} = \{(G_{i_1,\ldots,i_k}, G_{i_1,\ldots,i_k,0}), (G_{i_1,\ldots,i_k}, G_{i_1,\ldots,i_k,1}) \mid \text{if } G_{i_1,\ldots,i_k,0} \text{ and } G_{i_1,\ldots,i_k,1} \text{ are nodes of } V\}.$$

To illustrate Definition 3.4.2.7 we give a partition tree of the tree $G = (V, E)$ in Figure 3.15 according to 1 edge-separators. Instead of writing $G_{i_1,\ldots,i_k}$ we write the set of nodes of $G_{i_1,\ldots,i_k}$ directly in Figure 3.16.

Note that each internal node of the partition tree corresponds to a graph of at least two vertices and so it has exactly two sons. Each leaf of the partition tree corresponds to a graph of one node. The depth of the partition tree of a graph $G = (V, E)$ according to an $s(n)$ edge-separator is at most $\log_{3/2} |V|$ because each of the graphs $G_{i_1,i_2,\ldots,i_k,0}$ and $G_{i_1,i_2,\ldots,i_k,1}$ has at most 2/3 of the nodes of $G_{i_1,\ldots,i_k}$. We observe that the vertex $G_{i_1,\ldots,i_k} = (V_{i_1,\ldots,i_k}, E_{i_1,\ldots,i_k})$ of the partition tree corresponds to the cut of $G_{i_1,\ldots,i_k}$ into $G_{i_1,\ldots,i_k,0}$ and $G_{i_1,\ldots,i_k,1}$, and the cardinality of this cut is bounded by $s(|V_{i_1,\ldots,i_k}|) \leq s(|V| \cdot (2/3)^k)$. Thus, each vertex of depth k in the partition tree corresponds to a graph of size at most $(2/3)^k |V|$. Finally we observe that each path of the partition tree from the root to a leaf corresponds to a sequence of partitions corresponding to the nodes of the path, and that the number of all edges of the cuts corresponding to the partitions of this path is bounded by $\Gamma_s(n)$.

Now we prove the following helpful lemma.

Lemma 3.4.2.8 *Let $G = (V, E)$ be a graph of n nodes, $n \in \mathbb{N}$, and let $t \in \mathbb{N}, 0 \leq t < n$. Let G have an $s(n)$ edge-separator for a function $s : \mathbb{N} \to \mathbb{N}$. Then there exists a cut (E', V_1, V_2) such that*

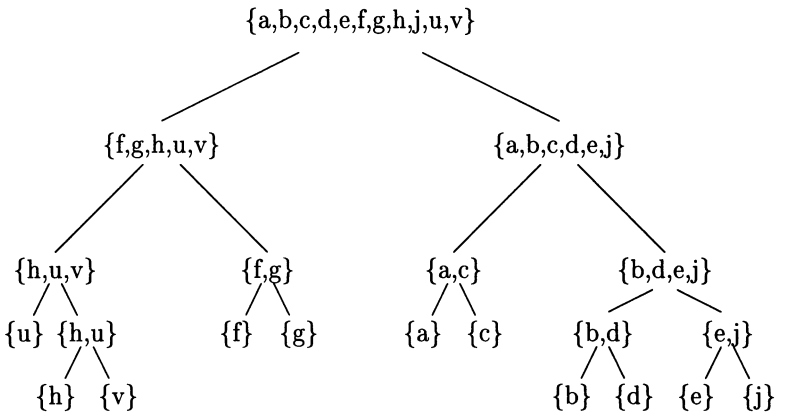

Fig. 3.16. The partition tree of the recursive partition of the tree in Figure 3.15

(i) $|E'| \leq \Gamma_s(n)$

(ii) $|V_1| = t$ *and* $|V_2| = n - t$.

Moreover the graphs $H_i = (V_i, E_i)$ *with* $E_i = E \cap (V_i \times V_i)$ *are unions of graphs corresponding to the nodes of the partition tree of* G *according to the* $s(n)$ *edge-separator.*

Proof. Let T be a partition tree of G according to $s(n)$. We prove the assertion of Lemma 3.4.2.8 by an induction on the depth of T.

First, if the depth is 0 then G has only one node, i.e., $t = 0$.

For the induction, let the left and right children of the root have n_1 and n_2 nodes respectively. We distinguish three cases according to the relations between n_1, n_2, and t.

If $n_1 < t$ take G_1 on the left side and the rest G_2 on the right side. Thus to get t nodes on the left side and $n - t$ ones on the right side one has to partition G_2 into $t - n_1$ nodes and $n_2 - (t - n_1)$ nodes. But the partition tree for G_2 has its depth smaller than the partition tree for G. So we can use the induction hypothesis to get the required separation of G_2 by a cut of at most $\Gamma_s(n_2)$ edges. Since the cut partitioning G into G_1 and G_2 has at most $s(n)$ edges we obtain the required cut (E', V_1, V_2) of G with

$$|E'| \leq s(n) + \Gamma_s(n_2) \leq s(n) + \Gamma_s\left(\frac{2}{3}n\right) = \Gamma_s(n).$$

If $n_1 > t$, but $n_2 \leq t$, changing the roles of G_1 and G_2 we have the same case as described above.

Finally, if n_1 and n_2 are both greater than t take G_1 on the left side and G_2 on the right side. To complete the partition we still need to give $n_1 - t$ nodes of G_1 to the right side. According to the induction we can do this by choosing at most $\Gamma_s(n_1)$ edges of G_1. So we have again obtained the required cut of G by removing at most $s(n) + \Gamma_s(n_1) \leq \Gamma_s(n)$ edges. □

Now we are prepared to give the relation between separators and strong separators.

Lemma 3.4.2.9 *Let $s : \mathbb{N} \to \mathbb{N}$ be a function, and let G be a graph having an $s(n)$ edge-separator. Then G has a strong $\Gamma_s(n)$ edge-separator.*

Proof. Lemma 3.4.2.8 shows that we can partition G having an $s(n)$ edge-separator into two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $||V_1| - |V_2|| \leq 1$ by removing at most $\Gamma_s(n)$ edges. Moreover, this partition has the property that, for $i = 1, 2$, G_i is a union of graphs corresponding to the vertices of the partition tree of G according to the $s(n)$ edge-separator. Clearly, to halve G_i, it is sufficient to cut only one of the graphs constituting G_i into two parts with prescribed sizes. But according to Lemma 3.4.2.8 we can do this by removing at most $\Gamma_s(|V_i|)$ edges from G_i. Moreover, the obtained quarter-graphs are again unions of graphs corresponding to the nodes of the partition tree of G. Thus, this procedure can continue recursively and we have proved that G has a strong $\Gamma_s(n)$ edge-separator. □

The following corollaries show that usually $s(n)$ does not differ too much from $\Gamma_s(n)$.

Corollary 3.4.2.10 *Let $s(n) = \lfloor n^a \rfloor$ for an a, $0 < a \leq 1$. If a graph G has an $s(n)$ edge-separator, then G has a strong $O(s(n)) = O(n^a)$ edge-separator.*

Proof.

$$\begin{aligned} \Gamma_s(n) &\leq n^a + \left(\frac{2}{3}n\right)^a + \left(\frac{4}{9}n\right)^a + \ldots \\ &= n^a \cdot \sum_{i=0}^{\lceil \log_{3/2} n \rceil} \left(\frac{2}{3}\right)^{i \cdot a} \leq \frac{1}{1 - (2/3)^a} \cdot n^a = O(n^a). \end{aligned}$$

□

Corollary 3.4.2.11 *Let $s(n) = \lfloor \log_2 n \rfloor$. If a graph G has an $s(n)$ edge-separator, then G has a strong $O((\log_2 n)^2)$ edge-separator.*

Proof.

$$\Gamma_{s(n)}(n) \leq \sum_{i=0}^{\lceil \log_{3/2} n \rceil} \log_2 \left(\left(\frac{2}{3}\right)^i n \right)$$

$$
\begin{aligned}
&= \log_2 n \cdot \lceil \log_{3/2} n \rceil + \sum_{i=1}^{\lceil \log_{3/2} \rceil} i \cdot \log_2(2/3) \\
&= \log_2 n \cdot \lceil \log_{3/2} n \rceil + \log_2(3/2) \cdot \sum_{i=1}^{\lceil \log_{3/2} \rceil} i \\
&= \log_2 n \cdot \lceil \log_{3/2} n \rceil + \log_2(3/2) \cdot \lceil \log_{3/2} n \rceil \cdot (\lceil \log_{3/2} n \rceil + 1)/2 \\
&= O((\log_2 n)^2)
\end{aligned}
$$

□

3.4.3 Lower Bounds on Boolean Circuits with a Sublinear Separator

Here we want to show nonlinear lower bounds on the combinational complexity of Boolean circuits having a sublinear vertex-separator. The informal idea is a simple extension of the idea leading to nonlinear lower bounds on the area complexity of Boolean circuits in Section 3.4.2. Here we use the separators of a Boolean circuit computing a function f to find a cut of the circuit corresponding to an almost balanced partition of the input nodes (variables). Similarly as in Section 3.4.2 the cardinality of this cut must be at least $\text{acc}(f)$. If we can show that the cardinality of this cut is sublinear for circuits with sublinear separators we obtain that circuits with sublinear separators require a nonlinear number of gates to compute a Boolean function with linear communication complexity.

To formalize this idea we start by partitioning the input vertices for Boolean circuits with vertex-separators.

Definition 3.4.3.1 *Let $G = (V, E)$ be a directed graph (circuit). (U, V_1, V_2) is called a* **vertex-cut of G** *if $U \cup V_1 \cup V_2 = V$, $U \cap V_1 = U \cap V_2 = V_1 \cap V_2 = \emptyset$ and $E \subseteq V_1 \times V_1 \cup V_2 \times V_2 \cup U \times U \cup V_1 \times U \cup U \times V_1 \cup V_2 \times U \cup U \times V_2$.*

Let X be a subset of V. We say that the vertex-cut (U, V_1, V_2) is **almost balanced according to X** *if there exists a partition of U into U_1 and U_2 $(U_1 \cup U_2 = U,\ U_1 \cap U_2 = \emptyset)$ such that*

$$|X \cap (V_1 \cup U_1)| \geq |X|/3 \text{ and } |X \cap (V_2 \cup U_2)| \geq |X|/3.$$

We say that (U, V_1, V_2) is **balanced according to X** *if there exists a partition of U into U_1 and U_2 such that*

$$|X \cap (V_1 \cup U_1)| \geq \lfloor |X|/2 \rfloor \text{ and } |X \cap (V_2 \cup U_2)| \geq \lfloor |X|/2 \rfloor.$$

Lemma 3.4.3.2 *Let $S = (V, E)$ be a Boolean circuit (directed graph). Let X be a subset of V, and let $|X| = n$, $|V| = m$, $n, m \in \mathbb{N}$. Let $s, h : \mathbb{N} \to \mathbb{N}$ be functions such that $s(m) = m/h(m)$. If S has a strong $s(m)$ vertex-separator, then there exists an almost balanced vertex-cut (U, V_1, V_2) of G such that*

(i) (U, V_1, V_2) *is almost balanced according to* X, *and*

(ii) $|U| \leq Z_s(m,n) = m \cdot \sum_{i=0}^{\log_2(m/n)} (2^i h(m/2^i))^{-1}$.

Proof. Since S has a strong $m/h(m)$ vertex-separator we can find a vertex-cut of S partitioning S into two parts S_1 and S_2, each of at most $\lceil m/2 \rceil$ nodes. If none of S_1 and S_2 contains more than $2\lceil n/3 \rceil$ vertices of X, then we are ready. Without loss of generality we assume S_1 contains more than $2n/3$ vertices of X. We halve S_1 by a cut of the cardinality at most $\frac{m}{2}/h(m/2)$ nodes and then we continue always halving the component containing at least $2\lceil n/3 \rceil$ nodes of X. Obviously, this recursive procedure consists of at most $\log_2(m/n)$ steps because $m/2^{\log_2(m/n)+1} = n \cdot m/2m = n/2$ and no component of at most $n/2$ nodes of S can involve more than $n/2$ nodes of X. We observe that if a component S' contains more than $2\lceil n/3 \rceil$ nodes of X and S' is partitioned into S_1 and S_2, none of them containing more than $2\lceil n/3 \rceil$ nodes of X, then one of S_1 and S_2 must contain at least $\lceil n/3 \rceil$ nodes of X. Since in the i-th recursive partition step we have a cut of at most $(m/2^{i-1})/h(m/2^{i-1}) = m \cdot (2^{i-1} \cdot h(m/2^{i-1}))^{-1}$ nodes the proof is completed. □

Now we show that $Z_s(m,n) = o(m)$ if $s(m) = o(m)$ and n is polynomially related to m.

Proposition 3.4.3.3 *Let* $s, h : \mathbb{N} \to \mathbb{N}$ *be functions such that* $s(m) = m/h(m)$, h *is unbounded and* $h(m) = O((\log_2 m)^k)$ *for a positive integer* k. *Then, for any* $n, m \in \mathbb{N}$, $m \leq n^d$ *for a positive constant* d *independent of* n *and* m,

$$Z_s(m,n) = O(m/h(m)) = O(s(m)).$$

Proof. Independently of the properties of the function h the following always hold:

$$\begin{aligned} Z_s(m,n) &= m \cdot \sum_{i=0}^{\log_2(m/n)} (2^i h(m/2^i))^{-1} \\ &\leq m \cdot (h(m/2^{\log_2(m/n)})^{-1} \cdot \sum_{i=0}^{\log_2(m/n)} 2^{-i} \\ &\leq 2 \cdot m/(h(m/(m/n))) = 2m/h(n). \end{aligned}$$

Since $n \geq m^{1/d}$ and $h(m) = O((\log_2 m)^k)$ we have

$$h(n) \geq h(m^{1/d}) = (\frac{1}{d})^k \cdot \Omega((\log_2 m)^k) = \Omega((\log_2 m)^k) = \Omega(h(m))$$

for any constant k independent of m. □

Lemma 3.4.3.4 *Let $s, h : \mathbb{N} \to \mathbb{N}$ be such functions that $s(m) = m/h(m) = o(m)$. Then, for any $n, m \in \mathbb{N}$, $m \leq n^d$ for a positive constant d independent of n and m,*

$$Z_s(m,n) = o(m).$$

Proof. If $h(m) = O(\log_2 m)$, then the sublinearity of $Z_s(m,n)$ follows from Proposition 3.4.3.3. If $h(m) = \Omega(\log_2 m)$ we also get $Z_s(m,n) = o(m)$ because $Z_{s_1}(m,n) \geq Z_{s_2}(m,n)$ for any functions s_1 and s_2 such that $s_1(m) \geq s_2(m)$. □

Lemma 3.4.3.4 shows that an almost balanced separation of the input nodes of a circuit with a sublinear separator is always possible by removing only a sublinear number of nodes. The following assertion shows that we can say much more about $Z_s(m,n)$ if s is a "nice" function.

Proposition 3.4.3.5 *Let $s(m) = k \cdot m^b$ for some constants $k \geq 0$, $0 < b < 1$. Then*

$$Z_s(m,n) = O(m^b) = O(s(m)).$$

Proof. Since $s(m) = k \cdot m/m^{1-b}$ we have

$$\begin{aligned} Z_s(m,n) &= km \cdot \sum_{i=0}^{\log_2(m/n)} (2^i(m/2^i)^{1-b})^{-1} \\ &\leq k \cdot m^b \sum_{i=0}^{\log_2 m} 1/2^{ib}. \end{aligned}$$

Since there exists a positive integer c with the property $1/c \leq b \leq 1/(c+1)$ we obtain

$$\begin{aligned} \sum_{i=0}^{\log_2 m} 1/2^{ib} &\leq \sum_{i=0}^{\log_2 m} 1/2^{i(1/c)} \\ &\leq \sum_{j=0}^{\log_2 m} \sum_{i=0}^{\log_2 m} 1/2^{c[i(1/c)]+j} \\ &\leq c \cdot \sum_{i=0}^{\log_2 m} 1/2^i \leq 2c \leq 2/b. \end{aligned}$$

Thus

$$Z_s(m,n) \leq \frac{2k}{b} \cdot m^b = O(m^b).$$

□

Now we are prepared to generally formulate the lower bound results provided by communication complexity for Boolean circuits with sublinear separators.

Theorem 3.4.3.6 *Let, for any $n \in \mathbb{N}$, S_n be a Boolean circuit computing a Boolean function $f_n : \{0,1\}^n \to \{0,1\}$. Let S_n have a strong $s(m)$ vertex-separator for a function $s : \mathbb{N} \to \mathbb{N}$. Then*

$$4 \cdot Z_s(\mathrm{CC}(S_n), n) \geq \mathrm{acc}(f_n).$$

Proof. Applying Lemma 3.4.3.2 we can find an almost balanced vertex-cut of S_n according to the set of input nodes of S_n with the cardinality bounded by $Z_s(\mathrm{CC}(S_n), n)$. Since each node of S_n has degree bounded by 4, we have a cut containing at most $4 \cdot Z_s(\mathrm{CC}(S_n), n)$ edges of S_n and partitioning the input variables in an almost balanced way. By the same argument as in Section 3.3 we see that the number of edges of this cut must be at least as large as the a-communication complexity of f_n. □

We observe that for $\mathrm{acc}(f_n)$ growing asymptotically faster according to n than $Z_s(\mathrm{CC}(S_n), n)$ we get a nonlinear lower bound on $\mathrm{CC}(S_n)$. Surely we get a nonlinear lower bound if $\mathrm{acc}(f_n) = \Omega(n)$ and $s(m) = o(m)$ because $Z_s(m, n) = o(m)$ in this case (see Lemma 3.4.3.4). Some special lower bounds for some special separators follow.

Theorem 3.4.3.7 *Let $s, h : \mathbb{N} \to \mathbb{N}$ be monotone functions such that $s(m) = m/h(m) = o(m)$ and $h(m) = O((\log_2 m)^k)$ for a positive integer k. Let, for any $n \in \mathbb{N}$, S_n be a Boolean circuit computing a Boolean function $f_n : \{0,1\}^n \to \{0,1\}$. If S_n has a strong $s(m)$ vertex-separator, then*

$$\mathrm{CC}(S_n) = \Omega(\mathrm{acc}(f_n) \cdot h(n)).$$

Proof. We may assume $\mathrm{CC}(S_n) \leq n^2$ because in the opposite case the lower bound is already proved. According to Proposition 3.4.3.3 we have $Z_s(\mathrm{CC}(S_n), n) = O(\mathrm{CC}(S_n)/h(\mathrm{CC}(S_n)))$. Inserting this in the claim of Theorem 3.4.3.6 we get the assertion of Theorem 3.4.3.7. □

Theorem 3.4.3.8 *Let $s(m) = k \cdot m^b$ for some constants $k \geq 0$, $0 < b < 1$. Let, for any $n \in \mathbb{N}$, S_n be a Boolean circuit computing a Boolean function $f_n : \{0,1\}^n \to \{0,1\}$ with $\mathrm{acc}(f_n) = \Omega(n)$. If S_n has a strong $s(m)$ vertex-separator, then*

$$\mathrm{CC}(S_n) = \Omega(n^{1/b}).$$

Proof. According to Proposition 3.4.3.5 we have $Z_s(\mathrm{CC}(S_n), n) = O((\mathrm{CC}(S_n))^b)$. Applying Theorem 3.4.3.6 we obtain $(\mathrm{CC}(S_n))^b = \Omega(n)$. Thus $\mathrm{CC}(S_n) = \Omega(n^{1/b})$. □

3.4.4 Circuit Structures for Which Communication Complexity Does Not Help

In Section 3.4.3 we have shown that communication complexity is able to provide nonlinear lower bounds on combinational complexity of circuits with sublinear separators. A very natural question is whether there exist Boolean circuits (graphs of degree at most 4) without sublinear separators (i.e., whether communication complexity can provide nonlinear lower bounds on combinational complexity or not). In this section we show that there are graphs with bounded degree which do not have any sublinear separator. Moreover, we show that there exists a sequence of Boolean functions $\{f_n\}_{n=1}^{\infty}$, $f_n : \{0,1\}^n \to \{0,1\}$ such that $CC(f_n) = O(n)$ and $acc(f_n) = \Omega(n)$. This means that

(i) for any $n \in \mathbb{N}$, there is a Boolean circuit having no sublinear separator and computing f_n with $O(n)$ gates, and

(ii) any Boolean circuit having a sublinear separator needs nonlinear number of gates to compute f_n.

This result is not only of theoretical interest. Usually the circuit architectures have $O(m/\log m)$ separators and typically the graphs without sublinear separators are random graphs of bounded degree. Thus, the sequence of functions $\{f_n\}_{n=1}^{\infty}$ is theoretically easy but not necessarily easy to compute. If one wants to design a circuit for f_n, then one has to produce either a circuit with nice "regular" structure but with a nonlinear number of gates, or to search for an optimal circuit for f_n with a structure close to random graphs of degree 4.

We start by showing the existence of graphs with degree bounded by 3 and without sublinear separators.

Definition 3.4.4.1 *Let m and d be positive integers, and let $c > 0$ be a real number. An* **(m, d, c)-magnifier** *is a graph $G = (V, E)$ having the following three properties:*

(i) $|V| = m$,

(ii) the degree of G is bounded by d, and

(iii) for each $X \subseteq V$ with $|X| \leq m/2$, there are at least $c \cdot |X|$ edges between the vertices in X and the vertices in $V - X$.

Observation 3.4.4.2 *Let d be a positive integer, and let $c > 0$ be a real number. Let $\{G_m\}_{m=1}^{\infty}$ be a sequence of graphs where G_m is an (m, d, c)-magnifier for every $m \in \mathbb{N} - \{0\}$. Let $s : \mathbb{N} \to \mathbb{N}$ be such a function that, for every $m \in \mathbb{N}$, G_m has a strong $s(m)$ edge-separator. Then*

$$s(m) = \Omega(m).$$

Proof. According to the property (iii) of Definition 3.4.4.1, for every $m \in \mathbb{N} - \{0\}$, every bisection of the (m, d, c)-magnifier G_m has size at least $c \cdot m/2$. □

To show that the lower bound method of Section 3.4.3 based on communication complexity cannot help to obtain a nonlinear lower bound on the combinational complexity of concrete Boolean functions it is sufficient to prove for all $m \in \mathbb{N} - \{0\}$ the existence of $(m, 4, c)$-magnifiers for some positive real number c. In what follows we show even that there exist magnifiers of degree bounded by 3. Note that there is no magnifier of degree bounded by 2 because connected graphs of degree at most two are either cycles or paths.

Definition 3.4.4.3 *Let m be a positive integer, and let* $\mathbf{Per_m}$ *denote the set of all permutations of m elements.*

Let $A \subseteq \mathrm{Per}_m$. We define the **permutation graph of A** *as $G_A = (V, E)$, where*

(i) $V = \{v_1, v_2, \ldots, v_m, u_1, u_2, \ldots, u_m\}$, *and*

(ii) $E = \bigcup_{\alpha \in A} \{(v_1, u_{\alpha(1)}), (v_2, u_{\alpha(2)}), \ldots, (v_m, u_{\alpha(m)})\}$.

We observe that the degree of the permutation graph G_A of a permutation set A is bounded by $|A|$.

The following result shows that if one randomly chooses a set of three permutations then the corresponding permutation graph is with a very high probability a magnifier. The proof of the next theorem is left as an advanced combinatorial exercise to the reader.

Theorem 3.4.4.4 *There exists a positive real number c such that for any $m \in \mathbb{N} - \{0\}$ almost all permutation graphs of sets $A \subseteq \mathrm{Per}_m$ with $|A| = 3$ are $(m, 3, c)$-magnifiers.*

Now we construct a sequence of Boolean functions $F = \{f_n\}_{n=1}^{\infty}$ such that $\mathrm{CC}(f_n) = O(n)$ and $\mathrm{cc}(f_n) = \Omega(n)$. Thus, F can be computed with a linear number of gates but any sequence of circuits with a sublinear separator requires a nonlinear number of gates to compute f.

We can assume that there are a positive constant c and a sequence $\{G_{2l}\}_{l=3}^{\infty}$ of graphs $G_n = (V_n, E_n)$ with the following three properties:

(i) G_{2l} is a 3-regular graph,

(ii) $|V_{2l}| = 2l$,

(iii) for each $X \subseteq V_{2l}$, $|X| \leq l$ there are at least $c \cdot |X|$ edges between the vertices in X and the vertices in $V_{2l} - X$.

The existence of $\{G_{2l}\}_{l=3}^{\infty}$ directly follows from Theorem 3.4.4.4. We note that there even exists an algorithm constructing G_n for every given even $n \geq 6$. But we omit the proof of this fact here. In what follows we also use without any proof (the proof is left as an exercise to the reader) the fact that each 3-regular graph with at least six vertices is 3-colorable (the vertices can be colored by three colors in such a way that for every edge (u, v) of the graph the vertices u and v are colored by different colors).

Now we construct a Boolean function $f_n : \{0,1\}^n \to \{0,1\}$ from $G_n = (V_n, E_n)$ for any $n \in \mathbb{N}$, $n \geq 6$. The construction is done in the following four steps.

1. Denote the n vertices from $G_n = (V_n, E_n)$ by n variables in an arbitrary way.

2. Color the vertices of G_n using 3 colors $\overline{1}, \overline{2}, \overline{3}$ by giving a function $h : V_n \to \{\overline{1}, \overline{2}, \overline{3}\}$ with the property $h(r) \neq h(s)$ for each $(r, s) \in E_n$.

3. For all $i, j \in \{1, 2, 3\}$, $i < j$, define

$$f_n(i,j) = \bigwedge_{(u,v) \in E_{i,j}} (u \vee v),$$

where

$$E_{i,j} = \{(u,v) \mid (u,v) \in E_n \wedge h(u) = \overline{i} \wedge h(v) = \overline{j}\}.$$

4. Define

$$\begin{aligned} f_n(x_1, \ldots, x_n) \quad = \quad & f_n(1,2)(x_1, \ldots, x_n) \vee f_n(1,3)(x_1, \ldots, x_n) \\ & \vee f_n(2,3)(x_1, \ldots, x_n). \end{aligned}$$

Note that f_n is a monotone function. Because G_n is defined only for even $n \geq 6$, we have defined f_n for even n only. But one can extend the definition for odd n in several distinct ways, for instance, $f_{n+1}(x_1, \ldots, x_{n+1}) = f_n(x_1, \ldots, x_n) \vee x_{n+1}$.

Theorem 3.4.4.5 $\mathrm{CC}(f_n) = O(n)$ *and* $\mathrm{acc}(f_n) = \Omega(n)$.

Proof. First we prove $\mathrm{CC}(f_n) = O(n)$. Since the number of vertices in G_n is n and the degree of G_n is bounded by 3, we have that the number of edges in G_n is at most $3n/2$. Thus, the function f_n expressed as the formula $f_n(1,2) \vee f_n(1,3) \vee f_n(2,3)$ has a linear size (i.e., it contains at most a linear number of symbols). Consequently, there is a Boolean circuit with a linear number of gates computing f_n.

To prove $\mathrm{acc}(f_n) = \Omega(n)$ we use the fooling set method. Here, we prefer to describe how to construct the fooling sets rather than to give a formal description of them.

Let Π_n be an arbitrary almost balanced partition of $X = \{x_1, x_2, \ldots, x_n\} = V_n$. Let $E_n(\Pi_n) = E_n \cap (\Pi_{L,X} \times \Pi_{R,X})$. Obviously, property (iii) of G_n implies

that $E_n(\Pi_n) \geq c \cdot \lfloor n/3 \rfloor$. Now we can assume there are $i, j \in \{1,2,3\}$, $i \neq j$, such that the number $d(n)$ of edges $(x_{r_1}, x_{s_1}), \ldots, (x_{r_{d(n)}}, x_{s_{d(n)}})$ leading between $\Pi_{L,X}$ and $\Pi_{R,X}$ and colored in such a way that $h(x_{r_1}) = h(x_{r_2}) = \ldots = h(x_{r_{d(n)}}) = \bar{i}$ and $h(x_{s_1}) = h(x_{s_2}) = \ldots = h(x_{s_{d(n)}}) = \bar{j}$ is at least $c \cdot \lfloor n/3 \rfloor / 6$. Thus we can write

$$f_n(i,j) = \bigwedge_{k=1}^{d(n)} (x_{r_k} \vee x_{s_k}) \bigwedge f'_n(i,j),$$

where $f'_n(i,j) = \bigwedge_{(u,v) \in E'_{i,j}} (u \vee v)$ for $E'_{i,j} = E_{i,j} - \{(x_{r_1}, x_{s_1}), \ldots (x_{r_{d(n)}}, x_{s_{d(n)}})\}$.

Set $A_L = (x_{r_1}, \ldots, x_{r_{d(n)}}) \in (\Pi_{L,X})^{d(n)}$ and $A_R = (x_{s_1}, \ldots, x_{s_{d(n)}}) \in (\Pi_{R,X})^{d(n)}$. Let $set(A_L) \subseteq \Pi_{L,X}$ ($set(A_R) \subseteq \Pi_{R,X}$) be the set of all distinct variables in the vector A_L (A_R). Let $X_L \subseteq set(A_L)$ and $X_R \subseteq set(A_R)$ be subsets of the set of input variables such that for all $x, z \in X_L$, for all $y, w \in X_R$, $((x,y) \in E_n(\Pi_n) \cap (X_L \times X_R) \wedge (z,w) \in E_n(\Pi_n) \cap (X_L \times X_R))$ implies that $(x,w) \notin E_n(\Pi_n) \cap (X_L \times X_R)$ and $(z,y) \notin E_n(\Pi_n) \cap (X_L \times X_R)$. [i.e., X_L and X_R are chosen in such a way that the subgraph spanned by $X_L \cup X_R$ consists of "independent" edges.] Taking X_L and X_R as large as possible we have

$$b(n) = |X_L| = |X_R| \geq d(n)/13 \geq c \cdot \lfloor n/3 \rfloor / 78.$$

To see this, consider the situation when by constructing X_L and X_R one adds one edge (z_1, z_2) to $X_L \times X_R$. Then, to secure the above property, one has to remove from $set(A_L) \times set(A_R)$ all (at most 12) edges connected with all (at most 4) vertices adjacent to z_1 and z_2.

Let $X_L = \{x_{u_1}, \ldots, x_{u_{b(n)}}\}$ and $X_R = \{x_{v_1}, \ldots, x_{v_{b(n)}}\}$.

Thus we can write

$$f_n(i,j) = \bigwedge_{k=1}^{b(n)} (x_{u_k} \vee x_{v_k}) \bigwedge \hat{f}_n(i,j) \bigwedge f'_n(i,j),$$

for some conjunction of elementary disjunctions $\hat{f}_n(i,j)$.

We describe the construction of the 1-fooling set $\mathcal{A}(\Pi_n, f_n)$ for Π_n and f_n as a subset of $\{0,1\}^n$ in the following stages:

(1) Choose four variables y_1, y_2, y_3, y_4 from X such that $y_1 = y_2 = y_3 = y_4 = 0$ implies $f_n(k,l) = 0$ for each $(k,l) \neq (i,j)$, $k, l \in \{1,2,3\}$, $k < l$ and $y_1 = y_2 = y_3 = y_4 = 0$ does not imply $f_n(i,j) = 0$. Fix the zero values of y_1, y_2, y_3, y_4 in all words in $\mathcal{A}(\Pi_n, f_n)$.

(2) Fix the value 1 for all variables in $X' = X - (X_L \cup X_R \cup \{y_1, y_2, y_3, y_4\})$.

(3) The variables in $X_L \cup X_R$ may have both values 0 and 1 with the following restriction:

$$\text{for all } k \in \{1, \ldots, b(n)\} \text{ if } x_{u_k} = 1(0) \text{ then } x_{v_k} = 0(1).$$

Now we show that $\mathcal{A}(\Pi_n, f_n)$ is a fooling set for Π_n and f_n. Stage (1) ensures that $f_n(\alpha) = f_n(i,j)(\alpha)$ for all $\alpha \in \mathcal{A}(\Pi_n, f_n)$. Stage (2) secures that $f'_n(i,j)(\alpha) = 1$ and $\hat{f}_n(i,j)(\alpha) = 1$ for all $\alpha \in \mathcal{A}(\Pi_n, f_n)$, which implies

$$f_n(\alpha) = \bigwedge_{k=1}^{b(n)} (\alpha_{u_k} \vee \alpha_{v_k}) \text{ for each } \alpha = \alpha_1\alpha_2 \ldots \alpha_n \in \mathcal{A}(\Pi_n, f_n).$$

From this fact and stage (3) we have that for all distinct $\alpha = \alpha_1 \ldots \alpha_n$, $\beta = \beta_1 \ldots \beta_n \in \mathcal{A}(\Pi_n, f_n)$ either

$$1 = f_n(\alpha) \neq f_n(\Pi_n^{-1}(\alpha_{\Pi_n,L}, \beta_{\Pi_n,R})) = \bigwedge_{k=1}^{b(n)} (\alpha_{u_k} \vee \beta_{v_k}) = 0$$

or

$$1 = f_n(\alpha) \neq f_n(\Pi_n^{-1}(\beta_{\Pi_n,L}, \alpha_{\Pi_n,R})) = \bigwedge_{k=1}^{b(n)} (\beta_{u_k} \vee \alpha_{v_k}) = 0.$$

Thus, $\mathcal{A}(\Pi_n, f_n)$ is a fooling set.

Now we estimate the cardinality of $\mathcal{A}(\Pi_n, f_n)$. Since $b(n) \geq c \cdot \lfloor n/3 \rfloor / 78$ we have

$$|\mathcal{A}(\Pi_n, f_n)| \geq 2^{(c \cdot \lfloor n/3 \rfloor / 78) - 4}.$$

Thus $\text{acc}(f_n) = \Omega(n)$. □

Thus, we have shown that $\{f_n\}_{n=6}^{\infty}$ has linear combinational complexity but because of linear communication complexity and Theorem 3.4.3.6, every sequence of Boolean circuits with a sublinear separator computing $\{f_n\}_{n=6}^{\infty}$ has a nonlinear number of gates. This also implies that the Boolean circuit constructed in the proof of Theorem 3.4.4.5 to compute f_n with a linear number of gates does not have any sublinear separator.

3.4.5 Planar Boolean Circuits

A Boolean circuit $S = (V, E)$ is called **planar** if (V, E) considered as a graph is planar. The aim of this section is to prove that any planar Boolean circuit computing a Boolean function f has to have $\Omega((\text{acc}(f))^2)$ gates. Note that we cannot reduce this task to obtaining $\Omega((\text{cc}(f))^2)$ on the area complexity of Boolean circuits (Theorem 3.3.3.4) because we do not have any technique enabling us to lay a planar graph of m nodes into an $O(m)$ area. But according to Theorem 3.4.3.6 and Proposition 3.4.3.5 it suffices to prove that every planar graph $G = (V, E)$ has a $4 \cdot \sqrt{2} \cdot \sqrt{|V|}$ vertex-separator. To do it in this way we first show that each planar graph has a special representation form, and than we use this form to prove the existence of an $O(\sqrt{m})$ vertex-separator.

Definition 3.4.5.1 *Let $G = (V, E)$ be a graph. A* **planar representation of G**, *denoted $Pl(G)$, is a picture in the plane with the following properties:*

(i) *every node v of G is depicted as a point $p(v)$ of the plane labeled by the name of the node v, any two distinct nodes of G are depicted as two distinct points of the plane,*

(ii) *every edge $(u,v) \in E$ is depicted as a curve $c(u,v)$ connecting points $p(u)$ and $p(v)$,*

(iii) *for every two different edges $(u_1,v_1),(u_2,v_2) \in E$, $|\{u_1,v_1,u_2,v_2\}| = 4$ implies that $c(u_1,v_1)$ and $c(u_2,v_2)$ do not have any common point (i.e., do not cross each other),*

(iv) *for every two edges $(u,v),(u,w) \in E$, $v \neq w$ implies that $c(u,v)$ and $c(u,w)$ have exactly one common point $p(u)$.*

Each graph having a planar representation is called **planar**.

To give an example, Figure 3.17 contains a planar representation of the graph $G = (\{v_1, v_2, \ldots, v_{13}\}, \{(v_1,v_2), (v_1,v_3), (v_1,v_4), (v_1,v_{13}), (v_2,v_6), (v_3,v_4), (v_3,v_5), (v_3,v_7), (v_3,v_{11}), (v_4,v_5), (v_4,v_8), (v_5,v_{10}), (v_5,v_{13}), (v_6,v_9), (v_7,v_9), (v_8,v_{12}), (v_{10}, v_{12}), (v_{11},v_{12})\}$.

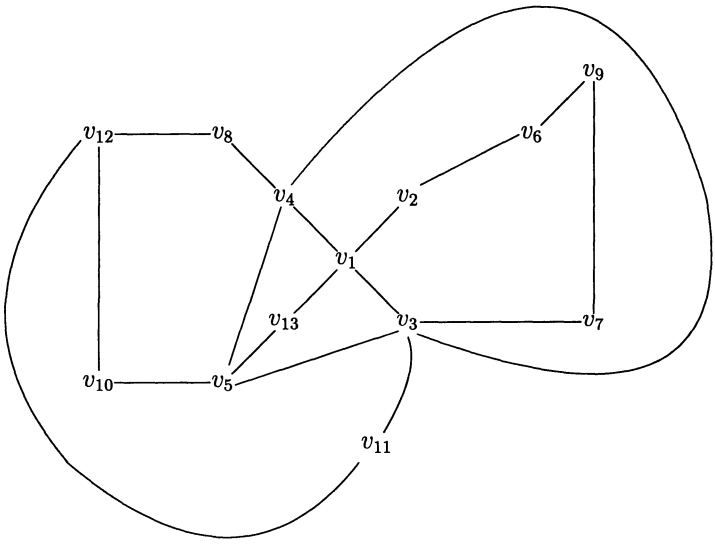

Fig. 3.17. A planar representation of a planar graph

Definition 3.4.5.2 *Let $G = (V,E)$ be a connected planar graph, and let v_1 be a node of G. Let $V = \bigcup_{i=0}^{k} V_i$ for some $k \in \mathbb{N}$, where $V_i = \{w \in V \mid \text{dist}_G(v_1,w) = i\}$ for $i = 1,\ldots,k$ and $V_k \neq \emptyset$. For every $i \in \{0,1,\ldots,k\}$, V_i is called the* **i-th**

ring of G. *A* **concentric representation of G with the origin v_1** *is a planar representation of a graph $G' = (V', E')$ having the following properties:*

(i) $V \subseteq V'$, *the nodes of* V *are called the* **original nodes** *and the nodes of* $V' - V$ *are called* **dummy nodes**,

(ii) $V_i \subseteq V_i' = \{x \in V' \mid \text{dist}_{G'}(v_1, x) = i\}$ *for all* $i = 0, 1, \ldots, k$, $V' = \bigcup_{i=0}^{k} V_i'$. *The set* V_i' *is called the* ***i*-th ring of G'**.

(iii) For every edge $(u, w) \in E'$, *either* u *and* w *are from the same ring or* $u \in V_i'$, $w \in V_j'$ *and* $|j - i| = 1$.

(iv) For every ring V_i' $(i = 1, \ldots, k)$, *the nodes of* V_i' *can be ordered in a sequence* $x_1, x_2, \ldots, x_l$ *such that* E' *contains the set of edges* $R_i = \{(x_1, x_2), (x_2, x_3)\}, \ldots, (x_{l-1}, x_l), (x_l, x_1)$. *Every element of* R_i *is called a* **ring edge** *of the* i*-th ring. The curves* $c(x_1, x_2), c(x_2, x_3), \ldots, c(x_{l-1}, x_l)$, $c(x_l, x_1)$ *of the planar representation of* G' *represent a closed curve called the* ***i*-th circle of G'**. *All nodes of* $\bigcup_{j=0}^{i-1} V_j'$ *and all curves realizing the edges leading between the nodes of* $\bigcup_{j=0}^{i-1} V_j'$ *as well as the curves realizing the edges* (u, w), $u \in V_{i-1}'$ *and* $w \in V_i'$ *lay inside the area bounded by the* i*-th circle. All nodes of* $\bigcup_{z=i+1}^{k} V_z'$, *all curves realizing the edges between the nodes of* $\bigcup_{z=i+1}^{k} V_z'$, *the curves realizing the edges* (x, y), $x \in V_i'$, $y \in V_{i+1}'$, *as well as the curves realizing the edges* (x_1, y_1), $x_1, y_1 \in V_i'$ *lay outside the area bounded by the* i*-th circle.*

(v) For every edge $(v, w) \in E$ *there exists a path* $P_{v,w} = v, u_1, u_2, \ldots, u_z, w$ *in* G' *such that*

- $u_1, u_2, \ldots, u_z$ *are dummy nodes for an integer* $z \geq 0$, *and*
- *either all edges* $(v, u_1), (u_1, u_2), \ldots, (u_z, w)$ *of the path* $P_{v,w}$ *are ring edges or all edges of the path* $P_{v,w}$ *are non-ring edges.*

The main idea of the above definition is to find a special planar representation of a planar graph G with property (iv), i.e.,

(1) to have a circle containing exactly the nodes with the same distance to the origin for every possible distance,

(2) the inside area of the i-th circle properly contains the inside area of the $(i-1)$-st circle and contains all curves realizing the edges between these two circles (rings) too, and

(3) all curves realizing the non-ring edges between the nodes of the i-th ring lay in the outside area of the i-th circle.

To achieve such a planar realization it is allowed to add some dummy nodes and ring edges to the given graph G. Informally, this can be achieved in the

following way. One takes a spanning tree of G (a tree which is a subgraph of G and contains all nodes of G) with the root v_1 obtained by the breath-first-search (i.e., the distances of every node w to v_1 in the spanning tree is the same as $\text{dist}_G(v_1, w)$). Considering a reasonable planar representation of the spanning tree one can add ring edges whose planar realization results in the circles of nodes with the same distance to the origin. After this it remains to find a planar realization of the edges of G which are not the edges of the spanning tree. These rest edges connect either two nodes of the same ring or two nodes of two neighboring rings. (The existence of an edge of G between V_i and V_j with $|j - i| \geq 2$ contradicts the definition of rings as sets of nodes with the same distances to the origin.) Let (v, w) be such an edge with $v \in V_{i+1}$, $w \in V_i$. First we look for a possibility to find a curve lying in the outside area of the i-th circle and in the inside area of the $(i + 1)$-th circle and crossing no curve of the current planar picture. If it is possible we do it. If it is impossible we are allowed to use a curve for the realization of (u, w) which crosses only the ring edges of G. The common points of $c(u, v)$ and ring edges define new dummy nodes added to G. It is possible to find such a curve $c(u, w)$ due to the planarity of G (more details are given in the proof of the following lemma). For the edges between the nodes of the same circle (ring) a curve lying outside this circle is constructed in the above way too.

Figure 3.18 gives a concentric representation of the planar graph of Figure 3.17 with the origin v_1. We observe that the rings are $V_0' = \{v_1\}$, $V_1' = \{v_2, v_3, v_4, v_{13}\}$, $V_2' = \{u_2, v_6, v_7, u_5, v_{11}, v_5, v_8\}$, $V_3' = \{u_3, v_9, u_4, u_1, v_{10}, v_{12}\}$, and u_1, u_2, u_3, u_4, u_5 are the dummy nodes. The ring edges are depicted by the dashed lines. The planar realization of two edges (v_{11}, v_{12}) and (v_3, v_4) requires us to cross the dummy edges. To realize the connection between v_{11} and v_{12} we cross the third circle in the dummy node u_1 and lay two curves $c(v_{11}, u_1)$ and $c(u_1, v_{12})$. To realize the connection between v_3 and v_4 we have to cross the second circle twice and the third circle twice. The realization of the original edge (v_3, v_4) is now the realization of the path $v_4, u_2, u_3, u_4, u_5, v_3$, all of whose inner nodes are dummy nodes. The planar realization of the edges (v_{11}, v_{12}) and (v_3, v_4) is possible without crossing another non-ring edge as it is shown in the planar realization of the graph in Figure 3.17. In fact we did nothing more than depict the edges (v_{11}, v_{12}) and (v_3, v_4) in almost the same way as in Figure 3.17.

We now show formally that each planar graph has a concentric representation.

Lemma 3.4.5.3 *For any connected planar graph $G = (V, E)$ and any node $v_1 \in V$ there exists a concentric representation of G with the origin v_1.*

Proof. Let $G = (V, E)$, $V_i = \{w \mid \text{dist}(v_1, w) = i\}$ for $i = 0, 1, \ldots, k$ and $V = \bigcup_{i=0}^{k} V_i$. The construction is done by induction on the distance d from the origin. First we take the origin and show how to build the first circle. Let us consider a planar realization $Pl(G)$ of the graph G. The node v_1 has $|V_1|$

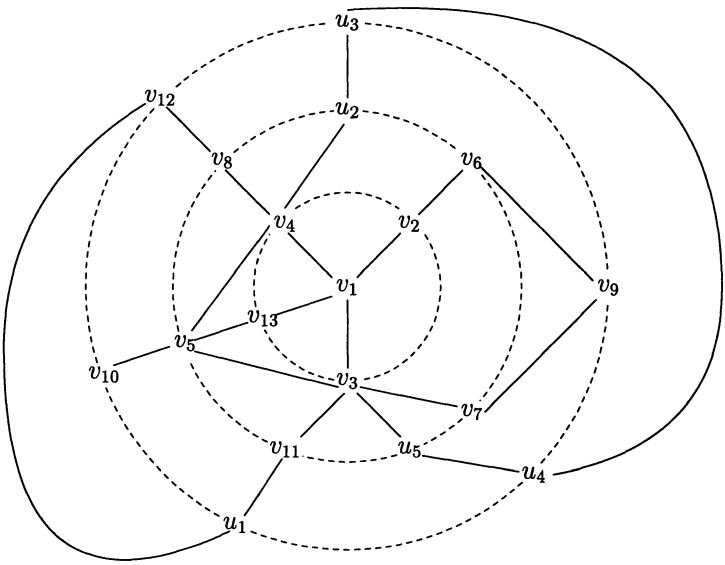

Fig. 3.18. A concentric representation of the planar graph in Figure 3.17

neighboring nodes there. Each neighbor of the origin can have edges leaving in various directions. However, because of the planarity of G there is surely room to connect the nodes of the first ring inside any other edges leaving those nodes. The introduced edges are the ring edges forming the first circle (see Figure 3.20 for an example; the dashed lines are the ring edges of the first circle). If there already was an edge of G between two consecutive nodes of the first circle, we remove it because the concentric representation must have the property that the edges connecting the consecutive codes of the circle be inside any other edges leaving either node. Now, we observe that for the first circle all the properties (i), (ii), (iii), (iv), (v) of Definition 3.4.5.2 are fulfilled. (Note that if we wish we can distort the plane so that the circle even forms a convex polygon). Thus, we have a planar representation Pl_1 of G (with some additional ring edges) whose first ring satisfies the conditions of concentric representation.

For the induction we assume that we have a planar representation Pl_d of G, where the first d rings satisfy the conditions of a concentric representation. Note that the edges outside the d-th ring go to the nodes of the d-th ring or to the nodes of V_{d+1}. We consecutively build the $(d+1)$-th circle in the following way. We take a node v of the d-th ring. This node has zero or more edges emanating from it outside the d-th ring. Some of these edges lead to nodes $r_1, \ldots r_z$ of the $(d+1)$-th ring, the rest lead to other nodes of the d-th ring. Order the nodes

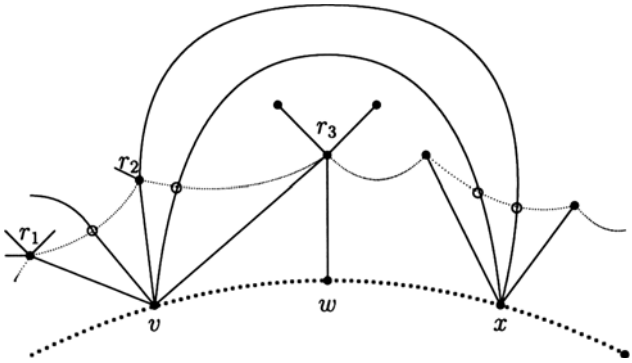

Fig. 3.19. The process of building a new circle of a planar representation of a planar graph

$r_1, \ldots r_z$ of the $(d+1)$-th ring in the order they are connected to v in the planar representation Pl_d. Now take a neighboring node w of v in the d-th circle and order the nodes $w_1, w_2, \ldots, w_a$ of the $(d+1)$-th ring connected to w. Continuing in this way we obtain an order $r_1, \ldots r_z, w_1, \ldots, w_a, \ldots$ of all nodes of V_{d+1}. Using this order we form the $(d+1)$-th circle by adding the ring edges to Pl_d in the same way as in the first step of the induction. We remove edges of the original graph that are duplicated by ring edges. If an original edge of G crosses a new ring edge in some point, then we put a new dummy node on this point. An example of this process is depicted in Figure 3.19. Note that every edge from the d-th ring (circle) to the d-th ring (circle) that passes outside any node of the $(d+1)$-th circle has two cuts with the ring edges of the $(d+1)$-th circle

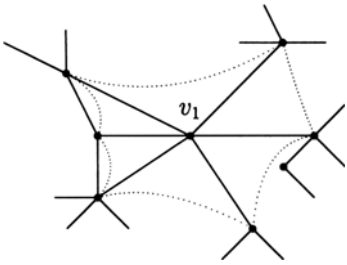

Fig. 3.20. The first circle of a planar representation of a planar graph

(edge (v, x) in Figure 3.19) and each edge from the d-th ring to the $(d+1)$-th ring going around nodes of the $(d+1)$-th ring in the current planar realization crosses exactly one ring edge of the $(d+1)$-th circle (edge (r_2, x) in Figure 3.19). No other dummy nodes than the ones described above are introduced. From the above construction we see that each dummy node on the $(d+1)$-th circle has one edge going to a node on the d-th circle (it does not matter whether this is a node of V_d or a dummy node of the d-th circle). One can easily observe that we have obtained a planar representation Pl_{d+1} of G whose first $d+1$ circles satisfy the conditions of Definition 3.4.5.2. This concludes the induction as well as the proof of Lemma 3.4.5.3. □

Observation 3.4.5.4 *Let $G = (V, E)$ be a planar graph, and let $G' = (V', E')$ be the concentric representation of G described above. If $(w, v) \in E$, then, according to condition (v) of Definition 3.4.5.2, the connection (w, v) is realized in G' as a path $P_{w,v} = w, u_1, \ldots, u_d, v$, where d is a nonnegative integer (possibly zero) and $u_1, \ldots, u_d$ are dummy nodes. Moreover, if (w, u) connects the nodes of two consecutive rings of G, then either $d = 0$ or d is odd and $P_{w,v}$ contains no ring edge of G'.*

Observation 3.4.5.5 *Let $G = (V, E)$ be a planar graph, and let $G' = (V', E')$ be the concentric representation of G described above. Let (v_1, v_2) and (w_1, w_2) be two different edges of G. Let, for $d \geq 1$, $v_1, u_1, u_2, \ldots, u_d, v_2$ be the realization of (v_1, v_2) in G', and let, for $r \geq 1$, $w_1, z_1, z_2, \ldots, z_r, w_2$ be the realization of (w_1, w_2) in G'. Then either*

(i) $\{u_1, u_2, \ldots, u_d\} \cap \{z_1, z_2, \ldots, z_r\} = \emptyset$

or

(ii) $|\{u_1, \ldots, u_d\} \cap \{z_1, \ldots, z_r\}| \in \{1, 2\}$ *and exactly one of the paths $v_1, u_1, u_2, \ldots, u_d, v_2$ and $w_1, z_1, z_2, \ldots, z_r, w_2$ consists of ring edges only.*

Proof. In the construction of the concentric representation G' of G we have inserted a dummy node on each intersection of one ring edge and a curve realizing some non-ring edge of G in a planar representation. Thus each dummy node u has exactly degree 4. Moreover, two of the edges adjacent to u are ring edges and two of the edges adjacent to u are non-ring edges. These non-ring edges are a part of exactly one path realizing a non-ring edge of G in G'. □

Thus, for every dummy node u of G' we can unambiguously assign to u two pairs of vertices

$$\mathbf{dum_1}(u) = (v_1, v_2),$$

where $v_1, z_1, \ldots, z_s, u, z_{s+1}, \ldots, z_r, v_2$ is a path in G', $z_1, \ldots, z_r$ are dummy nodes, and this path does not contain any ring edge of G', and

$$\mathbf{dum_2}(u) = (w_1, w_2),$$

where $w_1, y_1, \ldots, y_p, u, y_{p+1}, \ldots, y_k, w_2$ is a part of a circle of G' (i.e., it consists of ring edges only).

Next we show that the existence of a small vertex-cut $S' \subseteq V'$ disconnecting a concentric representation (V', E') of a planar graph $G = (V, E)$ implies the existence of a small vertex-cut $S \subseteq V$ of G. Thus, to find a small vertex-cut of a planar graph G it is sufficient to find a small vertex-cut of a concentric representation of G.

Lemma 3.4.5.6 *Let $G = (V, E)$ be a planar graph, and let $G' = (V', E')$ be the concentric representation of G fixed above. If there exists an almost balanced vertex-cut S' of $G' = (V', E')$ according to V, then there exists an almost balanced vertex-cut S of G with cardinality $|S| \leq 2|S'|$.*

Proof. Let (S', V_1', V_2') be an almost balanced vertex-cut of $G' = (V', E')$ according to the set of original nodes $V \subseteq V'$. If S' does not contain any dummy node, then obviously S' is an almost balanced partition of $G = (V, E)$ too. If S' contains at least one dummy node, then we replace each dummy node of S' by two original nodes of G in the following way. If $u \in S' \cap V' - V$, $\text{dum}_1(u) = (v, w)$, and $\text{dum}_2(u) = (x, y)$, then we replace u by v and x (note that it does not matter whether we take v $[x]$ or w $[y]$ from the pair (v, w) $[(x, y)]$).

Clearly, S constructed from S' in the way described above defines a vertex-cut (S, V_1, V_2) of G with cardinality $|S| \leq 2 \cdot |S'|$. Since $V_1 \subseteq V_1'$ and $V_2 \subseteq V_2'$, and $|V_i' \cap V| \leq 2n/3$ for $i = 1, 2$, (S, V_1, V_2) is almost balanced. □

Now we use the concentric representation of a planar graph G to find a small set of nodes of G whose removal disconnects G into two planar parts G_1 and G_2, each of which does not contain more than two thirds of the nodes of G. Since we again obtain planar graphs G_1 and G_2 by this separation, we can continue to disconnect G_1 and G_2 in the same way in order to show the existence of a small vertex-separator of G.

Lemma 3.4.5.7 *Let $G = (V, E)$ be a planar graph of n nodes. Then there exists $S \subseteq V$ such that*

(i) $|S| \leq 4 \cdot \sqrt{2} \cdot \sqrt{n}$, *and*

(ii) S is an almost balanced vertex-cut of G.

Proof. Let $G = (V, E)$ be a planar graph of n nodes, and let v_1 be a node of G. Let $V = \bigcup_{i=0}^{k} V_i$, where $V_i = \{w \mid \text{dist}_G(v_1, w) = i\}$ are the rings of G for every $i \in \{0, 1, \ldots, k\}$. For convenience we say that $V_{k+1} = \emptyset$ is the last ring of G. We say that a ring V_i is **tight** if $|V_i| \leq \sqrt{2} \cdot \sqrt{n}$. We observe that $V_0 = \{v_1\}$ and V_{k+1} are tight rings, i.e., there exist at least two tight rings in G.

Now we distinguish two possibilities depending on the existence of two numbers $i, j \in \{0, 1, \ldots, k+1\}$, $i \neq j$, such that

(a) V_i and V_j are tight rings,

(b) $|\bigcup_{m=i+1}^{j-1} V_m| \leq 2n/3$, and

(c) $|\bigcup_{r=0}^{i-1} V_r| + |\bigcup_{s=j+1}^{k} V_s| \leq 2n/3$.

In what follows we handle these two possibilities separately.

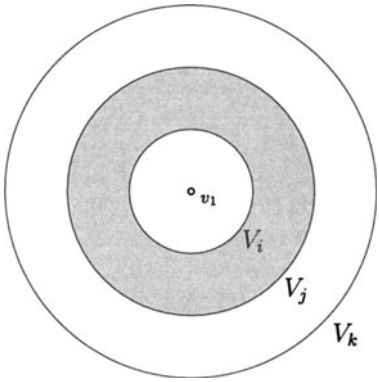

Fig. 3.21. The partition of a planar representation by removing two tight rings V_i and V_j

(1): We assume that there are two distinct $i, j \in \{0, 1, \ldots, k+1\}$ having the properties (a), (b), and (c). We see that every ring is a vertex-cut of G because there is no edge connecting two rings V_{l_1} and V_{l_2} with $|l_1 - l_2| \geq 2$ in G. Thus, the removal of two rings V_i and V_j divides G into three parts (see Figure 3.21). Because of the properties (a), (b), and (c), the vertex-cut $(V_i \cup V_j, \bigcup_{m=i+1}^{j-1} V_m, \bigcup_{r=0}^{i-1} V_r \cup \bigcup_{s=j+1}^{k} V_s)$ fulfills the conditions (i) and (ii) of Lemma 3.4.5.7. (Note we did not need to use the planarity of G to prove Lemma 3.4.5.7 in case (1).)

(2): Now we consider the opposite case, where no pair of tight rings fulfilling the conditions (a), (b), and (c) exists. This means that there exist two numbers $l, d \in \{0, 1, \ldots, k+1\}$, $l < d$ such that

(a') V_l and V_d are tight,

(b') V_m is not tight for any $m \in \{l+1, l+2, \ldots, d-1\}$, and

(c') $|\bigcup_{m=l+1}^{d-1} V_m| > 2n/3$.

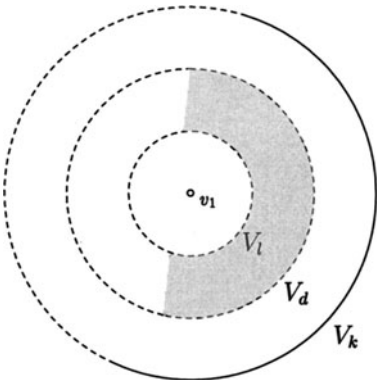

Fig. 3.22. A partition of the middle part of a planar representation by removing nodes of two radii

We observe that $d - l \leq \sqrt{n}/\sqrt{2}$ because $|\bigcup_{m=l+1}^{d-1} V_m| < n$ and $|V_m| > \sqrt{2} \cdot \sqrt{n}$ for every $m \in \{l+1, \ldots, d-1\}$.

Now we shall realize the following global strategy depicted in Figure 3.22. We consider a concentric representation $G' = (V', E')$ of G according to the node v_1. We search for an almost balanced vertex-cut of G' according to V as follows. First we take the nodes of the l-th circle and of the d-th circle in order to divide G' into three parts (see Figure 3.22). We know that the middle part contains the set of the original nodes $\overline{V} = \bigcup_{m=l+1}^{d-1} V_m$, and the cardinality of $\overline{V}$ is greater than $2n/3$. Thus, we shall attempt to find a set S' dividing the middle part of G' along two "radii" into two parts (see Figure 3.22), none of them containing more than $2n/3$ nodes of $\overline{V}$. Then S' together with the nodes of the l-th circle and of the d-th circle divides G' into 4 parts, each of them containing at most $2n/3$ original nodes of G. Obviously, we can combine these 4 parts into two groups containing no more than $2n/3$ vertices of V each. To get an almost balanced vertex-cut of G we take $V_l \cup V_d \cup S$, where $S \subseteq V$, $|S| \leq 2 \cdot |S'|$, is constructed from S' by exchanging the dummy nodes of S' by the original nodes as described in the proof of Lemma 3.4.5.6. Note that we need not replace the dummy nodes of the l-th circle and of the d-th circle by any original node of G in the vertex-cut of G because to remove the rings V_l and V_d is sufficient for the first division step separating G into three parts.

Thus, let $\overline{G}$ be the rest of the concentric representation G' after removing all nodes of all circles V_r for $r \in \{0, 1, \ldots, l\} \cup \{d, d+1, \ldots, k\}$.

In what follows a **radius** is a set of nodes $\{x_{l+1}, \ldots x_{d-1}\}$ of $\overline{G}$ such that $x_i \in V_i'$ for every $i \in \{l+1, l+2, \ldots, d-1\}$. An example of a radius are nodes of any path going from the $(d-1)$-st circle to the $(l+1)$-st circle and containing exactly one node of each circle of $\overline{G}$ (see, for instance, the set of nodes $\{v_{13}, v_5, v_{10}\}$ in Figure 3.18). For any three different nodes v, w, and x

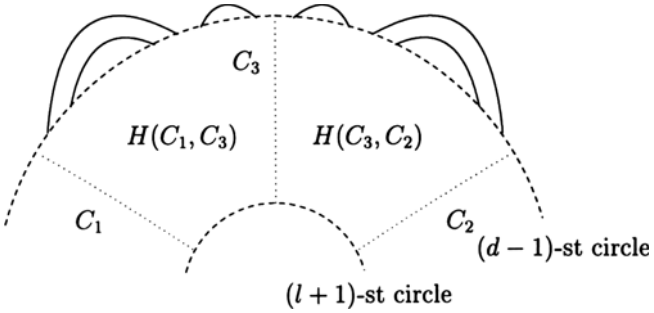

Fig. 3.23. The search for a candidate C_3 beeing clockwise between the candidates C_1 and C_2

of the same circle, we say that v **is clockwise before** w **according to** x if one can go from v to w and from w to x via the circle in the clockwise direction. (For instance, v_8 is clockwise before u_2, v_6, v_7, u_5, and v_{11} according to v_5 in Figure 3.18.) If v is clockwise before w according to x we also say that w **is clockwise between** v **and** x. Obviously, for any three different nodes w, v, and x of a circle, either w is clockwise between v and x or w is clockwise between x and v (for instance u_3 is clockwise between v_{12} and v_9 and v_{11} is clockwise between v_6 and v_5 in Figure 3.18). Let $C_1 = \{x_{l+1}, x_{l+2}, \ldots x_{d-1}\}$, $C_2 = \{y_{l+1}, y_{l+2}, \ldots y_{d-1}\}$, $C_3 = \{w_{l+1}, w_{l+2}, \ldots w_{d-1}\}$ be three radii such that $C_1 \cap C_3 = C_2 \cap C_3 = \emptyset$ and, for every $j \in \{l+1, l+2, \ldots, d-1\}$, x_j, y_j, and w_j are from V'_j. We say that C_2 **is clockwise between** C_1 **and** C_3 if, for every $j \in \{l+1, l+2, \ldots, d-1\}$, either $x_j \equiv y_j$ or y_j is clockwise between x_j and w_j in the j-th circle. (For instance, $\{u_3, u_2, v_4\}$ is clockwise between $\{v_{12}, v_8, v_4\}$ and $\{v_3, u_5, u_1\}$ in Figure 3.18 and C_3 is clockwise between C_1 and C_2 in Figure 3.23.) We say that two radii C_1 and C_2 **do not cross** each other if there exists a radius D, $D \cap C_1 = D \cap C_2 = \emptyset$ such that either C_2 is clockwise between C_1 and D or C_1 is clockwise between C_2 and D. In the opposite case we say that C_1 **crosses** C_2. (The radii $\{v_3, v_7, v_9\}$ and $\{v_3, u_5, u_4\}$ do not cross in Figure 3.18, but $\{v_3, v_5, v_{10}\}$ crosses $\{u_1, v_5, v_4\}$.) Finally, we say that two radii D_1 and D_2 are **candidates for** S' (for the separation of $\overline{G}$) iff $D_1 \cup D_2$ is a vertex-cut of $\overline{G}$ and D_1 and D_2 do not cross. If D_1 and D_2 are candidates, we define $H(D_1, D_2)$ as the set of nodes lying clockwise between D_1 and D_2 (see Figure 3.23). We observe that any two candidates C_1, C_2 divide $\overline{G}$ into two components given by the node sets $H(C_1, C_2)$ and $H(C_2, C_1)$. (For instance, the candidates $\{v_4, v_8, v_{12}\}$ and $\{v_3, u_5, u_4\}$ divide the tree outside the circles in Figure 3.18 into $H(\{v_4, v_8, v_{12}\}, \{v_3, u_5, u_4\}) = \{v_2, u_2, v_6, v_7, u_3, v_9\}$, and $H(\{v_3, u_5, u_4\}, \{v_4, v_8, v_{12}\}) = \{v_{13}, v_{11}, v_5, u_1, v_{10}\}$. Note that any two radii C_1, C_2 which do not cross have not to be candidates (a vertex-cut) because there

may be an edge connecting $H(C_1, C_2)$ and $H(C_2, C_1)$ via a curve lying outside the last circle of $\overline{G}$ (for instance, $C_1 = \{v_{13}, v_5, v_{10}\}$ and $C_2 = \{v_4, u_2, u_3\}$ do not cross, but the edge (v_{12}, u_1) connects $H(C_1, C_2)$ and $H(C_2, C_1)$ and so $C_1 \cup C_2$ is not a vertex-cut of the three outside circles of the graph of Figure 3.18.)

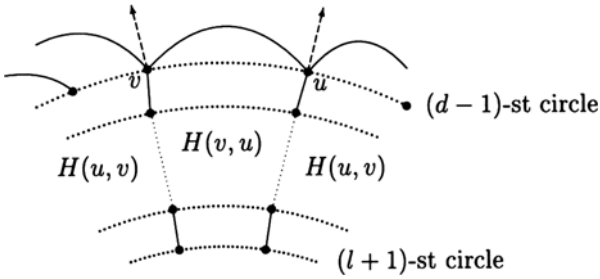

Fig. 3.24. The search for two initial candidates, where the nodes v and u are free to the outside

As we have already mentioned, to complete the proof of Lemma 3.4.5.7 it is sufficient to find two radii as in Figure 3.22 separating $\overline{G}$ into $\overline{G}_1 = (\overline{V}_1, \overline{E}_1)$ and $\overline{G}_2 = (\overline{V}_2, \overline{E}_2)$ such that $|\overline{V}_i \cap V| \leq 2n/3$ for $i = 1, 2$. The rough strategy searching for these two radii works as follows.

First, one finds two candidates separating $\overline{G}$ into two disconnected parts of arbitrary sizes. Then, step by step we replace one of the candidates for a new radius lying in the larger part (according to V) between the two current candidates (see Figure 3.23). This procedure halts when the two candidates separate $\overline{G}$ into $\overline{G}_1$ and $\overline{G}_2$ with the above properties. Note that this procedure must stop because if one part $H(C_1, C_2)$ of $\overline{G}$ between two candidates (radii) C_1 and C_2 is larger than $2n/3$ and after one candidate has been replaced by a new candidate C_3 in such a way that neither of the two parts $H(C_1, C_3)$ and $H(C_3, C_2)$ of $\overline{G}$ (see Figure 3.23) contains more than $2n/3$ nodes of V, then at least one of $H(C_1, C_3)$ and $H(C_3, C_2)$ must contain together with the two

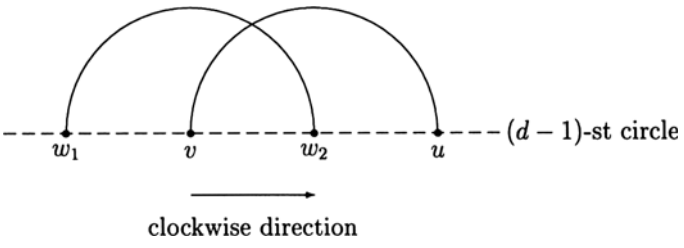

Fig. 3.25.

connected radii more than $n/3$ of nodes of V. Thus, the rest of $\overline{G}$ contains at most $2n/3$ nodes of G.

To find two initial candidates we look for an edge (v, u) between two nodes u and v of the last $(d-1)$-st circle such that the curve realizing (v, u) flows from v to u in the clockwise direction and the clockwise distance from v to u is the largest for all pairs of nodes of the $(d-1)$-st circle connected via an edge lying outside the $(d-1)$-st circle (The edge (u_3, u_4) has this property for the third circle of the graph representation in Figure 3.18). We claim v and u are "**free**" to the outside, which means that we can draw a line from $v(u)$ to a point at infinite distance that does not cross any edge of $\overline{G}$ (see Figure 3.24).

Now we prove this claim. Let us assume v is not free to the outside; i.e., there is an edge (w_1, w_2) between some nodes w_1 and w_2 on the $(d-1)$-st circle, where v is clockwise between w_1 and w_2 and w_1 is clockwise before v according to w_2. If w_2 is clockwise between v and u, then the curves realizing the edges (v, u) and (w_1, w_2) intersect, which contradicts the planarity of $\overline{G}$ (see Figure 3.25). If u is clockwise between v and w_2, then this contradicts the maximality of (v, u) because the distance between w_1 and w_2 on the circle is greater than the distance between v and u (see Figure 3.26).

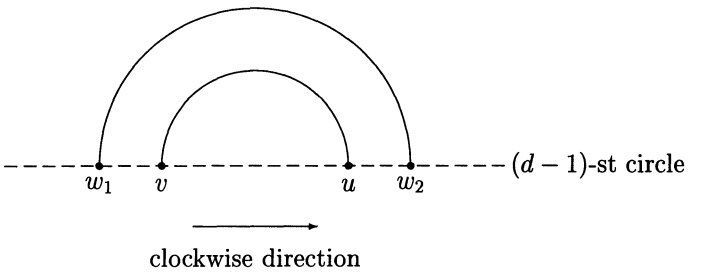

Fig. 3.26.

Obviously, the same argument works to show that u is free to the outside. We note that if there is no non-ring edge outside the $(d-1)$-st circle, then every node of the $(d-1)$-st circle is free to the outside. In such a case we choose u and v arbitrarily.

Now we choose one radius P_u starting in u and one radius P_v starting in v (see Figure 3.24). According to the definition of the concentric presentation, each node on a circle is directly connected to a node on the next lower ring. Thus starting from $v(u)$ we find a path from v to a node on the $(l+1)$-st circle containing exactly one node of each circle of $\overline{G}$. If there are several such paths from v to the $(l+1)$-st circle, it does not matter which one we choose. The only restriction we have is that P_v and P_u do not cross. Since we have nothing against the existence of common parts of P_v and P_u this restriction is no obstacle for the construction of P_v and P_u.

We observe that the initial radius P_v and P_u are candidates dividing $\overline{G}$ into $H(P_u, P_v)$ and $H(P_v, P_u)$ because v and u are free to the outside (no edge between $H(P_u, P_v)$ and $H(P_v, P_u)$ flows outside the last circle of $\overline{G}$) and there exists no edge connecting the nodes of the $(l+1)$-st circle and running below the $(l+1)$-st circle.

In fact it is not necessary that v and u are free to the outside. We say a node w of the outside circle of $H(P_v, P_u)$ is **free to outside according to $H(P_u, P_v)$** if w is free to outside after removing all edges leading between the nodes of the outside circle of $H(P_u, P_v)$. We observe that two non-crossing radii P_u and P_v are candidates for S' ($P_u \cup P_v$ is a vertex-cut of $\overline{G}$) if and only if v and u are free to outside according to $H(P_u, P_v)$.

If none of $H(P_u, P_v)$ and $H(P_v, P_u)$ constructed above contains more than $2n/3$ nodes of V we are ready. Without loss of generality we assume $H(P_v, P_u)$ contains more then $2n/3$ nodes. The aim is to find a new radius $P \subseteq H(P_v, P_u) \cup P_v \cup P_u$ such that P_v and P as well as P and P_u are candidates. If P differs from (is not identical with) P_v and from P_u, then $H(P_v, P)$ and $H(P, P_u)$ are smaller than $H(P_v, P_u)$. Since the number of nodes of $\overline{G}$ is finite, a finite number of steps is sufficient to find an "almost balanced" partition of $\overline{G}$ according to V.

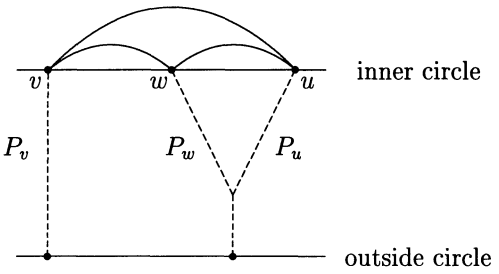

Fig. 3.27. The search for a new radii P_w if the node w is free to the outside according to $H(P_v, P_u)$

To complete the proof it remains to prove the existence of a radius P described above. In what follows we distinguish the following two cases:

(2.1) Let the part of the outside circle of $H(P_v, P_u)$ (the part between v and u in the clockwise direction) contain at least three nodes. For the same reason as given in the initial search for u and v, there exists a node w on the outside circle that is free to the outside according to $H(P_u, P_v)$. Consider a radius P_w as a path from w to the inside ring of $H(P_v, P_u)$ lying clockwise between P_v and P_u in $H(P_v, P_u)$. (Note that common parts between P_w

and P_u (P_v) are allowed.) Then the pair P_v, P_w as well as the pair P_w, P_u are candidates (vertex-cuts of $\overline{G}$.)

(2.2) Let the part of the outside circle of $H(P_v, P_u)$ contain at most two nodes. This means that the outside circle of $H(P_v, P_u)$ contains only the nodes u and v, which may be even the same. Let $P_v = v, v_1, \ldots, q$, $P_u = u, u_1, \ldots, p$, $P_{v_1} = v_1, \ldots, q$, and $P_{u_1} = u_1, \ldots, p$ (see Figure 3.28), i.e., $P_v = v, P_{v_1}$ and $P_u = u, P_{u_1}$. We remove the nodes v and u from $H(P_v, P_u)$ and we remove the whole outside circle of $\overline{G}$ to get the subgraph $\overline{G}_1$ of $\overline{G}$. Further, we commit u and v to S', i.e., we partially specify S'. First we observe that P_{v_1} and P_{u_1} are a vertex-cut of $\overline{G}_1$, i.e., candidates for the separation of $\overline{G}_1$. Secondly we observe that all paths (connections) leading between $H(P_{v_1}, P_{u_1})$ and the rest of $\overline{G}$ and having some part outside the circle containing v_1 and u_1 must contain either v or u. Using these facts we can conclude that $\{u, v\}$ together with any candidates P', P'' for the separation of $\overline{G}_1$ lying between P_{v_1} and P_{u_1} in $H(P_{v_1}, P_{v_2})$ is a vertex-cut of $\overline{G}$, i.e., uP' and vP'' are candidates for the separation of $\overline{G}$.

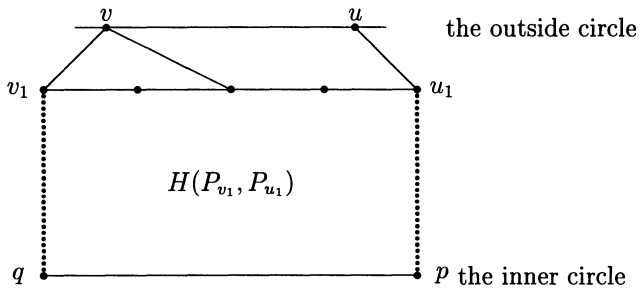

Fig. 3.28. The reduction of the middle part if $H(P_v, P_u)$ contains at most two nodes in the outside circle

Repeatedly using one of the steps (2.1) and (2.2) we achieve a required partition S' of $\overline{G}$ containing at most two nodes of any circle of $\overline{G}$. So $|S'| \leq 2 \cdot (d - h - 1) \leq 2 \cdot \sqrt{n}/\sqrt{2} = \sqrt{2} \cdot \sqrt{n}$, and $|S| \leq 2|S'| \leq 2 \cdot \sqrt{2}\sqrt{n}$. Since the vertex-cut achieved is $S \cup V_d \cup V_h$ we obtain $|S \cup V_d \cup V_h| \leq 2 \cdot \sqrt{2}\sqrt{n} + 2 \cdot \sqrt{2}\sqrt{n} = 4 \cdot \sqrt{2}\sqrt{n}$.

□

Now we are able to formulate the well-known Planar Separator theorem.

Theorem 3.4.5.8 *Every planar graph has a strong $O(\sqrt{m})$ vertex-separator.*

Proof. Let G be a planar graph of m nodes. By Lemma 3.4.5.7 we can find an almost balanced vertex-cut of G of at most $4 \cdot \sqrt{2}\sqrt{m}$ nodes. Since the obtained components are planar too, we can use Lemma 3.4.5.7 recursively. Thus G has a $4 \cdot \sqrt{2}\sqrt{m}$ vertex-separator. This implies by Corollary 3.4.2.10 that G has a strong $O(\sqrt{m})$ vertex-separator. □

Corollary 3.4.5.9 *Every planar graph of degree at most 4 has a strong $O(\sqrt{m})$ edge-separator.*

Since we have shown that planar graphs have a strong $O(\sqrt{m})$ vertex-separator we can apply the lower bound method of Section 3.4.3 to get quadratic lower bounds on the combinational complexity of planar Boolean circuits computing specific Boolean functions having linear communication complexity.

Theorem 3.4.5.10 *Let, for any $n \in \mathbb{N}$, S_n be a planar Boolean circuit computing a Boolean function $f_n: \{0,1\}^n \to \{0,1\}$. Then:*

$$\mathrm{CC}(S_n) = \Omega((\mathrm{acc}(f_n))^2).$$

Proof. Following Theorem 3.4.3.6 we see that each Boolean circuit having a strong $s(m)$ vertex-separator fulfills

$$4 \cdot Z_s(\mathrm{CC}(S_n), n) \geq \mathrm{acc}(f_n).$$

Since S_n is planar we obtain from Theorem 3.4.5.8 that $s(m) = O(\sqrt{m})$. By Proposition 3.4.3.5 $Z_s(m,n) = O(\sqrt{m})$ for $s(m) = O(\sqrt{m})$. Thus $\mathrm{CC}(S_n) = \Omega((\mathrm{acc}(f_n))^2)$. □

Defining the combinational complexity of Boolean circuits we have always considered semilective Boolean circuits. The reason is that allowing multilectivity of inputs cannot help to decrease the combinational complexity of any Boolean function. The situation may change rapidly if one restricts the Boolean circuit model in some way. This is also the case for planar Boolean circuit models. We do not want to deal with the comparison of the computational power of (semilective) planar Boolean circuits and the power of multilective planar Boolean circuits here. But we call attention to Exercises 3.4.6.12 and 3.4.6.13 as well to Problems 3.4.7.2 and 3.4.7.3 dealing with this comparison problem.

3.4.6 Exercises

Exercise 3.4.6.1 *Prove that every binary tree has a strong $O(\log n)$ edge-separator.*

Exercise 3.4.6.2 *Prove that each graph having an $O(1)$ edge-separator has a strong $O(\log_2 n)$ edge-separator.*

Exercise 3.4.6.3 *Estimate separators for two-dimensional grids (lattices) of a size $a \times b$ for arbitrary $a, b \in \mathbb{N}$.*

Exercise 3.4.6.4 *Which separator (strong separator) has a d-dimensional grid of the size $a \times a \times \ldots \times a$ for any $a \in \mathbb{N}$?*

Exercise 3.4.6.5 *For the tree in Figure 3.5 write a partition tree distinct from the partition tree in Figure 3.16.*

Exercise 3.4.6.6 *Write a partition tree for the 2-dimensional grid of size 3×7.*

Exercise 3.4.6.7 *Prove that any planar Boolean circuit of depth t must have $\Omega(n^2)$ gates to compute a Boolean function of a linear $(t-1)$-round communication complexity.*

Exercise 3.4.6.8 *Extend Theorem 3.4.3.6 for s-communication complexity.*

Exercise 3.4.6.9 * *Search for a version of Theorem 3.4.3.6 working for unbounded fan-in Boolean circuits.*

Exercise 3.4.6.10 ** *Construct, for some fixed constants $d \geq 3$ and c, a class of (m, d, c) magnifiers for any $m \in \mathbb{N} - \{0\}$.*

Exercise 3.4.6.11 * *Prove that each 3-regular graph of at least six nodes is 3-colorable.*

Exercise 3.4.6.12 * *Find a nonnegative integer $k \geq 2$, and a sequence $\{f_n\}_{n=1}^{\infty}$ of Boolean functions, $f_n : \{0,1\}^n \to \{0,1\}$, such that one can compute $\{f_n\}_{n=1}^{\infty}$ with k-multilective planar Boolean circuits much more efficiently than with (semilective) planar Boolean circuits.*

Exercise 3.4.6.13 ** *Let k be a positive integer. Give a sequence of Boolean functions $F = \{f_n\}_{n=1}^{\infty}$, $f_n : \{0,1\}^n \to \{0,1\}$, such that any sequence $\{B_n\}_{n=1}^{\infty}$ of k-multilective planar Boolean circuits computing F satisfies $\mathrm{CC}(B_n) = \Omega(n \cdot \log_2 n)$.*

Exercise 3.4.6.14 * *Improve the result of Lemma 3.4.5.7 by showing that each planar graph has a $2 \cdot \sqrt{2}\sqrt{n}$ vertex-separator.*

3.4.7 Problems

Problem 3.4.7.1 ** *Prove a nonlinear lower bound on the combinational complexity of some specific sequence* $\{f_n\}_{n=1}^{\infty}$ *of Boolean functions with* $f_n : \{0,1\}^n \to \{0,1\}$. *Almost all Boolean functions with a linear communication complexity are possible candidates for this result. Note that at present we do not have any nonlinear lower bound on any computing problem* $\{P_n^n\}_{n=1}^{\infty}$, $P_n : \{0,1\}^n \to \{0,1\}^n$ *for every* $n \in \mathbb{N}$. *Observe that according to the results of this section the main difficulty is in proving that random structures are not powerful enough to compute a specific function with a linear number of gates.*

Problem 3.4.7.2 * *Prove a lower bound* $\Omega(g(n))$ *on the combinational complexity of multilective planar Boolean circuits computing a specific sequence of Boolean functions for a function* $g: \mathbb{N} \to \mathbb{N}$ *fulfilling* $\lim_{n\to\infty} n \log n/g(n) = 0$.

Problem 3.4.7.3 * *Find for every positive integer* $k \geq 2$ *a sequence of Boolean functions* F_k *such that the combinational complexity of* k*-multilective planar Boolean circuits computing* F_k *grows asymptotically faster than the combinational complexity of* $(k+1)$*-multilective planar Boolean circuits computing* F_k.

3.5 Lower Bounds on the Size of Unbounded Fan-in Circuits

3.5.1 Introduction

While Boolean circuits have to have at least linear size to compute Boolean functions depending on all their input variables, linear lower bounds on the size of unbounded fan-in Boolean circuits computing specific Boolean functions are not so obvious. The bases of unbounded fan-in circuits are infinite and so, for a given base, there are a lot of functions with constant combinational complexity. Here, we shall present a lower bound method providing nontrivial lower bounds on the sizes of unbounded fan-in circuits by different standard bases. The lower bounds achieved are in the best cases linear. Then we combine the above mentioned method with the method of Section 3.4 to get nonlinear lower bounds for unbounded fan-in circuits with sublinear separators.

This section is organized as follows. Section 3.5.2 shows that the communication complexity of a Boolean function f divided by the maximum of some special communication complexities of functions of a base Bas provides a direct lower bound on the size of unbounded fan-in circuits over the base Bas. In Section 3.5.3 this technique is extended to get nonlinear lower bounds on the sizes of unbounded fan-in Boolean circuits having sublinear separators.

3.5.2 Method of Communication Complexity of Infinite Bases

The concept of the lower bound method presented here is an extension of the concepts of Sections 3.3 and 3.4 based on some cuts of Boolean circuits. The informal idea of this concept is as follows.

Let $S = (V, E)$ be an unbounded fan-in Boolean circuit over a base Bas, and let C be a vertex-cut of S dividing the input nodes of S in an almost balanced way. Let f be the Boolean function computed by S. The idea is to show that the cut provides an upper bound on $\mathrm{acc}(f)$ in some way. Clearly, the communication between the two parts of the circuits C_1 and C_2 divided by the cut C flows via the cut of C only (see Figure 3.29). A communication protocol can compute the function f in such a way that it computes the output values of all nodes in C in a topological order and gives these output values to both computers. In this way the left computer corresponding to C_1 gets all Boolean values flowing via directed edges from C to C_1 and the right computer corresponding to C_2 gets all Boolean values flowing via directed edges from C to C_2. Thus, our protocol can always compute the output, and its communication complexity is the sum of the lengths of communication messages used to compute and spread the outputs of the nodes of C. Note that such a communication length used to compute the output of a gate g may differ from $\mathrm{cc}(g)$ or $\mathrm{acc}(g)$. The reason for this is that the partition of inputs of this gate given by the cut C do not have to be almost balanced or even balanced. To handle this we define unbalanced communication complexity of g, $\mathrm{ucc}(g)$, as maximum over all partitions of input variables.

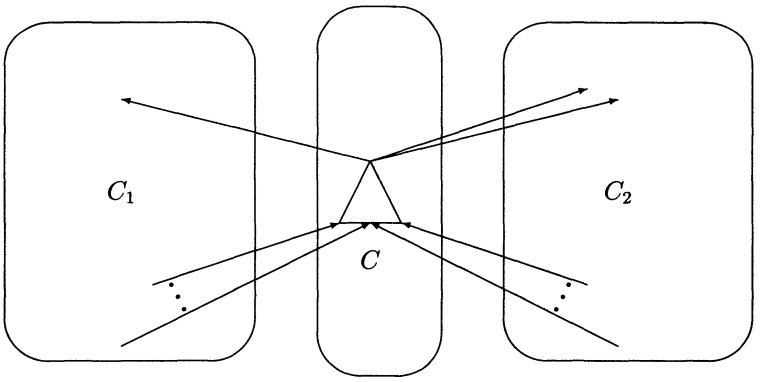

Fig. 3.29. The division of a circuit by removing the nodes of a vertex-cut C

Definition 3.5.2.1 *Let X be a set of input variables of a Boolean function $g : \{0,1\}^n \to \{0,1\}$. The* **unbalanced communication complexity** *of g is*

$$\mathbf{ucc}(g) = \max\{\mathrm{cc}(g, \Pi) \mid \Pi \text{ is a partition of } X\}.$$

Example 3.5.2.2 We illustrate Definition 3.5.2.1 by estimating the unbalanced communication complexity of $g_r^{\vee}(x_1, x_2, \ldots, x_r) = x_1 \vee x_2 \vee \ldots \vee x_r$ for any $r \in \mathbb{N} - \{0\}$. Let $X_r = \{x_1, x_2, \ldots, x_r\}$, and let Π be an arbitrary partition of X_r. Let $\Pi_{L,X} = \{x_{i_1}, \ldots, x_{i_z}\}$ and $\Pi_{R,X} = X - \Pi_{L,X}$ for some $z \in \mathbb{N}$. A protocol $\langle \Pi, \Phi \rangle$ computing $g_r^{\vee}$ proceeds as follows. For any input $\alpha = \alpha_1 \alpha_2 \ldots \alpha_r \in \{0,1\}^r$ the first computer sends the communication bit $\alpha_{i_1} \vee \alpha_{i_2} \vee \ldots \vee \alpha_{i_z}$ to the second computer. After that the second computer knows already the result $g_r^{\vee}(\alpha)$. So, we have shown that $\mathrm{ucc}(g_r^{\vee}) \leq 1$ for every $r \in \mathbb{N} - \{0\}$. □

Definition 3.5.2.3 *Let $S = (V, E)$ be an unbounded fan-in Boolean circuit computing a Boolean function f. A* **gate-cut of S** *is any vertex-cut (U, V_1, V_2) of the graph (V, E) such that U consists only of gates of S (i.e., U does not contain any input of S). The* **unbalanced communication complexity of the gate-cut (U, V_1, V_2)** *is*

$$\mathbf{ucc(}U, V_1, V_2\mathbf{)} = \sum_{g \in U} (\mathrm{ucc}(g) + 1).$$

Note that we add $+1$ to $\mathrm{ucc}(g)$ in the definition of $\mathrm{ucc}(U, V_1, V_2)$ because there exists a protocol which after the exchange of $\mathrm{ucc}(g) + 1$ communication bits secures that both computers know the output of the gate g.

Observation 3.5.2.4 *Let S be an unbounded fan-in circuit computing a Boolean function f defined over a set of input variables X. Let $\Pi \in \mathrm{Abal}(X)$ and let a gate-cut (U, V_1, V_2) be such that $\Pi_{L,X} \subseteq V_1$ and $\Pi_{R,X} \subseteq V_2$. Then*

$$\mathrm{cc}(f, \Pi) \leq \mathrm{ucc}(U, V_1, V_2).$$

Definition 3.5.2.5 *Let* Bas *be a (possibly infinite) set of functions. We define the* **communication complexity of Bas** *as a function*

$$\mathbf{ucc(Bas,n)} = \max\{\mathrm{ucc}(g) \mid g \in \mathrm{Bas} \cap B_2^n\} + 1.$$

In what follows we are mainly interested in bases whose communication complexity can be bounded by a constant. The next lemma claims that the fundamental infinite base $F = B_1^2 \cup \{g_r^{\triangle} \mid \text{for every } \triangle \in \{\vee, \wedge, \oplus\} \text{ and every } r \in \mathbb{N} - \{0\}\}$ has this property.

Lemma 3.5.2.6 $\mathrm{ucc}(F, r) \leq 2$ *for every $r \geq 2$.*

Proof. Obviously every Boolean function from B_1^2 has unbalanced communication complexity at most 1. In Example 3.5.2.2 we have shown that $\mathrm{ucc}(g_r^{\vee}) \leq 1$ for any positive integer r. Since $\oplus$ and $\wedge$ are commutative and associative operations, the same argument as that of Example 3.5.2.2 yields $\mathrm{ucc}(g_r^{\triangle}) \leq 1$ for every

$\triangle \in \{\vee, \wedge, \oplus\}$ and every positive integer r. Thus $\max\{\mathrm{ucc}(g) \mid g \in F \cap B_2^r\} = 1$ for any positive integer $r \geq 2$. □

Another intensively investigated base of unbounded fan-in Boolean circuits is

$$\begin{aligned}\textbf{Threshold} = \{&g_m^k \mid g_m^k : \{0,1\}^m \to \{0,1\} \text{ and } g_m^k(\alpha_1\alpha_2\ldots\alpha_m) = 1 \\ &\text{iff } \alpha_1 + \alpha_2 + \ldots + \alpha_m \geq k,\ k, m \in \mathbb{N} - \{0\},\ k \leq m\} \\ &\cup \{I\}.\end{aligned}$$

Unbounded fan-in Boolean circuits over the base Threshold are called **threshold circuits**.

Lemma 3.5.2.7 *For every positive integer* $n \geq 2$

$$\mathrm{ucc}(Threshold, r) \leq \lceil \log_2 r \rceil + 1.$$

Proof. It is sufficient to show $\mathrm{ucc}(g_r^k) \leq \lceil \log_2 r \rceil + 1$ for any $r, k \in \mathbb{N},\ k \leq r,\ r \geq 2$. Let $r, k,\ k \leq r,\ r \geq 2$ be two positive integers. Let $d = \lceil \log_2 r \rceil$. Let $X_r = \{x_1, x_2, \ldots, x_r\}$ be the set of input variables of the Boolean function g_r^k. Let Π be an arbitrary partition of X_r, $\Pi_{L,X} = \{x_{i_1}, \ldots, x_{i_z}\}$ for some $z \leq r$. A one-way protocol $\langle \Pi, \Phi \rangle$ computing g_r^k proceeds as follows. For any input $\alpha = \alpha_1\alpha_2\ldots\alpha_r \in \{0,1\}^r$ the first computer sends the following d communication bits $BIN_d^{-1}(\sum_{j=1}^{z} \alpha_{i_j})$. Now the second computer has enough information to compute $g_r^k(\alpha)$. □

We are now prepared to formulate the lower bound method on the combinational complexity of unbounded fan-in Boolean circuits.

Theorem 3.5.2.8 *Let* S *be an unbounded fan-in Boolean circuit over a base* Bas *computing a Boolean function* $f \in B_2^n$ *with the set* X *of input variables. Then*

$$\mathrm{CC}(S) \geq \max\{\mathrm{cc}(f, \Pi) \mid \Pi \in \mathrm{Abal}(X)\} / \mathrm{ucc}(\mathrm{Bas}, \frac{3n}{2}).$$

Proof. Let $S = (V, E)$. First we observe that we can assume the indegree of S is bounded by $3n/2$ (if not, then S has at least $n/2$ gates and so $\mathrm{CC}(S) \geq n/2 \geq \max\{\mathrm{cc}(f, \Pi) \mid \Pi \in \mathrm{Abal}(X)\}$).

Now it suffices to show $\mathrm{CC}(S) \geq \mathrm{cc}(f, \Pi)/\mathrm{ucc}(\mathrm{Bas}, \frac{3n}{2})$ for every $\Pi \in \mathrm{Abal}(X)$. Let Π be an arbitrary almost balanced partition of X. Then, for every gate-cut (U, V_1, V_2) of $S = (V, E)$ with the property $\Pi_{L,X} \subseteq V_1$ and $\Pi_{R,X} \subseteq V_2$,

$$\mathrm{ucc}(U, V_1, V_2) \geq \mathrm{cc}(f, \Pi)$$

because in the opposite case one can construct a protocol $\langle \Pi, \Phi \rangle$ computing f with $\mathrm{cc}(\langle \Pi, \Phi \rangle) < \mathrm{cc}(f, \Pi)$. Obviously $\mathrm{CC}(S) \geq |U|$ and $\mathrm{ucc}(U, V_1, V_2) \leq |U| \cdot \mathrm{ucc}(\mathrm{Bas}, \frac{3n}{2})$ (note that the indegree of gates of S is bounded by $3n/2$). So

$$\begin{aligned} \mathrm{CC}(S) \geq |U| &\geq \mathrm{ucc}(U, V_1, V_2)/\mathrm{ucc}(\mathrm{Bas}, \frac{3n}{2}) \\ &\geq \mathrm{cc}(f, \Pi)/\mathrm{ucc}(\mathrm{Bas}, \frac{3n}{2}). \end{aligned}$$

□

Corollary 3.5.2.9 *For any positive integer n, and for any Boolean function $f \in B_2^n$ defined over a set of input variables X,*

$$\text{unfi-CC}(f) \geq \max\{\mathrm{cc}(f, \Pi) \mid \Pi \in \mathrm{Abal}(X)\}/2.$$

Proof. The result follows from Theorem 3.5.2.8 and from Lemma 3.5.2.6 yielding $\mathrm{ucc}(F, r) \leq 2$ for every positive integer $r \geq 2$. □

Corollary 3.5.2.10 *Let S be a threshold circuit computing a Boolean function $f \in B_2^n$ defined over a set of input variables X. Then*

$$\mathrm{CC}(S) \geq \max\{\mathrm{cc}(f, \Pi) \mid \Pi \in \mathrm{Abal}(X)\}/\lceil \log_2(3n) \rceil.$$

Proof. The result is a direct consequence of Theorem 3.5.2.8 and of Lemma 3.5.2.7 which claims $\mathrm{ucc}(\mathit{Threshold}, r) \leq \lceil \log_2 r \rceil + 1$ for any positive integer $r \geq 2$. □

Concluding this subsection we call attention to the two following advantages of the lower bound method introduced over the method used in Sections 3.3 and 3.4. First, it is sufficient to obtain a lower bound on $\max\{\mathrm{cc}(f, \Pi) \mid \Pi \text{ is a balanced partition}\}$ instead of proving a lower bound on $\mathrm{acc}(f) = \min\{\mathrm{cc}(f, \Pi) \mid \Pi \text{ is a balanced partition}\}$. As we already know, the main difficulty in the use of communication complexity is to prove a lower bound on $\mathrm{cc}(f, \Pi)$ for all (almost) balanced partitions Π. In most cases, to find a partition Π with a large $\mathrm{cc}(f, \Pi)$ is much easier than to prove that $\mathrm{cc}(f, \Pi)$ are large for all Π. On the other hand, there are many Boolean functions f with small $\mathrm{cc}(f)$, but with large $\max\{\mathrm{cc}(f, \Pi) \mid \Pi \text{ is a balanced partition of the set of input variables of } f\}$.

The second advantage of the lower bound method of this section is that this method does not relate the communication complexity of a vertex-cut of a circuit to the number of edges adjacent to the nodes of the cut as the method of Section 3.4 does. Obviously the number of edges adjacent to the gates of a gate-cut of an unbounded fan-in Boolean circuit may be much larger than the unbalanced communication complexity of the gate cut defined in Definition 3.5.2.3. Whereas considering the communication between the two parts of the circuit as the sequence of bits flowing via the edges of the cut would provide in many cases very small lower bounds on the the size of the circuit, taking the unbalanced communication complexity of the gates of the cut provides for many bases even linear lower bounds.

3.5.3 Unbounded Fan-in Circuits with Sublinear Vertex-Separators

The aim of this subsection is to show that the lower bound method of Section 3.4.3 can be extended for unbounded fan-in Boolean circuits. More precisely, we show nonlinear lower bounds on the combinational complexity of unbounded fan-in Boolean circuits with sublinear vertex-separators and bounded communication complexity of their bases.

Theorem 3.5.3.1 *Let f be a Boolean function from B_2^n for some positive integer n. Let S be an unbounded fan-in Boolean circuit over a base* Bas *computing f. If S has a strong $s(m)$ vertex-separator for a function $s : \mathbb{N} \to \mathbb{N}$, then*

$$\mathrm{ucc}(\mathrm{Bas}, n + \mathrm{CC}(S)) \cdot Z_s(\mathrm{CC}(S), n) \geq \mathrm{acc}(f_n).$$

Proof. Let X be the set of input variables of f. Lemma 3.4.3.2 independently on the degree of the circuit claims that there exists an almost balanced vertex-cut (U, V_1, V_2) of G such that

(i) (U, V_1, V_2) is almost balanced according to X, and

(ii) $|U| \leq Z_s(\mathrm{CC}(S), n)$.

Now it suffices to describe a protocol computing f within communication complexity $|U| \cdot \mathrm{ucc}(\mathrm{Bas}, n + \mathrm{CC}(S))$. We consider the nodes in U in a topological order which gives the order in which communication bits flow between the two parts of the circuit S given by the vertex-cut (U, V_1, V_2). If a node $v \in U$ is an input of S, then v is either in V_1 or in V_2. In both cases we need to send only one bit containing the value of the variable assigned to v to the other part of the circuit. If a node $g \in U$ is a gate, then the communication of the optimal protocol computing g according to the partition of inputs of g given by the cut (U, V_1, V_2) is realized. Note that the number of inputs of g is at most $n + \mathrm{CC}(S)$. As we already know, this means at most $\mathrm{ucc}(g) \leq \mathrm{ucc}(\mathrm{Bas}, n + \mathrm{CC}(S)) - 1$ bits. One additional bit is needed to secure that both parts of S know the result computed by the gate g. □

We illustrate the use of the lower bound method given in Theorem 3.5.3.1 by formulating lower bounds for two specific bases.

Theorem 3.5.3.2 *Let f be a Boolean function from B_2^n for some positive integer n. Let S be an unbounded fan-in Boolean circuit over the base F computing f. If S has a strong $s(m)$ vertex-separator for a function $s : \mathbb{N} \to \mathbb{N}$, then*

$$Z_s(\mathrm{CC}(S), n) \geq \mathrm{acc}(f)/2.$$

Proof. The result follows directly from Theorem 3.5.3.1 because $\mathrm{ucc}(F, r) \leq 2$ for every $r \in \mathbb{N}$. □

Corollary 3.5.3.3 *Let $s, h : \mathbb{N} \to \mathbb{N}$ be monotone functions such that $s(m) = m/h(m) = o(m)$ and $h(m) = O((\log_2 m)^k)$ for a positive integer k. Let $\{f_n\}_{n=1}^{\infty}$ be a sequence of Boolean functions, $f_n : \{0,1\}^n \to \{0,1\}$, and let, for every $n \in \mathbb{N}$, S_n be an unbounded fan-in circuit over F computing f_n. If, for every $n \in \mathbb{N}$, S_n has a strong $s(m)$ vertex-separator, then*

$$\mathrm{CC}(S_n) = \Omega(\mathrm{acc}(f_n) \cdot h(n)).$$

Proof. According to Proposition 3.4.3.3 we have

$$Z_s(\mathrm{CC}(S_n), n) = O(\mathrm{CC}(S_n)/h(\mathrm{CC}(S_n))).$$

Inserting this in the claim of Theorem 3.5.3.2 we obtain the result.

Corollary 3.5.3.4 *Let $s(m) = k \cdot m^b$ for some constants $k \geq 1$, $0 < b < 1$. Let, for any positive integer n, S_n be an unbounded Boolean circuit over F computing a Boolean function $f_n : \{0,1\}^n \to \{0,1\}$. If $\mathrm{acc}(f_n) = \Omega(n)$ and, for every $n \in \mathbb{N} - \{0\}$, S_n has a strong $s(m)$ vertex-separator, then*

$$\mathrm{CC}(S_n) = \Omega(n^{1/b}).$$

Proof. According to Proposition 3.4.3.5 we have the upper bound

$$Z_S(\mathrm{CC}(S_n), n) = O((\mathrm{CC}(S_n))^b).$$

Applying Theorem 3.5.3.2 we obtain $(\mathrm{CC}(S_n))^b = \Omega(n)$. □

We still formulate the lower bounds for threshold circuits.

Theorem 3.5.3.5 *Let, for every positive integer n, f_n be a Boolean function from B_2^n. Let, for every positive integer n, S_n be a threshold circuit computing f_n, and let S_n have a strong $s(m)$ vertex-separator for some monotone function $s : \mathbb{N} \to \mathbb{N}$. Then*

(i) if $s(m) = m/h(m) = o(m)$, $h(m) = O((\log_2 m)^k)$ for a positive integer k, and h is monotone, then

$$\mathrm{CC}(S_n) = \Omega(\mathrm{acc}(f_n) \cdot h(n)/\log_2 n)$$

(ii) if $s(m) = k \cdot m^b$ for some constants $k \geq 1$, $0 < b < 1$, then

$$\mathrm{CC}(S_n) = \Omega((\mathrm{acc}(f_n)/\log_2 n)^{1/b}).$$

Proof. Theorem 3.5.3.1 claims

$$\mathrm{ucc}(Threshold, n + \mathrm{CC}(S_n)) \cdot Z_s(\mathrm{CC}(S_n), n) \geq \mathrm{acc}(f_n),$$

and Lemma 3.5.2.7 claims

$$\mathrm{ucc}(Threshold, r) \leq \lceil \log_2 r \rceil + 1$$

for every positive integer $r \geq 2$.

In case *(i)* we can assume that $\mathrm{CC}(S_n) \leq n^2$ because in the opposite case we already have a higher lower bound than the lower bound provided by *(i)*. Thus $\mathrm{ucc}(Threshold, n + \mathrm{CC}(S_n)) \leq \mathrm{ucc}(Threshold, n + n^2) \leq 3 \cdot \lceil \log_2 n \rceil + 3$. Since according to Proposition 3.4.3.3 we have $Z_S(\mathrm{CC}(S_n), n) = O(\mathrm{CC}(S_n)/h(\mathrm{CC}(S_n)))$, we obtain $(\log_2 n) \cdot \mathrm{CC}(S_n)/h(\mathrm{CC}(S_n)) = \Omega(\mathrm{acc}(f_n))$. This directly implies $\mathrm{CC}(S_n) = \Omega(\mathrm{acc}(f_n) \cdot h(n)/\log_2 n)$ because $h(n) = \Theta(h(\mathrm{CC}(S_n)))$.

For case *(ii)* we may assume $\mathrm{CC}(S_n) = O(n^{1/b})$. Because of this we have $\mathrm{ucc}(Threshold, n + \mathrm{CC}(S_n)) \leq \mathrm{ucc}(Threshold, O(n^{1/b})) = O(\log_2 n)$. Now it is sufficient to observe that $Z_S(\mathrm{CC}(S_n), n) = O((\mathrm{CC}(S_n))^b)$ according to Proposition 3.4.3.5. □

3.5.4 Exercises

Exercise 3.5.4.1 *Estimate the unbalanced communication complexity of the following languages:*

(i) $\tilde{L} = \{x \in \{0,1\}^+ \mid \#_0(x) = \#_1(x)\}$,

(ii) $S_m = \{xy \mid |x| = |y|,\ x, y \in \{0,1\}^+,\ BIN(x) < BIN(y)\}$,

(iii) $E_g = \{w \in \{0,1\}^* \mid w = uu\}$.

Exercise 3.5.4.2 * *Estimate* $\mathrm{ucc}(\mathrm{Bas}, n)$ *for every* $n \in \mathbb{N}$ *and the following bases* Bas*:*

(i) Bas *is the set of all symmetric Boolean functions.*

(ii) Bas *is the set of all monotone Boolean functions.*

Exercise 3.5.4.3 *Give a formal proof of Observation 3.5.2.4 containing a formal description of a protocol corresponding to the gate-cut* (U, V_1, V_2).

Exercise 3.5.4.4 *Prove for unbounded fan-in Boolean circuits with the base of all symmetric functions a similar result as Corollary 3.5.2.10 gives for threshold circuits.*

Exercise 3.5.4.5 * *Define unbalanced one-way communication complexity of gates and of bases. Then*

(i) prove some versions of Theorem 3.5.2.8, Corollary 3.5.2.9, and Corollary 3.5.2.10 using unbalanced one-way communication complexity of bases instead of unbalanced communication complexity of bases.

(ii) bound the number of rounds of a protocol communicating the messages of one-way protocols computing the gates of a gate-cut of an unbounded fan-in circuit S by twice the depth of S.

(iii) use (i) and (ii) and k-rounds communication complexity to formulate lower bounds on the communication complexity of unbounded fan-in Boolean circuits with bounded depth.

Exercise 3.5.4.6 *Which lower bounds can one obtain for unbounded fan-in planar circuits?*

3.5.5 Problems

Problem 3.5.5.1 *Let, for arbitrary positive rational numbers $r_1, r_2, \ldots, r_n$ and k, $g_k^{r_1,r_2,\ldots,r_n} : \{0,1\}^n \to \{0,1\}$ be a Boolean function defined as*

$$g_k^{r_1,r_2,\ldots,r_n}(\alpha_1, \alpha_2, \ldots, \alpha_n) = 1 \text{ iff } r_1 \cdot \alpha_1 + r_2 \cdot \alpha_2 + \ldots + r_n \cdot \alpha_n \geq k.$$

Estimate $\mathrm{ucc}(g_k^{r_1,r_2,\ldots,r_n})$ *and* $\mathrm{ucc}(Tr, n)$ *for every $n \in \mathbb{N}$ and the base of generalized threshold gates $Tr = \{g_k^{r_1,r_2,\ldots,r_n} \mid n \in \mathbb{N},\ r_1, r_2, \ldots, r_n, k$ are positive rational numbers$\}$.*

3.6 Lower Bounds on the Depth of Boolean Circuits

3.6.1 Introduction

This section differs from the previous ones of this chapter in the way communication complexity is used to get some lower bounds. In all previous sections we have used the communication complexity of a Boolean function f to get some lower bounds on the area of f or on the combinational complexity of f. Here, we do not know how to directly use (or whether it is even possible to use) $\mathrm{cc}(f)$ to get a lower bound on the depth complexity of f. What we are doing is making another use of the (communication) protocol model. For every Boolean function f, we define a specific relation $\mathrm{R}(f)$ depending on f and show that the communication complexity of computing $\mathrm{R}(f)$ by protocols is very strongly connected with $\mathrm{D}_{\{\vee,\wedge,\neg\}}(f)$. Using another relation $\mathrm{MR}(f)$ we obtain a method for proving lower bounds on the depth of monotone Boolean circuits.

This section is organized as follows. Section 3.6.2 contains the definition of monotone Boolean circuits and some fundamental observations about the

depth of (monotone) Boolean circuits. Section 3.6.3 gives the definition of the above mentioned relations $R(f)$ and $MR(f)$ for every Boolean function f, and introduces the communication complexity of $R(f)$ and $MR(f)$ respectively. The lower bound methods on the depth of Boolean circuits over the base $\{\vee, \wedge, \Gamma\}$ and of monotone Boolean circuits are presented in Section 3.6.4.

3.6.2 Monotone Boolean Circuits

In this section we give the definitions of monotone Boolean functions and monotone Boolean circuits.

Definition 3.6.2.1 *Let n be a positive integer, and let $f \in B_2^n$. We say that f is* **monotone** *if for every $\alpha, \beta \in \{0,1\}^n$, $\alpha \leq \beta$ implies $f(\alpha) \leq f(\beta)$.*

Observation 3.6.2.2 *Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables, and let $\{i_1, i_2, \ldots, i_r\} \subseteq \{1, \ldots, n\}$ for some positive integer $r \leq n$. Then $f(x_1, \ldots, x_n) = x_{i_1} \wedge x_{i_2} \wedge \ldots \wedge x_{i_r} \in B_2^n$ is monotone and $N^1(f) = \{\alpha \in \{0,1\}^n \mid \alpha \geq (\beta_1, \beta_2, \ldots, \beta_n)$, where $\beta_{i_j} = 1$ for $j \in \{1, \ldots, r\}$ and $\beta_k = 0$ for $k \notin \{i_1, i_2, \ldots, i_r\}\}$.*

Definition 3.6.2.3 *Let $f \in B_2^n$ be a monotone Boolean function over $X = \{x_1, \ldots, x_n\}$ for some positive integer n. A vector $\alpha \in \{0,1\}^n$ is called a* **lower one of f** *if*

(i) $f(\alpha) = 1$,

(ii) for all $\beta \in \{0,1\}^n$, $\beta \geq \alpha$ *implies* $f(\beta) = 1$, *and*

(iii) for all $\gamma \in \{0,1\}^n$, $\gamma < \alpha$ *implies* $f(\gamma) = 0$.

The set of all lower ones of f is denoted $\mathbf{LowN^1(f)} = \{\alpha \in \{0,1\}^n \mid \alpha$ *is a lower one of $f\}$. For any $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \{0,1\}^n$ we define* $\boldsymbol{S^1(\alpha)} = \{k \mid \alpha_k = 1\}$ *and* $\boldsymbol{S^0(\alpha)} = \{j \mid \alpha_j = 0\}$. *For any $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathrm{LowN}^1(f)$ we define*

$$\boldsymbol{f_\alpha^1(x_1, x_2, \ldots, x_n)} = \bigwedge_{i \in S^1(\alpha)} x_i.$$

For every $\alpha \in \mathrm{LowN}^1(f)$, the set $S^1(\alpha)$ is called a **minterm of f**. $\mathbf{Min(f)} = \{S^1(\alpha) \mid \alpha \in \mathrm{LowN}^1(f)\}$.

A vector $\omega \in \{0,1\}^n$ is called an **upper zero of f** *if*

(i) $f(\omega) = 0$,

(ii) for all $\beta \in \{0,1\}^n$, $\beta \leq \omega$ *implies* $f(\beta) = 0$, *and*

(iii) for all $\gamma \in \{0,1\}^n$, $\gamma > \omega$ *implies* $f(\gamma) = 1$.

The set of all upper zeros of f is denoted $\mathbf{UpN^0(f)}= \{\omega \in \{0,1\}^n \mid \omega$ *is an upper zero of* $f\}$. *For every* $\alpha \in \mathrm{UpN}^0(f)$, *the set* $S^0(\alpha)$ *is called a* **maxterm of** $\boldsymbol{f}$. $\mathbf{Max(f)} = \{S^0(\alpha) \mid \alpha \in \mathrm{UpN}^0(f)\}$.

We observe that each monotone Boolean function f can be unambiguously defined by giving $\mathrm{LowN}^1(f)$ or $\mathrm{UpN}^0(f)$.

Lemma 3.6.2.4 *For every monotone Boolean function* $f \in B_2^n$ *over an* $X = \{x_1, \ldots, x_n\}$,

$$f(x_1, \ldots, x_n) = \bigvee_{\alpha \in \mathrm{LowN}^1(f)} f_\alpha^1(x_1, \ldots, x_n).$$

Proof. Following Definition 3.6.2.3 we have

$$N^1(f) = \bigcup_{\alpha \in \mathrm{LowN}^1(f)} \{\beta \in \{0,1\}^n \mid \beta \geq \alpha\}.$$

Since $N^1(f_\alpha^1) = \{\beta \in \{0,1\}^n \mid \beta \geq \alpha\}$, the claim is proved. □

Observation 3.6.2.5 *Let* f *be a monotone Boolean function. For every* $Z_1 \in \mathrm{Max}(f)$ *and every* $Z_2 \in \mathrm{Min}(f)$, $Z_1 \cap Z_2 \neq \emptyset$.

In what follows we define monotone Boolean circuits and we show that they compute exactly the monotone Boolean functions.

Definition 3.6.2.6 *Each straight-line Boolean program over the basis* $\{\vee, \wedge\}$ *is called* **monotone**. *Each Boolean circuit over the basis* $\{\vee, \wedge\}$ *is called* **monotone**.

Lemma 3.6.2.7 *For every Boolean function* f, *there is a monotone Boolean circuit computing* f.

Proof. Let f be a Boolean function over a set $X = \{x_1, \ldots, x_n\}$ of input variables. Lemma 3.6.2.4 claims that f can be written as

$$f(x_1, \ldots, x_n) = \bigvee_{\alpha \in \mathrm{LowN}^1(f)} f_\alpha^1(x_1, \ldots, x_n),$$

where $f_\alpha^1(x_1, \ldots, x_n) = \wedge_{j \in S(\alpha)} x_j$ for every $\alpha \in \mathrm{LowN}^1(f)$. Since this formula contains only disjunctions and conjunctions, the straightforward construction of a Boolean circuit computing $f(x_1, \ldots, x_n)$ results in a monotone circuit. □

Lemma 3.6.2.8 *Every Boolean function* f *computed by a monotone Boolean circuits is monotone.*

Proof. We prove this lemma by induction on the depth of the circuit. Obviously, a monotone Boolean circuit of depth 1 computes either $x_1 \vee x_2$ or $x_1 \wedge x_2$, which

are monotone Boolean functions. Let every monotone Boolean circuit of depth $d \leq j$ compute a monotone Boolean function. We prove it for the depth $j+1$ too. Let S be a monotone Boolean circuit of depth $j+1$ and let g be an output gate with the distance $j+1$ to at least one input node. Let $x_1, \ldots, x_n$ be the inputs of S. We have to show that Result(g) is a monotone function. We distinguish two cases:

(1) if $g = (\vee, f_1, f_2)$, then Result$(g) = \tilde{g}(x_1, \ldots, x_n) = f_1(x_1, \ldots, x_n) \vee f_2(x_1, \ldots, x_n)$. Since f_1 and f_2 are monotone according to the induction hypothesis, we obtain for every $\alpha, \beta \in \{0,1\}^n$, $\alpha \leq \beta$,

$$\begin{aligned} \tilde{g}(\alpha_1, \ldots, \alpha_n) &= f_1(\alpha_1, \ldots, \alpha_n) \vee f_2(\alpha_1, \ldots, \alpha_n) \\ &\leq f_1(\beta_1, \ldots, \beta_n) \vee f_2(\beta_1, \ldots, \beta_n) \\ &= \tilde{g}(\beta_1, \ldots, \beta_n). \end{aligned}$$

(2) if $g = (\wedge, g_1, g_2)$, then Result$(g) = \tilde{g}(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_n) \wedge g_2(x_1, \ldots, x_n)$. Since g_1 and g_2 are monotone according to the induction hypothesis, we obtain $\tilde{g}(\alpha) \leq \tilde{g}(\beta)$ for every $\alpha, \beta \in \{0,1\}^n$ such that $\alpha \leq \beta$.

□

So, one can consider monotone Boolean circuits as a computing model for the class of all monotone Boolean functions. This means that one can be interested in some complexity measures of monotone Boolean functions defined according to this restricted circuit model.

Definition 3.6.2.9 *Let f be a monotone Boolean function. Then the* **monotone combinational complexity of f** *is*

$$\mathbf{MCC}(f) = \{\mathrm{CC}(B) \mid B \text{ is a monotone Boolean circuit computing } f\}.$$

The **monotone depth complexity of f** *is*

$$\mathbf{MD}(f) = \{\mathrm{D}(B) \mid B \text{ is a monotone Boolean circuit computing } f\}.$$

In what follows we define Boolean circuits over $\{\vee, \wedge, \Gamma\}$ with some restriction on the use of the negation Γ. Contrary to the case of monotone Boolean circuits we will show that this restriction does not have any essential influence on the computational power of the Boolean circuit model.

Definition 3.6.2.10 *A Boolean circuit over the basis $\{\vee, \wedge, \Gamma\}$ is in the* **quasimonotone form** *(or* **quasimonotone***) if the input of each gate containing Γ is a source element corresponding to an input variable (i.e., the outputs of negation gates are only negations of input variables).*

For convenience, we make the following agreement. For any variable x, we consider $\Gamma(x)$ as inputs. This means that the gate (Γ, x) is considered to have depth 0 in quasimonotone Boolean circuits.

Lemma 3.6.2.11 *For each Boolean circuit S over the basis $\{\vee, \wedge, \Gamma\}$, there exists an equivalent Boolean circuit S' in quasimonotone form such that*

(i) $\mathrm{D}(S') = \mathrm{D}(S)$, *and*

(ii) $\mathrm{CC}(S') \leq 2 \cdot \mathrm{CC}(S)$.

Proof. Let S be a Boolean circuit computing a function f over the set of input variables $\{x_1, \ldots, x_n\}$. The idea is to construct a circuit S' containing for each gate g of S two gates g^1 and g^2, where g^1 computes the same function as g ($\mathrm{Result}(g^1) = \mathrm{Result}(g)$) and g^2 computes the negation of g ($\mathrm{Result}(g^1) = \Gamma(\mathrm{Result}(g))$). We construct S' inductively according to the depth of the gates of S. If g is of depth 0, then g is an input variable x. In this case S' contains the gate (Γ, x).

For any h of S of depth smaller than $m \leq \mathrm{D}(S)$, let S' contain h^1 and h^2 with $\mathrm{Result}(h^1) = \mathrm{Result}(h)$ and $\mathrm{Result}(h^2) = \Gamma(\mathrm{Result}(h))$. Let g be of depth m in S. If $g = (\Gamma, h)$ for some h, then the depth of h in S is smaller than m. But this means that S' already contains h^1 and h^2 with $\mathrm{Result}(h^1) = \mathrm{Result}(h) = \Gamma(\mathrm{Result}(g))$ and $\mathrm{Result}(h^2) = \Gamma(\mathrm{Result}(h)) = \mathrm{Result}(g)$.

If $g = (\vee, h_1, h_2)$, then h_1 and h_2 have the depth smaller then m, and $\mathrm{Result}(g) = \mathrm{Result}(h_1) \vee \mathrm{Result}(h_2)$. Now, we add two gates $g^1 = (\vee, h_1^1, h_2^1)$ and $g^2 = (\wedge, h_1^2, h_2^2)$ to S'. Obviously, $\mathrm{Result}(g) = \mathrm{Result}(h_1^1) \vee \mathrm{Result}(h_2^1) = \mathrm{Result}(g^1)$, and $\Gamma(\mathrm{Result}(g)) = \Gamma(\mathrm{Result}(h_1^1) \vee \mathrm{Result}(h_2^1)) = \Gamma(\mathrm{Result}(h_1^1)) \wedge \Gamma(\mathrm{Result}(h_2^1)) = \mathrm{Result}(h_1^2) \wedge \mathrm{Result}(h_2^2) = \mathrm{Result}(g^2)$.

If $g = (\wedge, h_1, h_2)$, then we add the gates $g^1 = (\wedge, h_1^1, h_2^1)$ and $g^2 = (\vee, h_1^2, h_2^2)$ to S'. Again, applying the de Morgan rules one obtains $\mathrm{Result}(g^1) = \mathrm{Result}(g)$ and $\mathrm{Result}(g^2) = \Gamma(\mathrm{Result}(g))$. □

3.6.3 Communication Complexity of Relations

In this section we define the communication complexity of relations. Let X, Y, Z be some sets, and let $\mathrm{R} \subseteq X \times Y \times Z$ be a relation. Informally, a protocol computing R is the usual two-party protocol consisting of two computers C_I and C_{II}. Initially C_I has an input $x \in X$ and C_{II} has an input $y \in Y$. They communicate in order to compute an output $z \in Z$ such that $(x, y, z) \in \mathrm{R}$. We do not prescribe which z should be computed, but require only that $(x, y, z) \in \mathrm{R}$. In what follows we assume $Z \cap \{0,1\}^* = \emptyset$.

Definition 3.6.3.1 *Let X, Y, Z be some sets and let $\mathrm{R} \subseteq X \times Y \times Z$. A* **protocol computing the relation R** *is a function Φ with the following properties:*

(i) $\Phi : (X \cup Y) \times \bigcup_{i=0}^{\infty}(\{0,1\}^*\$)^i \to \{0,1\}^+ \cup Z$,

(ii) Φ *has the prefix-freeness property,*

(iii) *if* $\gamma = \Phi(\alpha, c_1\$c_2\$\ldots\$c_{2i}\$) \in Z$ *for some* $i \in \mathbb{N}$, *and* $c_1 = \Phi(\alpha, \lambda), c_2 = \Phi(\beta, c_1\$), \ldots, c_{2i} = \Phi(\alpha, c_1\$c_2\$\ldots\$c_{2i-1}\$) \in \{0,1\}^+$, $\alpha \in X$ $[\alpha \in Y]$, $\beta \in Y$ $[\beta \in X]$,
then $(\alpha, \beta, \gamma) \in \mathrm{R}$ $[(\beta, \alpha, \gamma) \in \mathrm{R}]$, *and* $\Phi(\beta, c_1\$c_2\$\ldots\$c_{2i}\$\$) = \gamma$,

(iv) *if* $\gamma = \Phi(\beta, c_1\$c_2\$\ldots\$c_{2i+1}\$) \in Z$ *for some* $i \in \mathbb{N}$, *and* $c_1 = \Phi(\alpha, \lambda), c_2 = \Phi(\beta, c_1\$), \ldots, c_{2i+1} = \Phi(\beta, c_1\$c_2\$\ldots\$c_{2i}\$) \in \{0,1\}^+$, $\alpha \in X$ $[\alpha \in Y]$, $\beta \in Y$ $[\beta \in X]$,
then $(\alpha, \beta, \gamma) \in \mathrm{R}$ $[(\beta, \alpha, \gamma) \in \mathrm{R}]$, *and* $\Phi(\alpha, c_1\$c_2\$\ldots\$c_{2i+1}\$\$) = \gamma$.

In case (iii) the word $D = c_1\$c_2\$\ldots\$c_{2i}\$\gamma$ *is called the* **computation of Φ on the inputs α and β**. *The* **communication complexity of the computation D** *is* $\mathbf{cc}(D) = |c_1c_2\ldots c_{2i}|$. $c_1c_2\ldots c_{2i}$ *is called the* **communication of Φ on the inputs α and β.**
In case (iv) the word $C = c_1\$c_2\$\ldots\$c_{2i+1}\$\gamma$ *is called the* **computation of Φ on the inputs α and β**. *The* **communication complexity of the computation C** *is* $\mathbf{cc}(C) = |c_1c_2\ldots c_{2i+1}|$. $c_1c_2\ldots c_{2i+1}$ *is called the* **communication of Φ on the inputs α and β.**
The **communication complexity of the protocol Φ** *is*

$$\mathbf{cc}(\Phi) = \max\{\mathrm{cc}(D) \mid D \text{ is a computation of } \Phi \text{ on some inputs } (\alpha, \beta) \in X \times Y \text{ such that there exists } \gamma \text{ with } (\alpha, \beta, \gamma) \in \mathrm{R}\}.$$

The **communication complexity of the relation R** *is*

$$\mathbf{cc}(\mathbf{R}) = \min\{\mathrm{cc}(\Phi) \mid \Phi \text{ is a protocol computing } \mathrm{R}\}.$$

We observe that the protocols computing relations do not prescribe which of the two computers starts the communication, unlike protocols computing functions, where C_I starts the communication for every input. On the other hand, one additionally requires for protocols computing relations that after the communication both computers C_I and C_{II} know the output.

In what follows we are interested in the two following relations depending on a given Boolean function f.

Definition 3.6.3.2 *Let* n *be a positive integer,* $d = \lceil \log_2 n \rceil$, *and let* $f \in B_2^n$. *We define* $\mathbf{R}(f)$ *as the maximal relation on* $\{0,1\}^n \times \{0,1\}^n \times \{1,2,\ldots,n\}$ *fulfilling the following conditions:*

(i) $\mathrm{R}(f) \subseteq N^1(f) \times N^0(f) \times \{1,2,\ldots,n\}$, *and*

(ii) *if* $(\alpha_1\alpha_2\ldots\alpha_n, \beta_1\beta_2\ldots\beta_n, i) \in \mathrm{R}(f)$, $\alpha_j, \beta_j \in \{0,1\}$ *for* $j = 1,\ldots,n$, *then* $\alpha_i \neq \beta_i$.

Let f *be a monotone Boolean function over a set of variables* X. *We define* $\mathbf{MR}(f)$ *as the maximal relation on* $2^X \times 2^X \times X$ *fulfilling the following conditions:*

(i) $\mathrm{MR}(f) \subseteq \mathrm{Min}(f) \times \mathrm{Max}(f) \times X$, *and*

(ii) if $(A, B, x) \in \mathrm{MR}(f)$, *then* $x \in A \cap B$.

Example 3.6.3.3 We consider the parity function $\mathrm{par}_n : \{0,1\}^n \to \{0,1\}$ defined by $\mathrm{par}_n(\alpha_1, \alpha_2, \ldots, \alpha_n) = \alpha_1 \oplus \alpha_2 \oplus \ldots \oplus \alpha_n$. We show that $\mathrm{cc}(\mathrm{R}(\mathrm{par}_n)) \leq 2 \log_2 n$ for $n = 2^k$. A protocol Φ computing $\mathrm{R}(\mathrm{par}_n)$ can be described as follows. Let $\alpha = \alpha_1 \ldots \alpha_n \in N^1(\mathrm{par}_n)$ and $\beta = \beta_1 \ldots \beta_n \in N^0(\mathrm{par}_n)$ be some inputs. Let $n > 1$. In the first two rounds C_I sends the bit $c_1 = \mathrm{par}_{n/2}(\alpha_1 \ldots \alpha_{n/2})$ to C_{II} and C_{II} sends the bit $c_2 = \mathrm{par}_{n/2}(\beta_1 \ldots \beta_{n/2})$ to C_I. If $c_1 \neq c_2$, then both C_I and C_{II} know there is an $i \in \{1, \ldots, n/2\}$ such that $\alpha_i \neq \beta_i$. If $c_1 = c_2$, then both C_I and C_{II} know there is an $j \in \{n/2+1, \ldots, n\}$ such that $\alpha_j \neq \beta_j$. Thus, using two bits Φ reduces the size of the problem by a factor of two. After $2 \log_2 n$ rounds of $2 \log_2 n$ bits together both C_I and C_{II} know an index l such that $\alpha_l \neq \beta_l$. □

3.6.4 Characterizations of Circuit Depth by the Communication Complexity of Relations

Proving a superlogarithmic lower bound on $\mathrm{D}(f)$ of a specific Boolean function is one of the most important open problems of the modern complexity theory. This is not only because $\mathrm{D}(f)$ corresponds to nonuniform parallel time complexity for computing f. A superlogarithmic lower bound on $\mathrm{D}(f)$ for some f would mean that there is no formula of polynomial size computing f. Note that the highest known lower bound on the size of formulas computing a specific Boolean function is $\Omega(n^{5/2})$.

In this section we show that $\mathrm{cc}(\mathrm{R}(f))$ can be used to estimate $\mathrm{D}_{\{\vee,\wedge,\neg\}}(f)$ for any Boolean function and that $\mathrm{cc}(\mathrm{MR}(f))$ can help to estimate the depth of monotone Boolean circuits for every monotone Boolean function f. To prove $\mathrm{D}_{\{\vee,\wedge,\neg\}}(f) \geq \mathrm{cc}(\mathrm{R}(f))$ we consider the following idea. Let S be a quasimonotone circuit computing f. Let g be the output gate of S with two inputs h_1 and h_2. Let $\alpha = \alpha_1 \ldots \alpha_n \in N^1(f)$, $\beta = \beta_1 \ldots \beta_n \in N^0(f)$. We show that using $\mathrm{D}(S)$ communication bits one can compute the path from g to some input x_i of S such that $\alpha_i \neq \beta_i$. Since $1 = \mathrm{Result}(g)(\alpha) = f(\alpha) \neq f(\beta) = \mathrm{Result}(g)(\beta) = 0$, it is obvious that $\mathrm{Result}(h_j)(\alpha) \neq \mathrm{Result}(h_j)(\beta)$ for at least one $j \in \{1,2\}$. Now, it is sufficient to observe that one communication bit suffices to estimate the gate h_j whose outputs differ on inputs α and β. The formalization of this idea is given in the next lemma.

Lemma 3.6.4.1 *Let f be a Boolean function. For every Boolean circuit S in the quasimonotone form computing f*

$$\mathrm{D}(S) \geq \mathrm{cc}(\mathrm{R}(f)).$$

Proof. Let f be a function from $\{0,1\}^n \to \{0,1\}$. We prove this lemma by induction on $D(S)$. If $D(S) = 0$ then $f(x_1, \ldots, x_n)$ is either x_i or $\Gamma(x_i)$ for some $i \in \{1, 2, \ldots, n\}$. In both cases i is always the correct output (i.e., $(\alpha, \beta, i) \in R(f)$ for all $\alpha \in N^1(f)$ and $\beta \in N^0(f)$). Thus, $cc(R(f)) = 0$.

We assume Lemma 3.6.4.1 is true for every circuit with a depth smaller than m, $m \in \mathbb{N}$. Let S have the depth m. We distinguish two cases according to the output gate g of S.

If $g = (\wedge, h_1, h_2)$, then, for all $\alpha \in N^1(f)$ and $\beta \in N^0(f)$, $1 = f(\alpha) = \text{Result}(g)(\alpha) = \text{Result}(h_1)(\alpha) \wedge \text{Result}(h_2)(\alpha)$, and $0 = f(\beta) = \text{Result}(g)(\beta) = \text{Result}(h_1)(\beta) \wedge \text{Result}(h_2)(\beta)$. Clearly, $\text{Result}(h_1)(\beta) = 0$ or $\text{Result}(h_2)(\beta) = 0$. If $\text{Result}(h_1)(\beta) = 0$, then C_{II} submits the bit 1 to C_I. If $\text{Result}(h_1)(\beta) = 1$ ($\text{Result}(h_2)(\beta) = 0$), then C_{II} submits the bit 0 to C_I. We observe that if the bit submitted was 0, then $1 = \text{Result}(h_1)(\alpha)$ (i.e., $\alpha \in N^1(\text{Result}(h_1))$) and $0 = \text{Result}(h_1)(\beta)$ (i.e., $\beta \in N^0(\text{Result}(h_1))$). Since h_1 is the gate of the depth smaller than m, the index i with $\alpha_i \neq \beta_i$ can be estimated in communication complexity smaller than m. Similarly, if the bit submitted was 0, then $\alpha \in N^1(\text{Result}(h_2))$ and $\beta \in N^0(\text{Result}(h_2))$. Since the depth of h_2 is smaller than m, the index i with $(\alpha, \beta, i) \in R(f)$ $[(\alpha, \beta, i) \in R(\text{Result}(h_2))]$ can be computed within communication complexity $m - 1$.

If $g = (\vee, h_1, h_2)$ the situation is similar. In this case the sender is C_I. C_I submits 1(0) if $\text{Result}(h_1)(\alpha) = 1$ ($\text{Result}(h_1)(\alpha) \neq 1$). So, the submission of the bit 1 means $\alpha \in N^1(\text{Result}(h_1))$ and $\beta \in N^0(\text{Result}(h_1))$. The submission of the bit 0 means $\alpha \in N^1(\text{Result}(h_2))$ and $\beta \in N^0(\text{Result}(h_2))$. Since both h_1 and h_2 have their depths smaller than m, the index i with $(\alpha, \beta, i) \in R(f)$ can be found by using at most $m - 1$ additional communication bits. □

Now, we prove a converse of Lemma 3.6.4.1.

Lemma 3.6.4.2 *Let $A_0, A_1 \subseteq \{0,1\}^n$, $A_0 \cap A_1 \neq \emptyset$, for an $n \in \mathbb{N}$. Then there exists a Boolean function $f \in B_2^n$ with $A_0 \subseteq N^0(f)$ and $A_1 \subseteq N^1(f)$ such that*

$$D_{\{\vee,\wedge,\Gamma\}}(f) \leq cc(R),$$

where $R \subseteq A_1 \times A_0 \times \{1, 2, \ldots, n\}$ and $(\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n, i) \in R$ if $\alpha_i \neq \beta_i$.

Proof. The proof is realized by induction on $cc(R)$. If $cc(R) = 0$, then there exists an $i \in \{1, \ldots, n\}$ such that for every $\alpha = (\alpha_1 \ldots \alpha_n) \in A_1$ and every $\beta = (\beta_1 \ldots \beta_n) \in A_0$, $\alpha_i \neq \beta_i$. Obviously, for all $(\alpha'_1 \ldots \alpha'_n)$, $(\alpha''_1 \ldots \alpha''_n) \in A_1$, $\alpha'_i = \alpha''_i$, and for all $(\beta'_1 \ldots \beta'_n)$, $(\beta''_1 \ldots \beta''_n) \in A_0$, $\beta'_i = \beta''_i$. If $\alpha'_i = \alpha''_i = 1$, then we take $f(x_1, \ldots, x_n) = x_i$. If $\alpha'_i = \alpha''_i = 0$ ($\beta'_i = \beta''_i = 1$), then we consider $f(x_1, \ldots, x_n) = \Gamma(x_i)$. In both cases $A_1 \subseteq N^1(f)$, $A_0 \subseteq N^0(f)$, and the Boolean circuit computing f has depth 0 (Remember that we agreed that the gates (Γ, x_i) have depth 0).

To prove the induction step, we distinguish two possibilities according to who starts the communication in the optimal protocol Φ computing R. First, we consider C_{II} sends the first bit c_1. Let $A_0 = A_{01} \cup A_{00}, A_{01} \cap A_{00} = \emptyset$,

and let C_{II} sends $c_1 = 0$ for inputs from A_{00}, and 1 for inputs from A_{01}. Now, we consider relations $\mathrm{R}_0 = (A_1 \times A_{00} \times \{1, 2, \ldots, n\}) \cap \mathrm{R}$ and $\mathrm{R}_1 = (A_1 \times A_{01} \times \{1, 2, \ldots, n\}) \cap \mathrm{R}$. We see that

$$\mathrm{cc}(\mathrm{R}) = 1 + \max\{\mathrm{cc}(\mathrm{R}_0), \mathrm{cc}(\mathrm{R}_1)\}$$

(if not, then Φ is not the optimal protocol for R). Since $\mathrm{cc}(\mathrm{R}_0) < \mathrm{cc}(\mathrm{R})$ and $\mathrm{cc}(\mathrm{R}_1) < \mathrm{cc}(\mathrm{R})$ we obtain by induction the existence of two functions h_0 and h_1 such that

(i) $A_1 \subseteq N^1(h_i)$ for $i = 0, 1$,

(ii) $A_{00} \subseteq N^0(h_0)$ and $A_{01} \subseteq N^0(h_1)$, and

(iii) $\mathrm{D}_{\{\vee,\wedge,\Gamma\}}(h_i) \leq \mathrm{cc}(\mathrm{R}_i)$ for $i = 0, 1$.

Taking now $f = h_0 \wedge h_1$ we obtain

$$A_1 \subseteq N^1(h_0) \cap N^1(h_1) = N^1(f),$$

$$A_0 \subseteq A_{00} \cup A_{01} \subseteq N^0(h_0) \cup N^0(h_1) = N^0(f),$$

and

$$\begin{aligned} \mathrm{D}_{\{\vee,\wedge,\Gamma\}}(f) &\leq 1 + \max\{\mathrm{D}_{\{\vee,\wedge,\Gamma\}}(h_0), \mathrm{D}_{\{\vee,\wedge,\Gamma\}}(h_1)\} \\ &\leq 1 + \max\{\mathrm{cc}(\mathrm{R}_0), \mathrm{cc}(\mathrm{R}_1)\} = \mathrm{cc}(\mathrm{R}). \end{aligned}$$

If C_I sends the first bit we consider a partition (A_{10}, A_{11}) of A_1 such that C_I sends 0 for inputs from A_{10} and C_I sends 1 for inputs from A_{11}. In this case we consider relations $\mathrm{R}'_i = (A_{1i} \times A_0 \times \{1, \ldots, n\}) \cap \mathrm{R}$ for $i = 0, 1$. Again, we have

$$\mathrm{cc}(\mathrm{R}) = 1 + \max\{\mathrm{cc}(\mathrm{R}'_0), \mathrm{cc}(\mathrm{R}'_1)\}.$$

By induction there exist f_0 and f_1 such that

(iv) $A_0 \subseteq N^0(f_i)$ for $i = 0, 1$,

(v) $A_{1i} \subseteq N^1(f_i)$ for $i = 0, 1$, and

(vi) $\mathrm{D}_{\{\vee,\wedge,\Gamma\}}(f_i) \leq \mathrm{cc}(\mathrm{R}'_i)$ for $i = 0, 1$.

Taking $f = f_0 \vee f_1$ we obtain

$$A_0 \subseteq N^0(f_0) \cap N^0(f_1) = N^0(f),$$

$$A_1 \subseteq A_{10} \cup A_{11} \subseteq N^1(h_1) \cup N^1(h_0) = N^1(f), \text{ and}$$

$$\begin{aligned} \mathrm{D}_{\{\vee,\wedge,\Gamma\}}(f) &\leq 1 + \max\{\mathrm{D}_{\{\vee,\wedge,\Gamma\}}(f_0), \mathrm{D}_{\{\vee,\wedge,\Gamma\}}(f_1)\} \\ &\leq 1 + \max\{\mathrm{cc}(\mathrm{R}'_0), \mathrm{cc}(\mathrm{R}'_1)\} = \mathrm{cc}(\mathrm{R}). \end{aligned}$$

□

Combining Lemma 3.6.4.1 and Lemma 3.6.4.2 we get the following result.

Theorem 3.6.4.3 *For every Boolean function f,*

$$D_{\{\vee,\wedge,\neg\}}(f) = cc(R(f)).$$

Thus, we have reduced the well-known open problem proving a superlogarithmic lower bound on the depth complexity of Boolean circuits to proving a lower bound on the communication complexity of relations. Unfortunately, nobody has been able to prove a superlogarithmic lower bound on $R(f)$ for any Boolean function f. The situation changes if one considers the communication complexity of $MR(f)$ for some monotone Boolean functions f. Here, one has proved several superlogarithmic lower bounds. The following claim shows that $MR(f)$ can provide a lower bound on $MD(f)$ and so we obtain a method for proving superlogarithmic (even linear) lower bounds on the depth complexity of monotone Boolean circuits.

Theorem 3.6.4.4 *For every monotone Boolean function f,*

$$MD(f) = cc(MR(f)) = cc(R(f)).$$

Proof. To prove $cc(R(f)) \leq MD(f)$ one can use the proof of Lemma 3.6.4.1 without any change because monotone circuits are a special case of quasimonotone circuits. On the other hand a protocol computing $R(f)$ for a monotone function f gives for every $\alpha = (\alpha_1 \dots \alpha_n) \in N^1(f)$ and every $\beta = (\beta_0 \dots \beta_n) \in N^0(f)$ an output i with the property $\alpha_i = 1$ and $\beta_i = 0$. Obviously, for every protocol with the above property the proof of Lemma 3.6.4.2 gives a monotone Boolean circuit. So, $MD(f) = cc(R(f))$.

It remains to show $cc(R(f)) = cc(MR(f))$ for every monotone Boolean function f. Let $\alpha = (\alpha_1, \dots, \alpha_n) \in N^1(f)$ be considered as the characteristic vector of a subset $Set(\alpha) \subseteq \{1, \dots, n\}$, and let $\beta = (\beta_1, \dots, \beta_n) \in N^0(f)$ be considered as the characteristic vector of the complement $Coset(\beta)$ of a subset $Set(\beta) \subseteq \{1, \dots, n\}$ (i.e., $\alpha_i = 1 \Leftrightarrow i \in Set(\alpha)$, $\beta_j = 0 \Leftrightarrow i \in Coset(\beta)$). If one interprets two inputs $\alpha \in N^1(f)$ and $\beta \in N^0(f)$ as $Set(\alpha)$ and $Coset(\beta)$, then every protocol computing $R(f)$ can be considered as protocol providing an element from $Set(\alpha) \cap Coset(\beta)$ for inputs $Set(\alpha)$ and $Coset(\beta)$. Since f is monotone the protocol for inputs $S_1, S_2 \subseteq \{1, 2, \dots, n\}$ can always behave as if it got inputs $S_1' \subseteq S_1$, and $S_2' \subseteq S_2$, where $S_1' \in Min(f)$ and $S_2' \in Max(f)$. Thus, $cc(R(f)) = cc(MR(f))$. □

To illustrate the applications of the theorems above we show how Theorem 3.6.4.3 can help to prove the $2\log_2 n$ lower bound on the depth $D(f)$ of specific symmetric functions. Obviously, this provides $\Omega(n^2)$ lower bound on the size of any formula over $\{\neg, \wedge, \vee\}$ computing these functions.

First, we give some helpful technical lemmas. We do so using the following notation. Let f be a Boolean function, and let Φ be a protocol computing $\mathrm{R}(f)$. For all inputs $\alpha \in N^1(f)$, $\beta \in N^0(f)$, $\mathbf{cc_I(\Phi, \alpha, \beta)}$ $[\mathbf{cc_{II}(\Phi, \alpha, \beta)}]$ denotes the sum of the lengths of all messages submitted from C_I to C_{II} [from C_{II} to C_I] in the computation of Φ on α and β. We set $\mathbf{cc(\Phi, \alpha, \beta)} = \mathrm{cc_I}(\Phi, \alpha, \beta) + \mathrm{cc_{II}}(\Phi, \alpha, \beta)$. For every $\alpha \in N^1(f)$, $\boldsymbol{N^0(f, \alpha)} = N^0(f) \cap \{\gamma \mid \gamma$ differs from α exactly in one position$\}$. Analoguously, for every $\beta \in N^0(f)$, $\boldsymbol{N^1(f, \beta)} = N^1(f) \cap \{\delta \mid \delta$ differs from β exactly in one position$\}$. We set $\mathbf{Neig(f)} = \{(\alpha, \beta) \mid \alpha \in N^1(f), \beta \in N^0(f),$ and α differs from β in exactly one position$\}$.

Lemma 3.6.4.5 *Let f be a Boolean function, and let Φ be a protocol computing $\mathrm{R}(f)$. For every $\alpha \in N^1(f)$ and every $\beta \in N^0(f)$,*

(i) $\sum_{\gamma \in N^0(f,\alpha)} \mathrm{c_{II}}(\Phi, \alpha, \gamma) \geq |N^0(f, \alpha)| \cdot \log_2 |N^0(f, \alpha)|$ *and*

(ii) $\sum_{\delta \in N^1(f,\beta)} \mathrm{c_I}(\Phi, \delta, \beta) \geq |N^1(f, \beta)| \cdot \log_2 |N^1(f, \beta)|$.

Proof. Because of the symmetry we prove (i) only. To obtain (i) it is sufficient to observe that, for any two different $\gamma_1, \gamma_2 \in N^0(f, \alpha)$, C_{II} has to send different communications in the computations of Φ on inputs (α, γ_1) and (α, γ_2). Let us assume the opposite. Since C_I reads α in computations of Φ on both (α, γ_1) and (α, γ_2) the submission of the same messages from C_{II} to C_I cause that the whole communications of Φ on (α, γ_1) and (α, γ_2) are the same. Since both C_I and C_{II} know the result after the communication we even see that the whole computations on (α, γ_1) and (α, γ_2) must be the same ($\Phi(\gamma_1, c) = \Phi(\alpha, c\$) = \Phi(\alpha, c'\$) = \Phi(\gamma_2, c')$ if $c = c' \in \{0, 1, \$\}^*$). This means that Φ gives the same output for inputs (α, γ_1) and (α, γ_2). But this is impossible because α differs from γ_i in exactly one element for $i = 1, 2$, and $\gamma_1 \neq \gamma_2$. □

Lemma 3.6.4.6 *For every Boolean function f*

$$\mathrm{D}_{\{\vee,\wedge,\neg\}}(f) \geq \log_2 \frac{|\mathrm{Neig}(f)|^2}{|N^1(f)| \cdot |N^0(f)|} .$$

Proof. To show this result we prove, for every protocol Φ computing $\mathrm{R}(f)$,

$$\mathbf{Sum(f, \Phi)} = \sum_{(\alpha,\beta) \in \mathrm{Neig}(f)} \mathrm{cc}(\Phi, \alpha, \beta) \geq |\mathrm{Neig}(f)| \cdot \log_2 \frac{|\mathrm{Neig}(f)|^2}{|N^1(f)| \cdot |N^0(f)|} .$$

Let Φ be a protocol computing $\mathrm{R}(f)$. Then

$$\begin{aligned} \sum_{(\alpha,\beta) \in \mathrm{Neig}(f)} \mathrm{cc}(\Phi, \alpha, \beta) &\geq \sum_{(\alpha,\beta) \in \mathrm{Neig}(f)} \mathrm{c_I}(\Phi, \alpha, \beta) + \mathrm{c_{II}}(\Phi, \alpha, \beta) \\ &\geq \sum_{\alpha \in N^1(f)} \sum_{\gamma \in N^0(f,\alpha)} \mathrm{c_{II}}(\Phi, \alpha, \gamma) \\ &\quad + \sum_{\beta \in N^0(f)} \sum_{\delta \in N^1(f,\beta)} \mathrm{c_I}(\Phi, \delta, \beta) . \end{aligned}$$

Applying Lemma 3.6.4.5 we obtain

$$\begin{aligned}
\mathrm{Sum}(f,\Phi) &\geq \sum_{\alpha\in N^1(f)} |N^0(f,\alpha)|\cdot\log_2|N^0(f,\alpha)| \\
&\quad + \sum_{\beta\in N^0(f)} |N^1(f,\beta)|\cdot\log_2|N^1(f,\beta)| \\
&\geq \sum_{\alpha\in N^1(f)} \frac{|\mathrm{Neig}(f)|}{|N^1(f)|}\cdot\log_2\left(\frac{|\mathrm{Neig}(f)|}{|N^1(f)|}\right) \\
&\quad + \sum_{\beta\in N^0(f)} \frac{|\mathrm{Neig}(f)|}{|N^0(f)|}\cdot\log_2\left(\frac{|\mathrm{Neig}(f)|}{|N^0(f)|}\right) \\
&\geq |\mathrm{Neig}(f)|\cdot\left[\log_2\frac{|\mathrm{Neig}(f)|}{|N^1(f)|}+\log_2\frac{|\mathrm{Neig}(f)|}{|N^0(f)|}\right] \\
&= |\mathrm{Neig}(f)|\cdot\log_2\frac{|\mathrm{Neig}(f)|^2}{|N^1(f)|\cdot|N^0(f)|}.
\end{aligned}$$

□

Now we apply Lemma 3.6.4.6 to get a lower bound on $\mathrm{D}_{\{\vee,\wedge,\Gamma\}}(\mathrm{par}_n)$ for the parity function from Example 3.6.3.3.

Theorem 3.6.4.7 *For every positive integer n,*

$$\mathrm{D}_{\{\vee,\wedge,\Gamma\}}(\mathrm{par}_n)\geq 2\cdot\log_2 n.$$

Proof. We observe that $N^1(\mathrm{par}_n)$ contains all inputs with an odd number of 1's and that $N^0(\mathrm{par}_n)$ contains all inputs with an even number of 1's. So, for every $\alpha,\beta\in\{0,1\}^n$ differing in exactly one element, $(\alpha,\beta)\in(N^1(f)\times N^0(f))\cup(N^0(f)\times N^1(f))$. Thus, $\mathrm{Neig}(\mathrm{par}_n)=n\cdot 2^{n-1}$.

Applying Lemma 3.6.4.6 we obtain

$$\mathrm{D}_{\{\vee,\wedge,\Gamma\}}(\mathrm{par}_n)\geq\log_2\left(\frac{|\mathrm{Neig}(\mathrm{par}_n)|^2}{|N^1(\mathrm{par}_n)|\cdot|N^0(\mathrm{par}_n)|}\right)\geq\log_2\left(\frac{n^2 2^{2n-2}}{2^{2n-2}}\right)=2\cdot\log_2 n\ .$$

□

3.6.5 Exercises

Exercise 3.6.5.1 *Conctruct a monotone circuit computing*

(i) the Boolean convolution,

(ii) Boolean matrix product, and

(iii) threshold functions.

Exercise 3.6.5.2 ** *Prove an exponential lower bound on the size of monotone Boolean circuits recognizing a specific language.*

Exercise 3.6.5.3 * *Give some estimation on the number of monotone Boolean functions of n variables.*

Exercise 3.6.5.4 *Give some lower and upper bounds on* $\mathrm{cc}(\mathrm{MR}(f))$ *for*

(i) the threshold functions,

(ii) the Boolean convolution.

Exercise 3.6.5.5 ** *Prove a superlogarithmic lower bound on* $\mathrm{MD}(h_n(L))$ *for a concrete language L.*

3.6.6 Research Problems

Problem 3.6.6.1 ** *Prove a superlogarithmic lower bound on* $\mathrm{D}(h_n(L))$ *for a language L.*

Problem 3.6.6.2 * *Prove a linear bound* $\mathrm{D}_{\{\vee,\wedge,\Gamma\}}(h_n(L)) \geq d \cdot n$ *for a constant* $d > 2$ *and a specific language L.*

3.7 Bibliographical Remarks

Boolean circuits are one of the oldest computing models considered in theoretical computer science. The complexity aspects of Boolean circuits have been studied already in 1949 by Shannon [Sh49]. Up till now there were published hundreds of papers dealing with this computing model from the complexity theory point of view. Because of this we do not try to give any survey on this topic. We mention only results and references directly connected to the topic of Chapter 3. For more information excelent monographs and surveys by Boppana and Sipser [BS90], Dunne [Du88], Nigmatulin [Ni83], Savage [Sa76], and Wegener [We87] may be consulted.

The first result on Boolean circuits was devoted to Shannon's function $\mathrm{ShCC}(n)$ of combinational complexity. The first estimations on $\mathrm{ShCC}(n)$ are due Shannon [Sh49]. Probably the main contribution on this topic is the method of Lupanov [Lu58]. An excellent and exhaustive survey on the estimations on Shannon's functions of basic Boolean complexity measures was given by Nigmatulin [Ni83].

The area complexity of Boolean circuits was already introduced and studied in 1967 by Kravcov [Kr67]. Albrecht [Al78] and Kravcov [Kr67] obtained Shannon's characterizations of the area complexity of almost all Boolean functions of

n variables. Kramer and van Leeuwen [KL83] proved that any Boolean function of n variables can be realized in area $O(2^n)$. Škalikova [Sk76] considered a circuit layout where all inputs lay on the border in the fixed order $x_1, x_2, \ldots, x_n$. For this model she proved:

- lower bound $\Omega(n2^n)$ and upper bound $O(n2^n)$ on the area complexity of Boolean circuits computing all 2^n elementary conjuction of n variables,
- lower bound $\Omega(n2^{2^n})$ and upper bound $O(n2^{2^n})$ on the area of Boolean circuits computing all 2^{2^n} Boolean functions of n variables,
- lower bound $\Omega(n^2)$ and upper bound $O(n^2)$ on the area of Boolean circuits computing the multiplication of two binary integers of length n, and
- lower bound $\Omega(n \log_2 n)$ and upper bound $O(n \log_2 n)$ on the area of Boolean circuits computing some specific symmetric Boolean functions.

The lower bound $\Omega(n^{3/2})$ on the area complexity of Boolean circuits computing a specific Boolean function in this layout model was obtained by Škalikova in [Sk82]. The highest lower bound $\Omega(n^2)$ on the general layout model considered in Section 3.3 was established by Ložkin et al. in [LRSH88]. The general lower bound method on the area complexity presented in Section 3.3.3 (Theorems 3.3.3.4 and 3.3.3.6) is based on the paper by Hromkovič, Ložkin, Rybko, Sapoženko and Škalikova [HLR92]. The comparison of the two area layout models in Section 3.3.4 is due to Hromkovič and Šuster [HrSu90].

The three-dimensional layout of Boolean circuits was considered by Škalikova [Sk76]. She proved in a constructive way that each circuit laid out in space D can be laid out in area $D^{3/2}$, and that there is no better area layout of Boolean circuits in the three-dimensional lattice. The general method for proving lower bounds on the three-dimensional layout of Boolean circuits (Section 3.3.5) is due to [HLR92]. This method provides the highest known lower bound $\Omega(n^{3/2})$ on the space complexity of a specific Boolean function (Corollary 3.3.5.8).

One of the most challenging problems in complexity theory is to prove a nonlinear lower bound on the combinational complexity of a specific Boolean function. The highest known lower bounds are only linear ones (for the base of all Boolean functions of two variables given in Blum [Bl84], Harper and Savage [HSa73], Harper, Hsieh, and Savage [HHS75], Kloss and Malyshev [KMa65], Paul [Pa77], Schnorr [Sc80], and Stockmeyer [St76], for some special complete bases given in Gorelik [Gor73], Redkin [Re81], Schnorr [Sc74], and Soprunenko [So65]) despite the fact mentioned above that almost all Boolean functions of n variables require $\Omega(2^n/n)$ combinational complexity [Sh49, Lu58]. Nonlinear lower bounds have been proved only for Boolean circuits with some additional restrictions. Harper and Savage [HSa73] proved a $\Omega(n \log_2 n)$ lower bound on the size of so-called "synchronized" Boolean circuits computing a specific Boolean function. They also proved a $\Omega(n^2)$ lower bound on the combinational complexity of planar Boolean circuits computing a collection of n Boolean functions of n variables. The first $\Omega(n^2)$ lower bound on the number of gates of the

planar Boolean circuits computing a specific Boolean function was obtained independently by Hromkovič [Hr91] and Turán [Tu89]. (Note that the fact that almost all Boolean functions of n variables require $2^{n-3} - n/4$ planar combinational complexity was proved by McColl [Mc85b].) In Section 3.4 we presented a generalization of the previous proof ideas showing how communication complexity can provide lower bounds on Boolean circuits with sublinear separators. The results presented are based on the ideas and proofs formulated by Gubáš, Hromkovič, and Waczulík in [Hr91, GHW92]. The existence of bounded-degree graphs without sublinear separators called magnifiers (see Theorem 3.4.4.4) was shown (even in a constructive way) by several authors (see, for instance Alon [Al86], Alon and Milman [AM85], and Gaber and Galil [GG81]). The results of Section 3.4.4 showing that magnifiers may have more computational power than Boolean circuits with sublinear separators are due to Turán [Tu89] and Gubáš, Hromkovič, and Waczulík [GHW92]. The quadratic lower bounds on the planar Boolean circuits are based on the Planar Separator theorem of Lipton and Tarjan [LT79, LT80]. In Section 3.4.5 (Theorem 3.4.5.8) we have presented a weaker version of this theorem in order to provide a detailed, structural proof suitable for teaching purposes.

The idea of using communication complexity to prove lower bounds on the number of gates of unbounded fan-in Boolean circuits is due to Hromkovič [Hr85]. In this paper communication complexity was used to prove lower bounds on the number of gates of unbounded fan-in Boolean circuits over a base consisting of associative and commutative functions. The general approach presented in Section 3.5 is based on the following fact. If a function f with high communication complexity is computed by elements each having small communication complexity, then many of these elements are required in order to compute f. Arguments along this line were used in varying levels of explicitness in Allender [All89], Goldmann, Håstad, and Razborov [GHR92], Hromkovič [Hr91], Håstad and Goldmann [HG91], Hofmeister, Honberg, and Köling [HHK91], Hajnal, Maas, Pudlák, Szegedy, and Turán [HMPST87], Nisan [Nis93], Smolensky [Sm90], and Siu and Bruck [SB91]. The lower bounds on unbounded fan-in Boolean circuits with sublinear vertex-separators presented in Section 3.5.3 are based on a combination of the ideas of Hromkovič [Hr91] and of Section 3.5.2.

Despite the fact that we are unable to prove nonlinear lower bounds on the combinational complexity of specific Boolean functions, 10 years ago a big success was achieved at least by proving lower bounds on the number of gates of monotone Boolean circuits. Andreev [An85] and Razborov [Ra85, Ra85a] independently developed two methods for proving exponential lower bounds for monotone combinational complexity. Their lower bounds were even improved by Alon and Boppana [AB87]. Though the mentioned results of Andreev and Razborov give superlogarithmic depth lower bounds for monotone circuits computing certain functions, the depth lower bound is always logarithmic in the size bound. So these techniques apply to size rather than to depth. The first method for proving superlogarithmic lower bounds on the depth of monotone circuits (independently of their sizes) was developed by Karchmer and Wigder-

son [KWi88]. The concept of the characterization of the depth complexity of Boolean circuits by the communication complexity of relations (Section 3.6) was taken from [KWi88]. The method for proving the highest known lower bound $2\log_2 n$ on the depth of Boolean circuits over base $\{\vee, \wedge, \neg\}$ (Lemma 3.6.4.6) is also from [KWi88]. Lemma 3.6.4.6 is in fact a version of the well-known theorem of Chrapchenko [Ch71, Ch71a, Ch71b] providing quadratic lower bounds on the length of Boolean formulas computing some specific symmetric Boolean function over the base $\{\vee, \wedge, \neg\}$. Thus communication complexity is even useful for proving one of the highest lower bounds on the size of formulas. Note that the quadratic lower bound is the highest known for formulas over some special complete base. For the base over all Boolean functions of two variables, the nonlinear lower bounds have been established by Hodes and Specker method [HoS68] (extended and exhaustively analysed by Pudlák [Pu84a]), by the method of Fischer, Meyer and Paterson [FMP82], and by the method of Nechiporuk [Ne66]. The highest lower bound $\Omega(n^2/\log_2 n)$ was established in Paul [Pa77] by applying Nechiporuk's method for indirect addressing.

But the main contribution of the characterization of circuit depth by the communication complexity of relations is in proving superlogarithmic lower bounds on the monotone depth of specific monotone Boolean functions. The first superlogarithmic $\Omega(\log_2 n)^2$ lower bound was presented by Karchmer and Wigderson [KWi88]. Futher lower bounds were achieved by Karchmer, Raz, and Wigderson [KRW91], and Raz and Wigderson [RW89, RW90]. The main contribution is the linear lower bound on the depth of monotone circuits [RW90] for the matching problem. This result was achieved by reducing the problem of proving a superlogarithmic lower bound on $\mathrm{cc}(\mathrm{MR}(f))$ to the problem of proving a superlogarithmic lower bound on the Monte Carlo complexity of set disjointness. The latter lower bound was proved by Kalyanasundaram and Schnitger [KS87]. The lower bound proof of [KS87] was later simplified by Razborov [Ra90]. This last result of Razborov and Wigderson [RW90] provides an exponential lower bound on the length of monotone formulas computing the matching problem. A survey about the communication complexity of relations was given by Wigderson in [Wi91].

4. VLSI Circuits and Interconnection Networks

4.1 Introduction

In this chapter we apply communication complexity to obtain methods for proving lower bounds on the complexity measures of VLSI circuits. A VLSI circuit is a more powerful computing model than the Boolean circuit model. While the gates (processors) of VLSI circuits are as simple as the gates of Boolean circuits, the communication structure (the graph describing the connections between processors) of VLSI circuits may contain cycles. This requires that a computation of a VLSI circuit be considered as an alternating sequence of synchronized steps $C_1, D_1, C_2, D_2 \ldots$, where C_i are communication steps and D_i are working steps. In every communication step each directed edge (p_1, p_2) (communication link) transfers a binary value as the output of the processor p_1 to the processor p_2. In every working step each node (processor) of the VLSI circuit computes outputs according to the values of its inputs. This contrasts to the Boolean circuits, where each processor and each directed edge were at most once active in the whole computation on an input. The main complexity measures of the VLSI circuit model are layout area A and time T (the number of working steps). Since many computing problems may be solved with small area A if one allows a large time T and with small T if one allows relatively large A, one prefers to consider tradeoffs of complexity measures like $A \cdot T$, $A \cdot T^2$ in order to objectify the measurement of the hardness of computing problems. In this chapter we show that communication complexity may be used to get lower bounds on A and AT^2. Similarly as for Boolean circuits, we can obtain more powerful lower bounds if we consider VLSI circuits with some restrictions on their topology. We shall show that considering some topological restrictions we can even obtain some lower bounds on the powerful interconnection network computations. The main difference between interconnection networks and VLSI circuits is that interconnection networks may consist of powerful processors with large local memories.

This chapter is organized as follows. Section 4.2 provides the definitions of basic VLSI circuit models as well as the definitions of VLSI complexity measures. In Section 4.3 the methods for proving lower bounds on VLSI complexity measures are presented. We show there that one-way communication complexity provides lower bounds on the number of processors of VLSI circuits (and so on the layout area too) and that communication complexity squared is a lower

bound on AT^2 complexity. Several further lower bounds on some restricted models of VLSI circuits are given too. In Section 4.4 a model of interconnection networks is introduced as a powerful generalization of the VLSI circuit model. We show there that the approach based on communication complexity is still powerful enough to provide some kinds of lower bounds for interconnection networks. In Section 4.5 we deal with another generalization of the basic VLSI circuit model. Here we allow a so-called "multiplicity" of input variables which means that we allow each input value to enter the circuit several times via different input processors (places). We show that by defining a slightly modified communication complexity measure we are still able to prove some nontrivial lower bounds on A and AT^2 complexity measures of multilective VLSI circuits. As usual, the last section is devoted to bibliographical remarks.

4.2 Definitions

4.2.1 Introduction

The aim of Section 4.2 is to give the basic definitions connected with VLSI circuit computations and with the related complexity measures. This section is organized as follows. Section 4.2.1 presents the basic VLSI circuit models. In Section 4.2.2 the fundamental complexity measures of VLSI computations are defined. Section 4.2.3 defines the basic models of probabilistic VLSI computations, and Section 4.2.4 contains exercises supporting the understanding of the definitions given in Section 4.2.

4.2.2 A VLSI circuit Model

Very-large-scale integrated (VLSI) circuits are built from conducting materials laid down on a wafer, usually of silicon. There are several technologies for producing VLSI chips. They differ in the kinds of materials used as well as in the way theses materials are collected and deposited. Fortunately, the differences between distinct technologies can be bounded by constant factors in area and time complexity measures. Thus, the lower bounds proved on the VLSI circuit model defined below work within a constant factor for all VLSI technologies.

Like the Boolean circuit, the VLSI circuit can be described as a directed graph whose nodes correspond to simple processors without any memory, and the edges correspond to wires (communication links) transporting Boolean values. But the directed graph representing a VLSI circuit need not be acyclic. Moreover, we mostly build VLSI circuits with many cycles because the processors of VLSI circuits work in every synchronized step of the computation of an input, in contrast to Boolean circuits, where each processor has worked only once during the whole computation on an input. The last main difference between Boolean circuits and VLSI circuits is that a pair (processor, step number) is assigned to each input variable of the problem computed. Two different pairs

must be assigned to two different variables. This means that one input processor may serve for several different input variables incoming in the VLSI circuit, but in different time units. The formal definition of a VLSI circuit follows.

Definition 4.2.2.1 *Let $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$ be sets of Boolean variables, $n, m \in \mathbb{N} - \{0\}$. A* **general program over the set of input variables X and the set of output variables Y** *is any set $S = \{g_1, g_2, \ldots, g_k\}$ in which each g is either*

(i) a nonempty set of pairs $\{(x_{i_1}, t_1), \ldots, (x_{i_r}, t_r)\}$ for some positive integer r, $t_j \in \mathbb{N}$, $t_1 < t_2 < \ldots < t_r$, and $i_j \in \{1, \ldots, n\}$ for $j = 1, \ldots, r$ or,

(ii) a tuple $(g_{j_1}, \ldots, g_{j_l}, \{(y_{i_1}, t_1), \ldots, (y_{i_s}, t_s)\}, f)$, where $1 \leq l \leq 3$, s is a positive integer, $t_j \in \mathbb{N}$, $t_1 < t_2 < \ldots < t_s$, $i_j \in \{1, \ldots, m\}$ for $j = 1, \ldots, s$, and f is a Boolean function from $\{0,1\}^l$ to $\{0,1\}$, or,

(iii) a tuple $(g_{i_1}, \ldots, g_{i_b}, f_1)$, where $1 \leq b \leq 3$, $i_j \in \{1, \ldots, k\}$ for $j = 1, \ldots, b$, and f_1 is a Boolean function from $\{0,1\}^b$ to $\{0,1\}$.

An element $g = \{(x_{i_1}, t_1), \ldots, (x_{i_r}, t_r)\} \in S$ $(1 \leq i \leq k)$ of type (i) is called an **input processor**. *t_b is called the* **input time unit of the variable x_{i_b} in the processor g** *for $b = 1, \ldots, r$. We also say that x_{i_b}* **enters S via the input processor g in the time unit t_b**. *An element $g = (g_{j_1}, \ldots, g_{j_l}, \{(y_{i_1}, t_1), \ldots, (y_{i_s}, t_s)\}, f) \in S$ of type (ii) is called an* **output processor**. *The number l is called the* **indegree of g** *and $g_{j_1}, \ldots, g_{j_l}$ are called* **the internal inputs of g**. *For $l = 1, \ldots, s$ we say that S* **computes y_{i_l} in the output processor g in time t_l**. *An element $g = (g_{i_1}, \ldots, g_{i_b}, f_1) \in S$ of type (iii) is called the* **(internal) processor**. *Further, we say that g* **realizes the function f_1** *and that an* **indegree of g** *is b. The elements $g_{j_1}, \ldots, g_{i_k}$ are called the* **internal inputs of g**. *The* **outdegree of an element $g \in S$** *is the number of occurrences of g as an internal input in all elements of S. The* **degree of g** *is the sum of the outdegree of g and the indegree of g. For every $x \in X$ we define the set*

$$\mathbf{In_S}(x) = \{(g,t) \mid g \text{ is an input processor of } S,\ t \in \mathbb{N},\ x \text{ enters } S \text{ via the processor } g \text{ in time } t\}.$$

For every $y \in Y$ we define the set

$$\mathbf{Out_S}(y) = \{(h,t) \mid h \text{ is an output processor of } S,\ t \in \mathbb{N},\ S \text{ computes } y \text{ in the output processor } h \text{ in time } t\}.$$

We say that a general program S is **consistent** *if $|\mathrm{Out_S}(y)| = 1$ for every $y \in Y$, and no output processor is an internal input of any element of S. A*

consistent general program S is called a **(semilective) VLSI program** *if each processor has its degree bounded by 4 and $|\mathrm{In}_S(x)| = 1$ for every $x \in X$.*

Any pair (g_i, g_j), $i, j \in \{1, \ldots, k\}$, is called a **wire of S** *if g_i is an internal input of g_j. For every input processor g we say that g has an* **input edge e_g**, *and for every output processor h we say that h has an* **output edge e_h**. *The output and input edges are also called the* **external edges of S**. *Let E_S denote the set of all wires and external edges of S.*

Now, to make precise how a VLSI program S computes we give the definition of a computation of S on an input string as a sequence of configurations, where a configuration unambiguously describes which Boolean values are communicated via wires and external edges of S in a time unit t.

Definition 4.2.2.2 *Let $S = g_1, g_2, \ldots, g_k$ be a VLSI program over the set of input variables $X = \{x_1, \ldots, x_n\}$ and the set of output variables $Y = \{y_1, \ldots, y_m\}$. Let $\alpha: X \to \{0,1\}$ be an input assignment. Let E_S be the set of wires and edges of S. For every $t \in \mathbb{N}$ we define the* **t-th configuration of S according to α, C_t^α** *as a function from E_S to $\{0,1\}$ inductively as follows:*

(i) For every external input edge e_g of an input processor $g = \{(x_{i_1}, t_1), \ldots, (x_{i_r}, t_r)\}$:

$$C_t^\alpha(e_g) = 0 \quad \text{if} \quad t \notin \{t_1, t_2, \ldots, t_r\},$$
$$C_t^\alpha(e_g) = \alpha_{i_j} \quad \text{if} \quad t = t_j \text{ for some } j \in \{1, \ldots, r\},$$

(ii) for every wire and every external output edge $h \in E_S$,

$$C_0^\alpha(h) = 0,$$

(iii) for every wire $(g_d, g_j) \in E_S$, where $g_d = (g_{i_1}, \ldots, g_{i_b}, f)$ is a non-input processor of S

$$C_t^\alpha((g_d, g_j)) = f(C_{t-1}^\alpha((g_{i_1}, g_d)), C_{t-1}^\alpha((g_{i_2}, g_d)), \ldots, C_{t-1}^\alpha((g_{i_b}, g_d))).$$

For every wire (g_d, g_j), where g_d is an input processor

$$C_t^\alpha((g_d, g_j)) = C_{t-1}^\alpha(e_{g_d}).$$

For every external output edge h_g of an output processor $g = (g_{j_1}, \ldots, g_{j_l}, \{(y_{i_1}, t_1), \ldots, (y_{i_s}, t_s)\}, f)$,

$$C_t^\alpha(h_g) = f(C_{t-1}^\alpha((g_{j_1}, g)), C_{t-1}^\alpha((g_{j_2}, g)), \ldots, C_{t-1}^\alpha((g_{j_l}, g))).$$

The infinite sequence $C = C_0^\alpha, C_1^\alpha, C_2^\alpha, \ldots$ is called **the computation of S on α**. *The transition from C_i^α to C_{i+1}^α is called a* **step of C**. *For every output processor $g = (g_{j_1}, \ldots, g_{j_l}, \{(y_{i_1}, t_1), \ldots, (y_{i_s}, t_s)\}, f)$ and for $\forall b \in \{1, \ldots, s\}$ we say that S* **computes the output value $C_{t_b}^\alpha(e_g)$ for y_{i_b} according to the**

input assignment α. *For every input assignment $\alpha \in \{0,1\}^n$, we say that* S **computes the output** $S(\alpha) = \beta_1, \beta_2, \ldots, \beta_m$ *if, for every $i = 1, \ldots, m$, S computes β_i for y_i according to α.*

We say that S **computes a computing problem** $\mathbf{P}_n^r = \{f_1, \ldots, f_r\}$, $f_i: \{0,1\}^n \to \{0,1\}$ *for $i = 1, \ldots, r$, if, for every $\alpha \in \{0,1\}^n$, $S(\alpha) = f_1(\alpha) f_2(\alpha) \ldots f_r(\alpha)$. If $r = 1$ we say that* S **computes the Boolean function** f_1.

Example 4.2.2.3 We illustrate the above definitions on an example of a specific VLSI program S computing two Boolean functions $f_1(x_1, \ldots, x_n, z_1, \ldots, z_n) = (x_1 \wedge z_1) \vee (x_2 \wedge z_2) \vee (x_3 \wedge z_3)$ and $f_2(x_1, \ldots, x_n, z_1, \ldots, z_n) = \bigvee_{i=1}^n (x_i \wedge z_i)$ for a positive integer n. We set $S = \{g_1, g_2, g_3, g_4\}$ over $X = \{x_1, \ldots, x_n, z_1, \ldots, z_n\}$ and $Y = \{y_1, y_2\}$, where

$$
\begin{aligned}
g_1 &= \{(x_1, 0), (x_2, 1), (x_3, 2), \ldots, (x_n, n-1)\} \text{ is an input processor,} \\
g_2 &= \{(z_1, 0), (z_2, 1), (z_3, 2), \ldots, (z_n, n-1)\} \text{ is an input processor,} \\
g_3 &= (g_1, g_2, \wedge), \text{ and} \\
g_4 &= (g_3, g_4, \{(y_1, 5), (y_2, n+2)\}, \vee).
\end{aligned}
$$

Figure 4.1 provides a visual description of S.

To illustrate the notion of a "computation" of a VLSI circuit we describe the first four configurations of S working on some input $\alpha\gamma = \alpha_1 \ldots \alpha_n \gamma_1 \ldots \gamma_n$.

$C_0^{\alpha\gamma}(e_{g_1}) = \alpha_1$, $C_0^{\alpha\gamma}(e_{g_2}) = \gamma_1$, $C_0^{\alpha\gamma}(g_1, g_3) = C_0^{\alpha\gamma}(g_2, g_3) = C_0^{\alpha\gamma}(g_3, g_4) = C_0^{\alpha\gamma}(g_4, g_4) = C_0^{\alpha\gamma}(e_{g_4}) = 0$.

$C_1^{\alpha\gamma}(e_{g_1}) = \alpha_2$, $C_1^{\alpha\gamma}(e_{g_2}) = \gamma_2$, $C_1^{\alpha\gamma}(g_1, g_3) = \alpha_1$, $C_1^{\alpha\gamma}(g_2, g_3) = \gamma_1$, $C_1^{\alpha\gamma}(g_3, g_4) = C_0^{\alpha\gamma}(g_1, g_3) \wedge C_0^{\alpha\gamma}(g_2, g_3) = 0 \wedge 0 = 0$, $C_1^{\alpha\gamma}(g_4, g_4) = 0$, $C_1^{\alpha\gamma}(e_{g_4}) = 0$.

$C_2^{\alpha\gamma}(e_{g_1}) = \alpha_3$, $C_2^{\alpha\gamma}(e_{g_2}) = \gamma_3$, $C_2^{\alpha\gamma}(g_1, g_3) = \alpha_2$, $C_2^{\alpha\gamma}(g_2, g_3) = \gamma_2$, $C_2^{\alpha\gamma}(g_3, g_4) = \alpha_1 \wedge \gamma_1$, $C_2^{\alpha\gamma}(g_4, g_4) = C_2^{\alpha\gamma}(e_{g_4}) = 0$.

$C_3^{\alpha\gamma}(e_{g_1}) = \alpha_4$, $C_3^{\alpha\gamma}(e_{g_2}) = \gamma_4$, $C_3^{\alpha\gamma}(g_1, g_3) = \alpha_3$, $C_3^{\alpha\gamma}(g_2, g_3) = \gamma_3$, $C_3^{\alpha\gamma}(g_3, g_4) = \alpha_2 \wedge \gamma_2$, $C_3^{\alpha\gamma}(g_4, g_4) = \alpha_1 \wedge \gamma_1 = C_3^{\alpha\gamma}(e_{g_4})$.

One can easily observe that, for every $i = 1, 2, \ldots, n$, the only external output edge e_{g_4} of S contains in time unit $i+2$ exactly the value $\bigvee_{j=1}^i (\alpha_j \wedge \gamma_j)$ for the input assignment $\alpha_1 \alpha_2 \ldots \alpha_n \gamma_1 \gamma_2 \cdots \gamma_n$. For every $i > n$ the output edge contains $\bigvee_{j=1}^n (\alpha_j \wedge \gamma_j)$. Thus, if one changes g_4 of S for $g_4^j = (g_3, g_4^j, \{(y, n+2+j)\}, \vee)$, then S computes the Boolean function $\bigvee_{i=1}^n (x_i \wedge z_i)$ for every positive integer j. □

Definition 4.2.2.4 *Let $S = \{g_1, g_2, \ldots, g_k\}$ be a VLSI program over the set of input variables X and the set of output variables Y. A* **VLSI circuit over X and Y corresponding to S** *is a directed graph $G_S = (\{g_1, g_2, \ldots, g_k\}, E_S)$.*

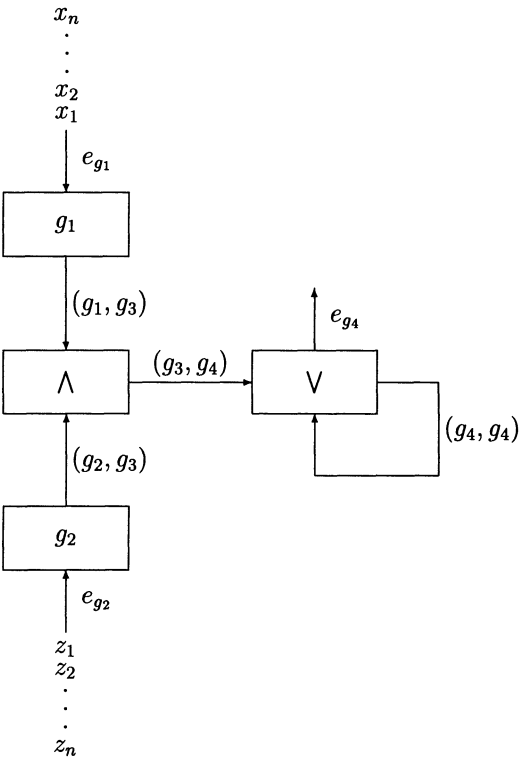

Fig. 4.1. A VLSI circuit with the input variables $x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_n$

Note that the representation of a VLSI program by the corresponding VLSI circuit G_S contains some repetition of information. According to Definition 4.2.2.4 each internal processor of G_S is labeled by some $g = (g_{i_1}, \ldots, g_{i_b}, f)$, but it is sufficient to consider the label f only because the edges $(g_{i_1}, g), \ldots, (g_{i_b}, g) \in E_S$ unambiguously determine which are the inputs of g. We observe that the VLSI circuit on Figure 4.1 is such a concise description of the VLSI program from Example 4.2.2.3.

Similarly as for Boolean circuits, we have to deal with the planar realization of VLSI circuits via the embedding into a two-dimensional grid. We observe that each VLSI circuit G_S is a directed graph of degree at most 4, and so G_S can be laid into the grid as described in Definition 3.3.2.1.

Definition 4.2.2.5 *Let S be a VLSI program, and let G_S be the corresponding VLSI circuit. Any grid-graph $\overline{G}$ of G_S is called a* **VLSI chip**. *We also say that* **the VLSI chip $\overline{G}$ is the layout of the VLSI circuit G,** *and that* **the VLSI**

chip $\overline{G}$ realizes the VLSI program S. *A VLSI chip $\overline{G}$ is a* **b-layout of G_S** *if the grid-graph $\overline{G}$ contains all input and output processors of G_S on the border of the smallest rectangle involving $\overline{G}$.*

To see some examples of grid-graphs one may review for Figures 3.4, 3.5, and 3.10 in Chapter 3.

4.2.3 Complexity Measures

The two main complexity measures investigated for the VLSI circuits are layout area and time. This contrasts to Boolean circuits, where the main complexity measure studied is the number of gates (combinational complexity). The reason for this difference is a more practical point of view on VLSI circuits and chips than on Boolean circuits, which represent one of the most important theoretical models of computing in complexity theory. The complexity theory of VLSI circuits has been developed so as to have strong relevance to real parallel computations on VLSI chips.

In this section we define the three following VLSI complexity measures:

- **parallel complexity** as the number of processors,
- **time complexity** as the minimal time after which all output values have appeared on the external output edges, and
- **area complexity** as the layout area of the circuit.

Definition 4.2.3.1 *Let $S = g_1, g_2, \ldots, g_k$ be a VLSI program over some X and Y. Let G_S be the VLSI circuit corresponding to S. We say that the* **parallel complexity of the VLSI program S, $\mathbf{P}(S)$,** *is the cardinality of S (i.e., $\mathrm{P}(S) = k$). We also say that the* **parallel complexity of the VLSI circuit G_S** *is $\mathbf{P}(G_S) = k$ = the number of vertices of G_S. For every computing problem F we define the* **parallel complexity of F** *as*

$$\mathbf{P}(F) = \min\{\mathrm{P}(S) \mid S \text{ is a VLSI program computing } F\}.$$

If $F = \{f\}$ for a Boolean function f we say that the **parallel complexity of the Boolean function f** *is $\mathbf{P}(f) = \mathrm{P}(F)$.*

We observe that the VLSI circuit in Figure 4.1 has parallel complexity of 4.

Definition 4.2.3.2 *Let $S = g_1, g_2, \ldots, g_k$ be a VLSI program over a set of input variables $X = \{x_1, \ldots, x_n\}$ and a set of output variables $Y = \{y_1, \ldots, y_m\}$, $k, n, m \in \mathbb{N} - \{0\}$. Let $\mathrm{Out}_S(y_i) = \{(g_{j_i}, t_i)\}$ for $i = 1, 2, \ldots, m$. The* **time complexity of S** *is defined as*

$$\mathbf{T}(S) = \max\{t_1, t_2, \ldots, t_m\}.$$

Let G_S be the VLSI circuit corresponding to S. We also say that the **time complexity of G_S** *is* $\mathbf{T(G_S)} = \mathrm{T}(S)$.

For every computing problem F we define the **time complexity of F** *as*

$$\mathbf{T(F)} = \min\{\mathrm{T}(S) \mid S \text{ is a VLSI program computing } F\}.$$

If $F = \{f\}$ for a Boolean function f we say that the **time complexity of the Boolean function f** *is* $\mathbf{T(f)} = \mathrm{T}(F)$.

We observe that the VLSI program from Example 4.2.2.3 has the time complexity $n + 2$. Thus, the time complexity of a VLSI program (circuit) S is the first time unit up to which the values of all output variables have appeared on the external output edges of S.

Now we define four area complexity measures. Outside the general one we consider three versions defined by the following two restrictions arising from some requirements of chip technologies:

(a) all input and output processors are on the border of the VLSI chip, and/or

(b) the lengths of the sides of the VLSI chip are approximately the same (i.e., the smallest rectangle containing the chip is a square).

Definition 4.2.3.3 *Let $S = \{g_1, g_2, \ldots, g_k\}$ be a VLSI program, and let G_S be the corresponding VLSI circuit. For any grid-graph $\overline{G}_S$ of G_S, the* **area of the VLSI chip $\overline{G}_S$, $\mathbf{A}(\overline{G}_S)$,** *is the area of the minimal rectangle* $\mathbf{Rect}(\overline{G}_S)$ *comprising all nonempty squares of the lattice. The* **balanced area of $\overline{G}_S$, $\mathbf{sA}(\overline{G}_S)$,** *is the area of the minimal squared rectangle comprising all nonempty squares of the lattice. The* **area of the VLSI program S (VLSI circuit G_S)** *is*

$$\mathbf{A(S)} = \mathbf{A(G_S)} = \min\{\mathrm{A}(\overline{G}) \mid \overline{G} \text{ realizes } S\}.$$

The **balanced area of S** *is*

$$\mathbf{sA(S)} = \min\{\mathrm{sA}(\overline{G}) \mid \overline{G} \text{ realizes } S\}.$$

For every computing problem F we define the **VLSI area complexity of F** *as*

$$\mathbf{A(F)} = \min\{\mathrm{A}(S) \mid S \text{ computes } F\}.$$

The **balanced VLSI area complexity of F** *is*

$$\mathbf{sA(F)} = \min\{\mathrm{sA}(S) \mid S \text{ computes } F\}.$$

If $F = \{f\}$ for a Boolean function f, we say that the **VLSI area complexity of f** *is* $\mathbf{A(f)} = \mathrm{A}(F)$, *and that the* **balanced VLSI area complexity of f** *is* $\mathbf{sA(f)} = \mathrm{sA}(F)$. *The* **b-area** *(border-area)* **of the VLSI program S (VLSI circuit G_S)** *is*

$$\mathbf{bA(S)} = \mathbf{bA(G_S)} = \min\{\mathrm{A}(\overline{G}) \mid \overline{G} \text{ is a b-layout of } G_S\}.$$

For every computing problem F we define the **VLSI b-area complexity of F** *as*

$$\mathbf{bA}(F) = \min\{\mathrm{bA}(S) \mid S \text{ computes } F\}.$$

If $F = \{f\}$ for a Boolean function f, we say that the **VLSI b-area complexity of f** *is* $\mathbf{bA}(f) = \mathrm{bA}(F)$. *The* **sb-area of the VLSI program** S *is*

$$\mathbf{sbA}(S) = \min\{\mathrm{sA}(\overline{G}) \mid \overline{G} \text{ is a b-layout of } G_S\}.$$

For every computing problem F the **VLSI sb-area complexity of F** *is*

$$\mathbf{sbA}(F) = \min\{\mathrm{sbA}(S) \mid S \text{ computes } F\}.$$

If $F = \{f\}$ for a Boolean function f, we say that the **VLSI sb-area complexity of f** *is* $\mathbf{sbA}(f) = \mathrm{sbA}(F)$.

Note that the VLSI circuit of Fig. 4.1 can fit in a 4×5 grid with area 20.

Observation 4.2.3.4 *For every Boolean function $f \in B_2^n$, $n \in \mathbb{N}$,*

$$\mathrm{A}(f) \leq 16 \cdot (\mathrm{P}(f))^2.$$

Proof. The proof is exactly the same as the proof of Observation 3.3.2.3 giving a layout of every graph of m nodes, e edges and degree 4 into an $(e+2) \times 4m$ lattice. □

A typical case of the study of the VLSI complexity measures of a specific problem F is that if one minimizes $\mathrm{A}(F)$, then $\mathrm{T}(F)$ is growing, and if one minimizes $\mathrm{T}(F)$, $\mathrm{A}(F)$ is growing. Obviously, from the practical point of view we are interested in minimizing both $\mathrm{T}(F)$ and $\mathrm{A}(F)$, which generally seems to be impossible. To give a clear framework, what and how much should be minimized, one considers tradeoffs of area complexity and time complexity. The two mainly considered complexity tradeoffs for VLSI circuits are area·time and area·time squared.

Definition 4.2.3.5 *Let $F = \{f_1, \ldots, f_m\}$ be a computing problem. The* **AT (area-time) complexity** *of F is*

$$\mathbf{AT}(F) = \min\{\mathrm{A}(S) \cdot \mathrm{T}(S) \mid S \text{ is a VLSI program computing } F\}.$$

If $F = \{f\}$ we say that the **AT complexity of the Boolean function f** *is* $\mathbf{AT}(f) = \mathrm{AT}(F)$.

The $\mathbf{AT^2}$ **(area·time squared) complexity of F** *is*

$$\mathbf{AT^2}(F) = \min\{\mathrm{A}(S) \cdot (\mathrm{T}(S))^2 \mid S \text{ is a VLSI program computing } F\}.$$

If $F = \{f\}$ we say that the $\mathbf{AT^2}$ **complexity of the Boolean function f** *is* $\mathbf{AT^2}(f) = \mathrm{AT}^2(F)$.

Observation 4.2.3.6 *Let* $\{f_n\}_{n=1}^{\infty}$, $f_n: \{0,1\}^n \to \{0,1\}$, *be a sequence of Boolean functions. Then*

$$\mathrm{AT}(f_n) = \Omega(\mathrm{CC}(f_n)).$$

Proof. We know that $\mathrm{CC}(f_n)$ is the number of Boolean operations (functions from $B_2^1 \cup B_2^2$) needed to compute f_n. For every VLSI program S, $\mathrm{A}(S)$ is the upper bound on the number of operations from $B_2^1 \cup B_2^2 \cup B_3^2 \cup B_4^2$ realized in one time unit by S. So, the number of operations executed in the first $\mathrm{T}(S)$ steps of the computation of S is bounded by $\mathrm{A}(S) \cdot \mathrm{T}(S)$. Since the operations of VLSI circuits are not much more complex than those of Boolean circuits, $\mathrm{A}(S) \cdot \mathrm{T}(S) \geq d \cdot \mathrm{CC}(f_n)$ for a suitable constant d independent on n. □

4.2.4 Probabilistic Models

In this section we define two basic models of probabilistic VLSI circuits:

- one-sided-error Monte Carlo VLSI circuits, and
- two-sided-error Monte Carlo VLSI circuits.

Informally, we do it by considering the input of a VLSI circuit S as an input consisting of the following two parts:

- the input of a Boolean function f computed by S, and
- a random input.

If S computes the right outputs for at least 2/3 of all random inputs, we speak about two-sided-error Monte Carlo VLSI circuits (recall two-sided-error Monte Carlo communication complexity from Section 2.5.5). If S computes the output $f(\alpha) = 0$ for every $\alpha \in \mathbb{N}^0(f)$ independently of the random input part, and if S computes $1 = f(\beta)$ for at least half of the random inputs for any $\beta \in \mathbb{N}^1(f)$, we speak about one-sided-error Monte Carlo probabilistic circuits.

Definition 4.2.4.1 *Let S be a VLSI program over a set X of input variables and a set $Y = \{y\}$ of ouput variables. Let $X = X^1 \cup X^2$, $X^1 \cap X^2 = \emptyset$, where $X^1 = \{x_1, \ldots, x_n\}$, and $X^2 = \{z_1, \ldots, z_r\}$ for some positive integers n and r. We say that S is a* **one-sided-error r-VLSI program computing a Boolean function f** $: \{0,1\}^n \to \{0,1\}$ *if the following conditions hold:*

(i) For every $\alpha \in \mathbb{N}^1(f)$, there exist at least 2^{r-1} input assignments $\beta: X^2 \to \{0,1\}$ such that $S(\alpha, \beta) = 1$ $(= f(\alpha))$.

(ii) For every $\gamma \in \mathbb{N}^0(f)$ and every input assignment $\delta: X^2 \to \{0,1\}$, $S(\gamma, \beta) = 0$ $(= f(\gamma))$.

In this case we also say that the **VLSI circuits G_S** *corresponding to S is a* **one-sided-error r-VLSI circuit computing f**. *The set X^2 is called the* **set of random variables of S (G_S)**.

Definition 4.2.4.2 *Let S be a VLSI program over a set X of input variables and a set $Y = \{y_1, \ldots, y_m\}$ of output variables. Let $X^1 \cup X^2$, $X^1 \cap X^2 = \emptyset$, where $X^1 = \{x_1, \ldots, x_n\}$, and $X^2 = \{z_1, \ldots, z_r\}$. We say that S is a* **two-sided-error r-VLSI program computing a problem $F = \{f_1, f_2, \ldots, f_m\}$** *with the set of input variables X^1 and the set of output variables Y if*

(iii) for every $\alpha: X^1 \to \{0,1\}$ there are at least $\lceil 2^{r+1}/3 \rceil$ input assignments $\beta: X^2 \to \{0,1\}$ such that

$$S(\alpha\beta) = F(\alpha) = f_1(\alpha) f_2(\alpha) \ldots f_m(\alpha).$$

We also say that the **VLSI circuit G_S** *corresponding to S* **is a two-sided-error r-VLSI circuit computing f**. *The set X^2 is called the* **set of random variables of S (G_S)**.

To see an example of probabilistic VLSI circuits consider the VLSI program S of Example 4.2.2.3 (Figure 4.1). Let us say that $z_1, z_2, \ldots, z_n$ are random variables and $x_1, x_2, \ldots, x_r$ are the "standard" input variables. Then, one can easily observe that S is a one-sided-error Monte Carlo r-VLSI circuit computing the Boolean function

$$f(x_1, x_2, \ldots, x_n) = x_1 \vee x_2 \vee \ldots \vee x_n.$$

4.2.5 Exercises

Exercise 4.2.5.1 *Consider the language $\tilde{L} = \{\alpha \in \{0,1\}^+ \mid \#_0(\alpha) = \#_1(\alpha)\}$. Show $\mathrm{A}(h_r(\tilde{L})) = O(\log_2 n)$.*

Exercise 4.2.5.2 *Let $R_0 = \{1w_1w_2 \ldots w_m \mid m \in \mathbb{N},\ w_i \in \{0\}^m \cup \{1\}^m$ for $i = 1, \ldots, m\}$. Construct a VLSI circuit computing $h_n(R_0)$ with area $O(\sqrt{n})$ and in time $O(\sqrt{n})$. Does there exist a VLSI circuit computing $h_n(R_0)$ in a constant area?*

Exercise 4.2.5.3 *Prove that, for every regular language L, $\mathrm{A}(h_n(L)) = O(1)$.*

Exercise 4.2.5.4 *Find a class $\mathcal{L}$ of languages properly containing the class of all regular languages and having the property $\mathrm{A}(h_n(L)) = O(1)$ for every $L \in \mathcal{L}$.*

Exercise 4.2.5.5 *Find a time optimal VLSI circuit computing the function $f_n(x_1, \ldots, x_n, z_1, \ldots, z_n) = \bigvee_{i=1}^{n}(x_i \wedge z_i)$ from Example 4.2.2.3. Prove that $\mathrm{AT}(f_n) = \Omega(n)$.*

Exercise 4.2.5.6 * *Define the Shannon function* $\mathrm{ShP}(n)$ *for the parallel complexity of VLSI circuits and prove some lower and upper bounds on* $\mathrm{ShP}(n)$.

Exercise 4.2.5.7 * *Prove that, for every linear Boolean function* $f_n: \{0,1\}^n \to \{0,1\}$, $\mathrm{P}(f_n) = O(\log_2 n)$.

Exercise 4.2.5.8 *Design an area optimal VLSI circuit computing the sum of two integers a and b, each one binary coded on the length n (i.e., the input is of the length* $2n$ *and the output of the length* $n+1$*).*

Exercise 4.2.5.9 * *Design a time optimal VLSI circuit computing the sum of two integers a and b, each one binary coded on the length n.*

4.3 Lower Bounds on VLSI Complexity Measures

4.3.1 Introduction

The aim of this section is to show how one can apply the communication complexity approach to get lower bounds on fundamental VLSI complexity measures. Section 4.3 is organized as follows. In Section 4.3.2 we show that the one-way communication complexity $\mathrm{acc}_1(f)$ of a Boolean function f provides a direct lower bound on the number of processors of VLSI circuits computing f. In Section 4.3.3 it is proved that the communication complexity $\mathrm{cc}(f)$ of a Boolean function f provides a lower bound on AT complexity of any circuit computing f, and that $(\mathrm{acc}(f))^2$ is a lower bound on the AT^2 complexity measure. Similarly as for Boolean circuits in Section 3.4, we give in Section 4.3.4 some special lower bounds on circuits with some topological restrictions.

4.3.2 Lower Bounds on Area Complexity

In this section we show that already the simplest communication complexity measure, one-way communication complexity, provides direct lower bounds on the parallel complexity of VLSI circuits, hence on the area complexity too. The idea of the use of one-way communication complexity differs from the previous approaches cutting a circuit into two parts and measuring the amount of information exchanged between these two parts. Here, we consider the VLSI circuit as a memory saving Boolean values on its wires. Thus, for a VLSI circuit G_S computing a problem F, one can say that $\{C_t^\alpha(e) \mid e \text{ is a wire of } G\}$ is the information (message) transferred from the time unit t to the time unit $t+1$ by the circuit G_S working on some input α. This implies that if one finds a time unit t such that the number n_t of input values entering G_S in the first t steps of G_S fulfills $\frac{1}{3}n \leq n_t \leq \frac{2}{3}n$ (n is the number of input variables), then the number of wires of G_S must be at least $\mathrm{acc}_1(F)$. Using this idea we obtain the following result.

Theorem 4.3.2.1 *For any computing problem F*

$$A(F) \geq P(F) \geq acc_1(F).$$

Proof. The fact $A(F) \geq P(F)$ is obvious and so it is sufficient to prove $P(F) \geq acc_1(F)$. Let $X = \{x_1, \ldots, x_n\}$ be the set of input variables of F, and let S (G_S) be a VLSI program (circuit) computing F. In what follows we distinguish two cases according to the existence of a time unit d in which G_S reads at least $\lceil n/3 \rceil$ variables.

(1) Let there be a time unit d such that the set $In(d) = \{x \in X \mid In_S(x) = \{(g, d)\}$ for some g from $S\}$ has the cardinality at least $\lceil n/3 \rceil$. Then G_S has at least $\lceil n/3 \rceil$ input processors. Thus $P(F) \geq \lceil n/3 \rceil$. Because $acc_1(F') \leq \lceil n/3 \rceil$ for every problem F' of n input variables we have $P(F) \geq acc_1(F)$.

(2) Let, for every $d \in \mathbb{N}$, $|In(d)| < \lceil n/3 \rceil$. Then there exists a time unit t such that

$$\lceil n/3 \rceil \leq \sum_{i=0}^{t} |In(i)| < 2 \cdot \lceil n/3 \rceil.$$

Since the wires outcoming from the same processor of G transfer the same value in any step of the computation, the information stored by the VLSI circuit G_S in the step from the t-th configuration into the $(t+1)$-st configuration can be coded as a binary message of length l equal to the number of internal processors of G_S. Since the time unit t defines an almost balanced partition Π of X, one can construct a one-way protocol (Π, Φ) computing F within communication complexity l. (For each input α the first computer submits the message coding the content of $C_t^\alpha(e)$ for all wires e.) Thus the number of internal processors of G_S must be at least $acc_1(F)$. □

Note that this lower bound approach can be easily extended to probabilistic circuits because of the following two reasons:

- part (1) of the proof of Theorem 4.3.2.1 is independent of whether S is deterministic or probabilistic, and
- if S is Monte Carlo probabilistic, then the one-way protocol constructed in (2) is also Monte Carlo probabilistic.

Thus we omit the formulation of this straightforward extension. We call also attention to the fact that Theorem 4.3.2.1 can be generalized to the following result.

Theorem 4.3.2.2 *For any computing problem F*

$$P(F) \geq sacc_1(F).$$

Since the proof of Theorem 4.3.2.2 is a straightforward extension of the proof of Theorem 4.3.2.1 we leave it as a simple exercise to the reader. We recall the fact that to prove lower bounds on the one-way communication complexity of specific problems is usually much easier that to do it for the general two-way communication complexity model. So, we have obtained a simply applicable technique for the estimation of area complexity of computing problems.

4.3.3 Lower Bounds on Tradeoffs of Area and Time

In this section we give lower bound methods on the complexity measures AT and AT^2 of VLSI circuits. We start with AT complexity. For AT complexity the method is based on the simple fact that a VLSI circuit has to read all "essential" input variables before computing all output values.

Theorem 4.3.3.1 *Let $f \in B_2^n$ be a Boolean function essentially depending on m input variables. Then*

$$\mathrm{AT}(f) \geq \mathrm{PT}(f) \geq m \geq \mathrm{cc}(f).$$

Proof. The inequalities $\mathrm{AT}(f) \geq \mathrm{PT}(f)$ and $m \geq \mathrm{cc}(f)$ are obvious. Now we prove $\mathrm{PT}(f) \geq m$. Let S be a VLSI program computing f. Since f essentially depends on m input variables, all these m input variables of f must be read by S before the time unit $\mathrm{T}(S)$. Since the number of input processors of S is bounded by $\mathrm{P}(S)$, S can read at most $\mathrm{P}(S)$ input values in one configuration (step). Thus, $\mathrm{T}(S) \cdot \mathrm{P}(S) \geq m$. □

Corollary 4.3.3.2 *Let $f \in B_2^n$ be a Boolean function essentially depending on all its input variables. Then*

$$\mathrm{AT}(f) \geq n.$$

Next we show that one can obtain a stronger lower bound if the squared layout area with input (output) processors on the border is required.

Theorem 4.3.3.3 *Let S be a VLSI program computing a Boolean function $f \in B_2^n$ depending on all its n input variables. Then*

$$\mathrm{sbA}(S) \cdot (\mathrm{T}(S))^2 \geq n^2/16.$$

Proof. Let the chip $\overline{G}_S$ be a squared b-layout of the VLSI circuit G_S corresponding to S. Then the area of $\mathrm{Rect}(\overline{G}_S)$ is $m \times m = m^2$ for some positive integer m. Since $\overline{G}_S$ is a squared b-layout of G_S the number of input variables read by S in one time unit is bounded by $4 \cdot m$. Thus $4 \cdot m \cdot \mathrm{T}(S) \geq n$, i.e., $16 \cdot m^2 \cdot (\mathrm{T}(S))^2 \geq n^2$. □

In what follows we realize the standard application of communication complexity by cutting the VLSI circuit into two parts and measuring how many bits can be exchanged between these two parts of the circuit in some time restricted computation. This is very similar to the lower bound technique for proving lower bounds on area complexity of Boolean circuits. The two main differences are that in the VLSI case we cannot remove time from consideration because each edge is active in each time unit and that we need almost balanced communication complexity instead of communication complexity because there are input processors reading several input variables.

Theorem 4.3.3.4 *For every computing problem F essentially depending on all its input variables*

$$\mathrm{AT}^2(F) \geq (\mathrm{sacc}(F))^2/4.$$

Proof. Let X be the set of input variables of F, and let $Z \subseteq X$ be such that $\mathrm{sacc}(F) = \mathrm{acc}(F, \langle Z \rangle)$. We have to prove that $\mathrm{AT}^2(F) \geq (\mathrm{acc}(F, \langle Z \rangle))^2$.

Let S be a VLSI program computing F, and let G_S be the corresponding VLSI circuit. We shall distinguish two cases depending on the fact whether S contains an input processor reading at least $\lceil |Z|/3 \rceil$ input variables from Z.

(1) Let S have an input processor $g = \{(x_{i_1}, t_1), \ldots, (x_{i_k}, t_k)\}$ such that $|Z \cap \{x_{i_1}, \ldots, x_{i_k}\}| \geq \lceil |Z|/3 \rceil$. Since $|\{t_1, \ldots, t_k\}| = k$ we have $\mathrm{T}(S) \geq \lceil |Z|/3 \rceil$. Thus $\mathrm{A}(S) \cdot (\mathrm{T}(S))^2 \geq (\mathrm{T}(S))^2 \geq |Z|^2/9$. Since $\mathrm{acc}(F, \langle Z \rangle) \leq \lceil |Z|/3 \rceil$ we obtain $\mathrm{A}(S) \cdot (\mathrm{T}(S))^2 \geq (\mathrm{sacc}(F))^2$.

(2) Let no input processor of S read more than $\lfloor |Z|/3 \rfloor$ input variables from Z. Let $\overline{G}_S$ be a grid-graph of G_S (a VLSI chip realizing S). Let $\mathrm{Rect}(\overline{G}_S)$ be of the size $a \times b$, $a \geq b$, for some positive integers a and b. One can simply observe that there is a cut-line ω of $\mathrm{Rect}(\overline{G}_S)$ (see Figure 3.7) such that

   (i) the length of ω is at most $b + 1$, and

   (ii) the cut $(E(\omega), V_1, V_2)$ of G_S induced by ω fulfills the property that the number of input variables from Z read by the input processors from V_1 (V_2) is at least $\lceil |Z|/3 \rceil$ and at most $2|Z|/3$.

   Thus ω defines an almost balanced partition Π of X according to Z. The line ω crosses at most $b+1$ wires of G_S. This means that at most $b+1$ bits are exchanged in one configuration of S between these two parts of G_S. If S computes in time $\mathrm{T}(S)$, then at most $(b+1) \cdot \mathrm{T}(S)$ bits are exchanged between V_1 and V_2 until all outputs are computed. Thus one can construct a protocol $D = \langle \Pi, \Phi \rangle$ computing F within communication complexity $\mathrm{cc}(D) = (b+1) \cdot \mathrm{T}(S)$. This implies $(b+1) \cdot \mathrm{T}(S) \geq \mathrm{acc}(F, \langle Z \rangle) = \mathrm{sacc}(F)$. Since $a \geq b$ we have $4 \cdot \mathrm{A}(S) \cdot (\mathrm{T}(S))^2 \geq (2a) \cdot (2b) \cdot (\mathrm{T}(S))^2 \geq (a+1) \cdot (b+1) \cdot (\mathrm{T}(S))^2 \geq (b+1)^2 \cdot (\mathrm{T}(S))^2 \geq (\mathrm{sacc}(F))^2$. □

Now we show that considering b-layout of VLSI circuits one can obtain still stronger lower bounds than Theorem 4.3.3.4 provides.

Theorem 4.3.3.5 *For every computing problem F essentially depending on all its n input variables*

$$\mathrm{bA}(F) \cdot (\mathrm{T}(F))^2 \geq n \cdot \mathrm{sacc}(F)/6.$$

Proof. Let X, Z, S, and G_S have the same meaning as in the proof of Theorem 4.3.3.5. We separately handle the same two cases as in that proof.

In the case (1) we have proved $\mathrm{T}(S) \geq \lceil |Z|/3 \rceil \geq \mathrm{sacc}(F)$. On the other hand Theorem 4.3.3.1 provides $\mathrm{A}(S) \cdot \mathrm{T}(S) \geq n$. So, $\mathrm{A}(S) \cdot (\mathrm{T}(S))^2 \geq n \cdot \mathrm{sacc}(F)$.

In the case (2) we have proved $(b+1) \cdot \mathrm{T}(S) \geq \mathrm{sacc}(F)$, where $a \geq b$ are the sizes of $\mathrm{Rect}(\overline{G}_S)$. Since $\overline{G}_S$ is a b-layout of G_S the number of input processors is at most $2a + 2b < 4a$. To read all inputs $4a \cdot \mathrm{T}(S) \geq n$ must hold. Thus we obtain

$$4a \cdot (b+1) \cdot (\mathrm{T}(S))^2 \geq n \cdot \mathrm{sacc}(F)$$

which implies

$$6 \cdot \mathrm{bA}(S) \cdot (\mathrm{T}(S))^2 \geq n \cdot \mathrm{sacc}(F).$$

□

In what follows we show that the lower bound of Theorem 4.3.3.4 holds for the AT^2 of probabilistic VLSI circuits according to probabilistic communication complexity too. As a consequence we obtain that the almost balanced nondeterministic communication complexity squared is a lower bound on $\mathrm{A}(S) \cdot (\mathrm{T}(S))^2$ of any one-sided-error Monte Carlo VLSI circuit computing f. We denote by r-1MCacc(f) (r-2MCacc(f)) the version of r-1MCcc(f) (r-2MCcc(f)) communication complexity according to the almost balanced partitions of the set of input variables of f.

Lemma 4.3.3.6 *Let $f \in B_2^n$ be a Boolean function essentially depending on all its n input variables. Then, for every one-sided-error [two-sided-error] Monte Carlo r-VLSI program S computing f*

$$\begin{aligned} \mathrm{A}(S) \cdot (\mathrm{T}(S))^2 &\geq (r\text{-1MCacc}(f))^2/4 \ [(r\text{-2MCacc}(f))^2/4], \text{ and} \\ \mathrm{bA}(S) \cdot (\mathrm{T}(S))^2 &\geq n \cdot r\text{-1MCacc}(f)/6 \ [n \cdot r\text{-2MCacc}(f)/6]. \end{aligned}$$

Proof. The proof is the same as the proof of Theorems 4.3.3.4 and 4.3.3.5. Instead of constructing a deterministic protocol communicating the bits exchanged between the two parts of the deterministic VLSI circuit one constructs a one-sided-error (two-sided-error) Monte Carlo r-protocol communicating the bits flowing across the line ω in the Monte Carlo r-VLSI circuit. Note that the protocols constructed have private random source. □

Theorem 4.3.3.7 *Let $f \in B_2^n$ be a Boolean function essentially depending on all its input variables. Then, for every one-sided-error Monte Carlo r-VLSI program S computing f,*

$$\begin{aligned} \mathrm{A}(S) \cdot (\mathrm{T}(S))^2 &\geq (\mathrm{ancc}(f))^2/4 \text{ and} \\ \mathrm{bA}(S) \cdot (\mathrm{T}(S))^2 &\geq n \cdot \mathrm{ancc}(f)/6. \end{aligned}$$

Proof. These lower bounds follow directly from Lemma 4.3.3.6 and from the fact that $\mathrm{ancc}(h) \leq r\text{-}1\mathrm{MCacc}(h)$ for any Boolean function h and any positive integer r. Note that this is really true because of the privacy of random sources of the protocols constructed in Lemma 4.3.3.6. □

Since we are able to investigate the nondeterministic communication complexity for many problems Theorem 4.3.3.7 provides a powerful method for proving lower bounds on one-sided-error Monte Carlo VLSI computations. The above stated lower bound methods help also to compare the computational powers of deterministic and probabilistic circuits as well as to study the influence of the layout restrictions on the increase of the computational resources (time, area) of specific computing problems. An example showing that one-sided-error VLSI circuits computing a specific problem can be more efficient than deterministic VLSI circuits is formulated in Exercise 4.3.5.9. We omit such examples in the text because they require a lot of technicalities in the construction of VLSI circuits, and contribute no novelty in the lower bound proofs representing the main topic of the book. But, in order to have an analogy to the results of Section 3.3.4 comparing the two layouts of Boolean circuits, we give the following lemma.

Lemma 4.3.3.8 *Let F be a computing problem computed by a VLSI circuit $\tilde{S}$ with m input processors. Then there exists a VLSI circuit S computing F such that*

(i) $\mathrm{bA}(S) \leq d \cdot \left(\mathrm{A}(\tilde{S}) + m\sqrt{\mathrm{A}(\tilde{S})}\right)$ *and*

(ii) $\mathrm{T}(S) \leq d \cdot \mathrm{T}(\tilde{S})$.

for a constant d independent on F and $\tilde{S}$.

Proof. Let $\tilde{S}$ be a VLSI circuit computing F, and let k be a positive integer. One can construct an equivalent VLSI circuit S' such that no processor outputs 1 in the k-th step, $\mathrm{T}(S') \leq c \cdot \mathrm{T}(\tilde{S})$ and $\mathrm{A}(S') \leq c \cdot \mathrm{A}(\tilde{S})$ for a constant c independent of S and F. This circuit S' has the property that if somebody sends all input values to S' with the same delay k, then S' computes the right outputs with the delay k. Due to this we can use the construction of Lemma 3.3.4.1 to obtain a b-layout of a VLSI circuit S equivalent to S' with $\mathrm{bA}(S) \leq 2 \cdot \mathrm{A}(S') + 12m \cdot \sqrt{\mathrm{A}(S')}$ and $\mathrm{T}(S) \leq \mathrm{T}(S') + 2$. □

Corollary 4.3.3.9 *There is a positive integer d such that for every Boolean function f*

$$\mathrm{bA}(f) \leq d \cdot \left(\mathrm{A}(f) + m\sqrt{\mathrm{A}(f)}\right) \leq d \cdot (\mathrm{A}(f))^{3/2}.$$

Corollary 4.3.3.10 *There is a positive integer d' such that for every Boolean function f*

$$\mathrm{bA}(f) \cdot (\mathrm{T}(f))^2 \leq d' \cdot \left(\mathrm{AT}^2(f)\right)^{3/2}.$$

To show that the construction of Lemma 4.3.3.8 is effective for both area and time complexity measures we consider the Boolean function

$$g_n(x_{1,1}, x_{1,2}, \ldots, x_{1,m}, x_{2,1}, x_{2,2}, \ldots, x_{2,m}, \ldots, x_{m,1}, x_{m,2}, \ldots, x_{m,m}) = \bigwedge_{i=1}^{m} (x_{i,1} \equiv x_{i,2} \equiv \ldots \equiv x_{i,m}) \bigvee \bigwedge_{j=1}^{m} (x_{1,j} \equiv x_{2,j} \equiv \ldots \equiv x_{m,j})$$

for any $n = m^2$, $m \in \mathbb{N}$, $m \geq 2$.

Theorem 4.3.3.11 *For every $n = m^2$, $m = 2k$, $k \in \mathbb{N} - \{0\}$:*

(i) $\mathrm{bA}(g_n) \cdot (\mathrm{T}(g_n))^2 = \Omega(n^{3/2})$,

and

(ii) $\mathrm{sA}(g_n) = O(n \cdot (\log_2 n)^2)$ *and* $\mathrm{AT}^2(g_n) = O(n \cdot (\log_2 n)^4)$.

Proof. In part *(ii)* of Lemma 3.3.4.2 we proved that $\mathrm{cc}(g_n) = \Omega(\sqrt{n})$. It is straightforward to obtain $\mathrm{acc}_1(g_n) = \Omega(\sqrt{n})$. Since g_n depends on all its n input variables applying Theorem 4.3.3.5 yields $\mathrm{bA}(f) \cdot (\mathrm{T}(f))^2 = \Omega(n^{3/2})$.

Now we describe a construction of a VLSI circuit S_n computing g_n with $\mathrm{A}(S_n) = O(n(\log n)^2)$ and $\mathrm{T}(S_n) = O(\log_2 n)$. S_n is very similar to the Boolean circuit for g_n in Figure 3.13 of Section 3.3.4. The placement of input processors of S_n is the same as in Figure 3.13. The only difference is that the equality of all elements in one row (column) is checked via a binary tree of processors. Similarly the $\sqrt{n}$ conjunctions of the values produced by the columns (rows) roots are again computed with a binary tree of processors (note that we have no cycle in S_n). Since a binary tree of $m = \sqrt{n}$ nodes can be laid in the area $O(\sqrt{n} \cdot \log_2 \sqrt{n}) = O(\sqrt{n} \cdot \log_2 n)$, S_n has a squared layout of size $d \cdot \sqrt{n} \cdot \log_2 n \times d \cdot \sqrt{n} \cdot \log_2 n$ for some suitable constant d. On the other hand $\mathrm{T}(S_n) = 2 \cdot \lceil \log_2(2 \cdot \sqrt{n}) \rceil + 1 \leq \log_2 n + 3$. □

4.3.4 VLSI circuits with Special Communication Structures

Here we show that communication complexity provides still stronger lower bounds than those presented in Sections 4.3.2 and 4.3.3 for VLSI circuits with

Fig. 4.2. One-dimensional systolic array of m processors

fixed, special communication structure. The presented lower bounds can help to prove the optimality of VLSI algorithms based on special communication structures like arrays of processors, complete binary trees, etc.

There are several practical reasons to use special communication structures as the basis for special VLSI models. If the structure is regular, then its hardware production as well as the verification of its correctness is much cheaper than the production of chips based on irregular communication structures. Moreover, if they are also modular (one can simply connect two or more of them in order to get the same communication structure of more processors) then interest in their use increases.

First, we consider the simplest parallel architecture – one-dimensional grids called also one-dimensional arrays. Because of their simplicity one-dimensional arrays have been widely used in many applications. Here we prove that already one-way communication complexity can provide lower bounds on their A and T complexities.

Definition 4.3.4.1 *A VLSI circuit $G = (V, E)$ of m non-input processors $p_1, p_2, \ldots, p_m$ is called a* **one-dimensional array** *(of m processors) if there exists a permutation $(i_1, i_2, \ldots, i_m)$ such that the set of all wires between two non-input processors of G is a subset of $\{(p_{i_k}, p_{i_{k+1}}), (p_{i_{k+1}}, p_{i_k}) \mid k = 1, 2, \ldots, m-1\}$.*

In what follows we always assume that a one-dimensional array of m processors has m non-input processors $p_1, p_2, \ldots, p_m$ and $\{(p_i, p_{i+1}), (p_{i+1}, p_i) \mid i = 1, 2, \ldots, m-1\}$ is the set covering all wires between $p_1, p_2, \ldots, p_m$. Thus, the inner communication structure of one-dimensional arrays is bounded by the bidirectional path depicted in Figure 4.2. Note that each of the processors $p_1, p_2, \ldots, p_m$ may be connected with at most two input processors.

An example of a one-dimensional array of m processors is the VLSI circuit (program) $G = (V, E)$ over the set of input variables $\{x_1, \ldots, x_m, z_1, \ldots, z_m\}$ and the set of output variables $\{y\}$, where

- $V = \{p_1, \ldots, p_m, g_1, \ldots, g_m, h_1, \ldots, h_m\}$,
- $g_i = \{(x_i, i-1)\}$ for $i = 1, \ldots, m$,
- $h_i = \{(z_i, i-1)\}$ for $i = 1, \ldots, m$,
- $p_1 = (g_1, h_1, \vee)$,
- $p_i = (g_i, h_i, p_{i-1}, f)$, where $f(u_1, u_2, u_3) = (u_1 \vee u_2) \wedge u_3$, for $i = 2, \ldots, m-1$,

- $p_m = (g_m, h_m, p_{m-1}, \{y, m+1\}, f)$.

We observe that G computes the Boolean function

$$g(x_1, \ldots, x_m, z_1, \ldots, z_m) = \bigwedge_{i=1}^{m} (x_i \vee z_i).$$

Theorem 4.3.4.2 *Let $F = \{f_1, \ldots, f_m\}$ be a computing problem, f_i: $\{0,1\}^n \to \{0,1\}$ for every $i = \{1, \ldots, m\}$. Let there exist a $j \in \{1, \ldots, m\}$ such that f_j essentially depends on all its input variables, and let every f_i essentially depend on at least one input variable. Then, for every one-dimensional array S computing F*

(i) $\mathrm{A}(S) \geq \mathrm{P}(S) \geq \mathrm{acc}_1(F)$ *and*

(ii) $\mathrm{T}(S) \geq \mathrm{acc}_1(F)/18$.

Proof. The property *(i)* is proved in Theorem 4.3.2.1 for every VLSI circuit. To prove *(ii)* it suffices to show $\mathrm{T}(S) \geq \mathrm{P}(S)/6$ for every one-dimensional array S computing F. Without loss of generality we may assume that S does not contain any "dummy" processors (a "dummy" processor is a processor at a distance larger than $\mathrm{T}(S)$ from any output processor, i.e., a processor never influencing any output value).

Let p be a processor of S producing the output value of $f_j \in F$. Since f_j depends on all n input variables of F, all input processors are at a distance at most $\mathrm{T}(S)$ from p. Since every $f_i \in F$ essentially depends on at least one input variable, all output processors are at a distance at most $2\mathrm{T}(S)$ from p (if not, all input processors are at a distance greater than $\mathrm{T}(S)$ from a given output processor, and so the outputs of this processor do not depend on any input variable). From the same reason we can conclude that all processors of S are at a distance at most $3\mathrm{T}(S)$ from p (each processor at a distance from p larger than $3\mathrm{T}(S)$ would be a dummy processor). Thus, we have $\mathrm{P}(S) \geq 6 \cdot \mathrm{T}(S)$ because there are at most $2 \cdot d$ processors at a distance at most d from p. □

Corollary 4.3.4.3 *For every one-dimensional array computing a problem F fulfilling the assumption of Theorem 4.3.4.2,*

$$\mathrm{A}(S) \cdot (\mathrm{T}(S))^2 = \Omega((\mathrm{acc}_1(F))^3)$$

The lower bounds of Theorem 4.3.4.2 can be used to show that a one-dimensional array is not a suitable parallel architecture for solving some computing problems with large one-way communication complexity and possibly small communication complexity. Another possible application of Theorem 4.3.4.2 is to give proofs of the optimality of some one-dimensional array algorithms for specific computing problems. The following example illustrates such an application.

Example 4.3.4.4 Let $F_{2n}^r = \{f_{2n}^1, f_{2n}^2, \ldots, f_{2n}^m\}$, $m = 2n-2$, where

$$f_{2n}^i(x_0, x_1, \ldots, x_{n-1}, z_0, z_1, \ldots, z_{n-1}) = \left(\sum_{j+k=i \ 0 \le j,k \le n-1} (x_j \wedge z_k) \right) \bmod 2$$

for $i = 0, 1, \ldots, 2n-2$. The computing problem F_{2n}^r is called the **convolution** and it is used, for instance, to compute the coefficients of the multiplication of two polynomials or to compute the greatest common divisor of two numbers. In the literature there are several one-dimensional arrays computing the convolution in linear time with a linear number of processors. Here we show that these parallel VLSI algorithms are area and time optimal in the class of one-dimensional array algorithms. By proving $\mathrm{acc}_1(F_{2n}^r) = \Omega(n)$ we even obtain that any one-dimensional array S_n computing F_{2n}^r with $\mathrm{A}(S_n) = O(n)$ and $\mathrm{T}(S_n) = O(n)$ is area optimal in the class of all VLSI algorithms and time optimal in the class of all one-dimensional array algorithms.

To prove $\mathrm{acc}_1(F_{2n}^r) \geq n/24$ for any positive integer n we observe that it is sufficient to prove that for each $\Pi \in \mathit{Abal}(X)$, $X = \{x_1, \ldots, x_n, z_1, \ldots, z_n\}$, there exists $k \in \{1, \ldots, r\}$ such that $\mathrm{acc}_1(f_{2n}^k, \Pi) \geq n/24$ (note that each communication protocol for F_{2n}^r is also a protocol for f_{2n}^k for any $k \in \{0, 1, \ldots, 2n-2\}$).

To prove for every $\Pi \in \mathit{Abal}(k)$ the existence of a $k \in \{0, 1, \ldots, 2n-2\}$ with high $\mathrm{acc}_1(f_{2n}^k, \Pi)$ we define so-called separated pairs. Let $\Pi = (\Pi_L, \Pi_R)$ be an arbitrary almost balanced partition of X. We denote $A = \{x_0, x_1, \ldots, x_{n-1}\}$ and $B = \{z_0, z_1, \ldots, z_{n-1}\}$. For any $i, j \in \{0, \ldots, n-1\}$ the pair (x_i, z_j) is called a **separated pair according to Π** if $(x_i, z_j) \in S_\Pi$, where

$$S_\Pi = (A \cap \Pi_L) \times (B \cap \Pi_R) \cup (A \cap \Pi_R) \times (B \cap \Pi_L).$$

We call S_Π the **set of separated pairs according to Π**. For any $k \in \{0, \ldots, n-2\}$ we say that a pair $(x_i, z_j) \in S_\Pi$ is a **separated pair of f_{2n}^k according to Π** if $i + j = k$ (i.e., if $x_i \wedge z_j$ is a term of the formula representation of f_{2n}^k). Let $S_\Pi^k = \{(x_r, z_s) \in S_\Pi \mid s + r = k\}$.

We show that, for every $k \in \{0, 1, \ldots, 2n-2\}$, $\mathrm{acc}_1(f_{2n}^k, \Pi) \geq |S_\Pi^k|$. Let $S_\Pi^k = \{(x_{i_1}, z_{j_1}), (x_{i_2}, z_{j_2}), \ldots, (x_{i_m}, z_{j_m})\}$. The one-way fooling set $\mathcal{A}(f_{2n}^k, \Pi)$ for f_{2n}^k and Π can be constructed as a set of input assignments $\alpha \in \{0,1\}^{\Pi_L} \to \{0,1\}$, where $\alpha(u) = 0$ for $u \in \Pi_L - \{x_{i_1}, \ldots, x_{i_m}, z_{j_1}, \ldots, z_{j_m}\}$ and $\alpha(v) \in \{0,1\}$ may be chosen arbitrarily if $v \in \Pi_L \cap \{x_{i_1}, \ldots, x_{i_m}, z_{j_1}, \ldots, z_{j_m}\}$. Obviously $|\mathcal{A}(f_{2n}^k, \Pi)| \geq 2^m = 2^{|S_\Pi^k|}$. If $\alpha, \beta \in \mathcal{A}(f_{2n}^k, \Pi)$, and $\alpha \neq \beta$, then we may assume there is a $w \in \{x_{i_s}, z_{j_s}\}$ for an $s \in \{1, \ldots, m\}$ such that $\alpha(w) \neq \beta(w)$. Without loss of generality we assume $w = x_{i_s}$. Then, for the following input assignment $\gamma\colon \Pi_R \to \{0,1\}$ with $\gamma(z_{j_s}) = 1$ and $\gamma(v') = 0$ for all $v' \in \Pi_R - \{z_{j_s}\}$, we see that $f_{2n}^k(\Pi^{-1}(\alpha, \gamma)) \neq f_{2n}^k(\Pi^{-1}(\beta, \gamma))$.

Thus it is sufficient to show that for every $\Pi \in \mathit{Abal}(X)$ there is a $k \in \{0, 1, \ldots, 2n-2\}$ such that $|S_\Pi^k| \geq n/24$. We observe that $S_\Pi = \bigcup_{i=0}^{2n-2} S_\Pi^i$ and that $S_\Pi^i \cap S_\Pi^j = \emptyset$ for any $i \neq j$, $i, j \in \{0, 1, \ldots, 2n-2\}$. Thus $|S_\Pi| = \sum_{i=0}^{2n-2} |S_\Pi^i|$. To prove the existence of an integer $k \in \{0, 1, \ldots, 2n-2\}$ with $|S_\Pi^k| \geq n/24$ it is sufficient to prove $|S_\Pi| \geq n^2/12$.

Remember $S_\Pi = (A \cap \Pi_L) \times (B \cap \Pi_R) \cup (A \cap \Pi_R) \times (B \cap \Pi_L)$. We see that there exists at least one pair of sets $(D, H) \in \{(A, \Pi_L), (A, \Pi_R), (B, \Pi_L), (B, \Pi_R)\}$ such that $|D \cap H| \geq n/2$ (if such a pair does not exist, then $|X| < 2n$, and we have the contradiction). Without loss of generality we suppose $|A \cap \Pi_L| \geq n/2$. Now we estimate the size of $B \cap \Pi_R$. From the facts

$$\begin{aligned} \Pi_R &= (A \cap \Pi_R) \cup (B \cap \Pi_R), \text{ and} \\ |\Pi_R| &= |A \cap \Pi_R| + |B \cap \Pi_R| \geq 2n/3 \end{aligned}$$

we get

$$|B \cap \Pi_R| \geq 2n/3 - |A \cap \Pi_R| \geq 2n/3 - n/2 = n/6.$$

It is evident that $(A \cap \Pi_L) \times (B \cap \Pi_R) \subseteq S_\Pi$ and so $|S_\Pi| \geq |A \cap \Pi_L| \cdot |B \cap \Pi_R| \geq (n/2) \cdot (n/6) \geq n^2/12$. □

The main argument of the proof of Theorem 4.3.4.2 is in fact based on a more general idea combining communication complexity with a large radius of the circuit in a lower bound argument. Using it we obtain the following assertion.

Theorem 4.3.4.5 *Let $F = \{f_1, \ldots, f_m\}$ be a computing problem for some positive integer m. Let there exist a $j \in \{1, \ldots, m\}$ such that f_j essentially depends on all its input variables, and let every f_i essentially depend on at least one input variable. Then, for every VLSI circuit S computing F and containing no dummy processor,*

(i) $\mathrm{P}(S) \geq \mathrm{acc}_1(F)$, *and*

(ii) $\mathrm{T}(S) \geq \mathrm{radius}(S)/3$.

Proof. The fact *(i)* is a special case of the assertion of Theorem 4.3.2.1. In the proof of Theorem 4.3.4.2 we have proved that S contains a processor p with a distance at most $3\mathrm{T}(S)$ to any other processor. Thus $\mathrm{radius}(S) \leq 3\mathrm{T}(S)$. □

One can find a lot of applications of Theorem 4.3.4.5 whose precise formulation is omitted here. For instance, an interesting consequence follows for VLSI circuits with the communication structure of two-dimensional grids (two-dimensional arrays). If a Boolean function $f \in B_2^n$ depending on all its input variables has linear one-way a-communication complexity, then $\mathrm{P}(f)$ is linear and $\mathrm{T}(S) = \Omega(\sqrt{n})$ for any two-dimensional array. This follows from the fact that any two-dimensional grid of $m^2 = \mathrm{P}(f)$ nodes has radius at least $m/2$.

Another approach for proving high lower bounds for VLSI circuits with special communication structures is based on graph separators. This helps for instance for balanced trees which have small radius and so Theorem 4.3.4.5 provides only trivial lower bounds on their time complexity. Since the approach based on graph separators provides even lower bounds for interconnection networks (which are a powerful generalization of VLSI circuits), we leave the presentation of this approach to the next section devoted to lower bounds for interconnection networks.

4.3.5 Exercises

Exercise 4.3.5.1 * *Let F be a sorting problem defined as follows. The input consists of m numbers, each number binary coded as a word of length $\lceil \log_2 m \rceil$ (i.e., the input length is $n = m \cdot \lceil \log_2 m \rceil$). The output has to contain these m numbers in a sorted order. Prove*

$$\mathrm{A}(F) = \Omega(m).$$

Exercise 4.3.5.2 *Prove that there is a context-free language L with $\mathrm{A}(h_n(L)) = \Omega(n)$.*

Exercise 4.3.5.3 *Give a formal proof of Theorem 4.3.2.2.*

Exercise 4.3.5.4 *Give an algorithm which, for every VLSI-chip $\overline{G}_S$ realizing a VLSI program S, constructs a chip $\overline{G}'_S$ whose minimal rectangle $\mathrm{Rect}(\overline{G}'_S)$ is a square.*

Exercise 4.3.5.5 *Give an algorithm which, for every VLSI-chip $\overline{G}_S$ realizing a VLSI program S, constructs a chip $\overline{G}'_S$ containing all input and output processors on the border of $\mathrm{Rect}(\overline{G}'_S)$.*

Exercise 4.3.5.6 *Extend the result of Theorem 4.3.3.1 for computing problems. (Do not forget to take the number of output variables into account too.)*

Exercise 4.3.5.7 *Give a formal description of a VLSI circuit S_n computing the Boolean function g_n from Theorem 4.3.3.11 with $\mathrm{AT}^2(S_n) = O(n(\log_2 n)^4)$.*

Exercise 4.3.5.8 *Define the three-dimensional layout of VLSI circuits in the same way as was done for the three-dimensional layout of Boolean circuits in Chapter 3. Give a version of Theorem 4.3.3.4 for the three-dimensional layout.*

Exercise 4.3.5.9 *Prove that the one-way almost balanced nondeterministic communication complexity of a Boolean function f provides a lower bound on the area of any one-sided-error Monte Carlo VLSI circuit computing f.*

Exercise 4.3.5.10 * *Consider the language L_{shift} from Section 2.5.6 as a candidate for showing that one-sided-error Monte Carlo VLSI circuits may be more powerful than deterministic VLSI circuits. Prove that*

(i) $\mathrm{AT}^2(h_n(L_{shift})) = \Omega(n^2)$, *and*

(ii) there exists an one-sided-error Monte Carlo $(\log_2 n)^2$-VLSI circuit S such that $\mathrm{T}(h_n(L_{shift})) = O((\log_2 n)^d)$ and $\mathrm{A}(h_n(L_{shift})) = O(n(\log_2 n)^k)$ for some positive integers k, d.

Exercise 4.3.5.11 *Extend Theorem 4.3.4.2 by exchanging* $\mathrm{acc}_1(F)$ *for* $\mathrm{sacc}_1(F)$.

Exercise 4.3.5.12 *Prove a version of Theorem 4.3.4.2 for computing problems with a weaker assumption according to the essential dependence on input variables.*

Exercise 4.3.5.13 *In Example 4.3.4.4 we have proved* $\mathrm{acc}_1(F^r_{2n}) \geq n/24$. *Improve this result by finding a constant* $d < 24$ *for which* $\mathrm{acc}_1(F^r_{2n}) \geq n/d$.

Exercise 4.3.5.14 *Let* f *be a Boolean function with* $\mathrm{acc}(f) = \Omega(n)$. *Prove that every VLSI circuit computing* f *with tree communication structure has* AT^2 *complexity in* $\Omega(n^3)$.

4.3.6 Problems

Problem 4.3.6.1 * *We use one-way communication complexity to obtain lower bounds on the area complexity of VLSI circuits despite the fact that this lower bound method has nothing to do with the layout of VLSI circuits. Find another method yielding good lower bounds on* $\mathrm{A}(F)$ *also in cases when* $\mathrm{A}(F)$ *essentially differs from* $\mathrm{P}(F)$.

Problem 4.3.6.2 *Find a language* L *such that* $\mathrm{A}(h_n(L))$ *is not in* $O(\mathrm{P}(h_n(L)))$.

Problem 4.3.6.3 *Either improve the result of Lemma 4.3.3.8 or prove the asymptotic optimality of the construction of Lemma 4.3.3.8 according to* AT^2 *complexity.*

4.4 Interconnection Networks

4.4.1 Introduction

Interconnection networks can be considered as a generalization of VLSI circuits. The main point is that instead of simple processors without memory the interconnection networks have processors that may have the power of a sequential computer. This means that we assume that each processor of a network has a memory of unbounded size and that it may compute very complex operations in one step. The aim of this section is to show that if the input data are distributed between the network processors and the architecture of the network has not-too-large separators (this is often the case for parallel architectures used in the practice), then we still can apply communication complexity to get nontrivial lower bounds on the time complexity and on the number of processors of interconnection networks.

This section is organized as follows. Section 4.4.2 contains a description of the interconnection network model considered here. The lower bounds are formulated in Section 4.4.3. As usual, the last sections contain exercises and research problems.

4.4.2 A Model of Interconnection Networks

In this section we briefly describe the model of interconnection networks considered here. We omit the exact complex definition and give instead an informal description which is sufficient for our purposes.

An **interconnection network**, shortly **network**, S can be viewed as an undirected graph $G = (V, E)$. Each node of G is a **register machine** (a sequential computer) which has a local memory containing a potentially infinite number of **memory registers**. Each register can contain a finite word over the alphabet $\{0, 1\}$. Each register machine R has a **read-only input tape** with random access, i.e., in any step R may decide to read the content of an arbitrary position of the input tape. The content of any position of the input tape is from $\{0, 1\}$. There is no restriction on the number of accesses to one position of the input tape during the computation of S. Each register machine of S contains also a **write-only output tape**, where one binary value may be written in one computation step.

A **computation of S** can be described as a sequence of steps, each **step** consisting of the computational part and of the communicational part. In one step each register machine R may:

1. read one position of the input tape (which position is read is decided (precomputed) by R, i.e., there is no prescribed input time unit for the input variables as with VLSI circuits),

2. execute a program (sequence of arithmetic and logic operations) over the input bit read and the data stored in the local memory,

3. write a word over the alphabet $\{0, 1\}$ on the output tape, and

4. send one bit via any edge adjacent to R to the neighbouring register machines.

Let F be a computing problem over the set of input variables X and over the set of output variables Y. Let S with the communication structure (V, E), $V = \{R_1, R_2, \ldots R_p\}$, be a network. If S has to compute F we assume that X is partitioned into p pairwise disjoint sets $X(R_1), \ldots, X(R_p)$ (some of the sets may be empty), and that each register machine R_i obtains exactly the values of the variables in $X(P_i)$ on the first $|X(P_i)|$ positions of its input tape for $i = 1, \ldots, p$. Similarly, we assume that Y is partitioned into p pairwise disjoint sets $Y(R_1), \ldots, Y(R_p)$ which means that R_i computes the output values of the variables in $Y(R_i)$ for every $i = 1, \ldots, p$. Informally, **S computes F** if, for every

given input α distributed between the register machines in the way described above, the outputs on the output tapes of S correspond to the result $F(\alpha)$.

We consider two complexity measures for networks. The **parallel complexity of a network, PC(S)**, is the number of register machines of S. The **time complexity of S, PT(S)**, is the minimal number of steps of S after which all output values are written on the output tapes. The **parallel complexity of a computing problem F** is

$$\mathbf{PC}(F) = \min\{\mathrm{PC}(S) \mid S \text{ computes } F\}.$$

The **parallel time complexity of a computing problem F** is

$$\mathbf{PT}(F) = \min\{\mathrm{PT}(S) \mid S \text{ computes } F\}.$$

We observe that networks are more powerful than VLSI circuits. One register machine can solve any computing problem alone. Thus, we cannot prove any lower bounds on the number of nodes of networks as we did for parallel complexity of VLSI circuits in Section 4.3.2. So we shall try to obtain some lower bounds on the tradeoffs of time complexity and the number of processors (register machines) in the following section.

Finally, we note that our model may seem to be out of proportion regarding the relation between the computational part and the communicational part of one step because we allow a lot of work with the local memory in one step but only the transmission of one bit via the links (edges) of the network. Since we are interested in asymptotic lower bounds it does not matter whether one allows the transmission of one bit via the links in one communication step or the transmission of a binary word of a fixed length. Further, we want to obtain lower bounds on the number of synchronized communication steps independently of the amount of local work done by the individual register machines. Obviously, such lower bounds also work if one restricts the number of operations done by the register machines in the computational part of one step.

4.4.3 Separators and Lower Bounds

The separators of the parallel architectures modeled by interconnection networks are usually smaller than those of the circuits. One-dimensional arrays (grids), trees, and tree-like communication structures were often used for parallel computers. All the structures mentioned above have polylogarithmic edge-separators. The following lower bounds show the weakness of such architectures for computing problems with high communication complexity.

Theorem 4.4.3.1 *Let $F = \{f_n\}_{n=1}^{\infty}$ be a computing problem, where f_n is a Boolean function essentially depending on all its n input variables. Let* $\mathrm{acc}(f_n) = \Omega(n)$, *and let $\{S_n\}_{n=1}^{\infty}$ be a sequence of networks, where S_n computes f_n for every positive integer n. Let S_n have a polylogarithmic edge-separator. Then:*

(i) If $\mathrm{PT}(S_n)$ *is a polylogarithmic function, then* $\mathrm{PC}(S_n) = 2^{\Omega(n^b)}$ *for some constant* $b > 0$.

(ii) If the degree of S_n *is bounded by some constant* k *for every* $n \in \mathbb{N}$, *then* $\mathrm{PT}(S_n) \in \Omega(n^d)$ *for some constant* $d > 0$ *(i.e.,* $\mathrm{PT}(S_n)$ *grows more quickly than any polylogarithmic function independently of the number of processors of* S_n*).*

Proof. For any $n \in \mathbb{N}$, let S_n compute f_n in time $z \cdot (\log_2 n)^m$ for some constants $z, m > 0$, and let each S_n have a strong $r \cdot (\log_2 \mathrm{PC}(S_n))^j$ edge-separator for some constants r, j. Let $\mathrm{acc}(f_n) \geq kn$ for a constant $k > 0$. First we show that there exists a positive constant d depending only on r and k such that

(1) $\mathrm{PT}(S_n) \cdot (\log_2 \mathrm{PC}(S_n))^{j+1} \geq d \cdot n$.

To prove (1) we shall distinguish two cases (a) and (b).

(a) Let there exist a processor R in S_n which has values of at least $\lceil n/3 \rceil$ input variables on its input tape. Since f_n essentially depends on all its input variables, R has to read all values of its input tape before all output values are computed by S_n. So, $\mathrm{PT}(S_n) \geq n/3$ which implies the validity of the equality (1) for $d = 1/3$.

(b) Let each processor of $S_n = (V_n, E_n)$ have at most $\lfloor n/3 \rfloor$ input values on its input tape. Using a partition technique similar to that of Lemma 3.4.3.2 we find a cut (E, V_n^1, V_n^2) of (V_n, E_n) such that at least $\lceil n/3 \rceil$ of input variables are assigned to processors in V_n^i for $i = 1, 2$. Moreover, $|E| \leq r \cdot (\log_2(\mathrm{PC}(S_n)))^{j+1}$ because we make at most $\log_2(\mathrm{PC}(S_n))$ partition steps and we remove at most $r \cdot (\log_2(\mathrm{PC}(S_n)/2^{i-1}))^j$ edges in the i-th step. This implies that at most $2 \cdot |E|$ bits of information can flow between the two parts of S_n specified by the cut (E, V_n^1, V_n^2) in one computing step. Since at least $\mathrm{acc}(f_n)$ bits must flow between these two parts during the whole computation on some input, we obtain

$$\mathrm{PT}(S_n) \geq \mathrm{acc}(\mathrm{P}_n)/2r \cdot (\log_2(\mathrm{PC}(S_n)))^{j+1}.$$

Thus (1) holds for $d = k/2r$. The inequality (1) directly implies *(i)*. More precisely, $\mathrm{PC}(S_n) = 2^{\Omega(n^b)}$ for any constant $b < 1/(j+1)$.

Now we prove the claim *(ii)* for bounded-degree networks. The idea of the proof of *(ii)* is based on the fact that no bounded-degree network can simultaneously use a large number of processors and compute in a short time.

Let the degree of all processors of S_n be bounded by a constant k. Let R' be the processor of S_n computing the output. Obviously, every processor having an influence on the output value computed by R' has to have the distance to R' at most $\mathrm{PT}(S_n)$. Since the degree of the network is bounded by a constant k there are at most $k^{\mathrm{PT}(S_n)}$ processors with the distance at most $\mathrm{PT}(S_n)$ to R'. Thus, we may assume $\mathrm{PC}(S_n) \leq k^{\mathrm{PT}(S_n)}$. This together with (1) implies

$$(\mathrm{PT}(S_n))^{j+2} = \Omega(\mathrm{acc}(\mathrm{P}_n).$$

Since $\mathrm{acc}(\mathrm{P}_n) = \Omega(n)$, we obtain $\mathrm{PT}(S_n) = \Omega(n^{1/(j+2)})$. □

It is an usual case that one tries to compute a function f very quickly, which means in polylogarithmic time. In fact, Theorem 4.4.3.1 shows that if f with a high communication complexity ($\mathrm{acc}(f) = \Omega(n^\epsilon)$ for some $\epsilon > 0$ is enough) has to be computed by a parallel architecture with a polylogarithmic separator, then one can do it only with an exponential number of processors, which is unrealistic. Additionally, if the degree of the network is bounded by a constant, it is impossible to compute f in polylogarithmic time. We are unable to prove such strong results for topologies with higher communication facilities. But we are still able to prove nonlinear lower bounds on the number of processors if one requires a polylogarithmic computation time.

Theorem 4.4.3.2 *Let $F = \{f_n\}_{n=1}^{\infty}$ be a sequence of Boolean functions, where $f_n: \{0,1\}^n \to \{0,1\}$ depends on all its input variables for $n = 1, 2, \ldots$. Let $\mathrm{acc}(f_n) = \Omega(n)$, and let $\{S_n\}_{n=1}^{\infty}$ be a sequence of networks, where S_n computes f_n in polylogarithmic time for every positive integer n. Let S_n have a strong $c \cdot n^b$ edge-separator for some positive constants $c, b < 1$. Then, for every $\epsilon > 0$,*

$$\mathrm{PC}(S_n) = \Omega(n^{1/b-\epsilon}).$$

Proof. Let $\mathrm{acc}(f_n) \geq dn$ for some constant $d > 0$. Let, for every positive integer n, S_n compute f_n in time $\mathrm{PT}(S_n) \leq k \cdot (\log_2 n)^m$ for some constants k and m. Let S_n have a strong $c \cdot n^b$ edge-separator for some constants $c > 0$, $0 < b < 1$. Since every f_n depends on all its input variables, and $\mathrm{PT}(S_n) \leq k \cdot (\log_2 n)^m$, no processor has assigned more than $\mathrm{PT}(S_n) \leq n/3$ input variables. Using the same argument as in the proof above, we can find a cut (E, V_n^1, V_n^2) of S_n such that it determines an almost balanced partition of the set of input variables and

$$|E| \leq \sum_{i=0}^{\log_2(\mathrm{PC}(S_n))} c \cdot (\mathrm{PC}(S_n)/2^i)^b.$$

Thus $|E| \leq a \cdot (\mathrm{PC}(S_n))^b$ for a suitable positive integer a. Since the number of bits flowing between V_n^1 and V_n^2 in one computation step of S_n is at most $2 \cdot |E| \leq 2a \cdot (\mathrm{PC}(S_n))^b$, we obtain

(2) $\mathrm{PT}(S_n) \cdot 2a(\mathrm{PC}(S_n))^b \geq \mathrm{acc}(f_n) \geq dn.$

Since $\mathrm{PT}(S_n) \leq k \cdot (\log_2 n)^m$, (2) implies

$$\mathrm{PC}(S_n) = \Omega(n^{1/b-\epsilon})$$

for every constant $\epsilon > 0$. □

Finally we present a result giving a general tradeoff between parallel time, parallel complexity, and separators of bounded-degree networks. The proof is based on the ideas already formulated in Theorems 4.4.3.1 and 4.4.3.2.

Theorem 4.4.3.3 *Let d be a positive integer, and let $h: \mathbb{N} \to \mathbb{N}$ be a monotone function. Let $F = \{f_n\}_{n=1}^{\infty}$ be a sequence of Boolean functions, and let $f_n: \{0,1\}^n \to \{0,1\}$ depend on all its input variables for any $n \in \mathbb{N}$. Let $\{S_n\}_{n=1}^{\infty}$ be a sequence of d-degree bounded networks having a strong h edge-separator. Then*

$$\mathrm{PT}(S_n) \geq \left(\frac{\mathrm{acc}(f_n)}{2(\log_2 d) \cdot h(\mathrm{PC}(S_n))} \right)^{1/2}.$$

Proof. As in the proof of previous theorems, we may assume that each processor of S_n has assigned at most $\lfloor n/3 \rfloor$ input variables. Then, removing at most

$$\sum_{j=0}^{\log_2(\mathrm{PC}(S_n))} h([\mathrm{PC}(S_n)/2^j]) \leq h(\mathrm{PC}(S_n)) \cdot \log_2(\mathrm{PC}(S_n))$$

edges, one can obtain a cut of S_n separating the input variables in an almost balanced way. Thus, as in the proofs of Theorems 4.4.3.1 and 4.4.3.2, we obtain

(1) $\mathrm{PT}(S_n) \geq \mathrm{acc}(f_n)/(2 \cdot h(\mathrm{PC}(S_n)) \cdot \log_2(\mathrm{PC}(S_n)))$.

Obviously, we may assume that S_n does not contain any "dummy" processor that never influences any output of S_n. This assumption implies that all processors of S_n have to be at a distance at most $\mathrm{PT}(S_n)$ from the processor producing the output. Thus, we have

(2) $d^{\mathrm{PT}(S_n)} \geq \mathrm{PC}(S_n)$

because the degree of S_n is bounded by d. (1) and (2) give together

$$\mathrm{PT}(S_n) \geq \frac{\mathrm{acc}(f_n)}{2 \cdot h(\mathrm{PC}(S_n)) \cdot \log_2(d^{\mathrm{PT}(S_n)})}$$

which directly implies the result of Theorem 4.4.3.3. □

We note that despite the fact that the lower bounds proved in Section 4.4.3 seem to be much weaker than the lower bounds obtained for A and AT^2 complexity measures of Boolean circuits in Section 4.3, the results of the above theorems have also interesting applications. For instance, in Section 2.3.5 we have proved that there is a context-free language having a linear communication complexity. Thus Theorem 4.4.3.1 implies that one cannot recognize context-free languages in polylogarithmic time by tree-like architectures (by degree-bounded networks with polylogarithmic separators). Theorem 4.4.3.2 shows that even for powerful networks with relatively large n^a, $0 < a < 1$, edge-separators (multidimensional grids, for instance) one has to pay for the polylogarithmic time of

the recognition of context-free languages by a nonlinear number of processors. The general consequence of the results of this section is that the architecture with small bisections (separators) are not suitable for problems with large communication complexity if one wants to compute very quickly.

4.4.4 Exercises

Exercise 4.4.4.1 *Extend the lower bound results presented in Section 4.4.3 to s-communication complexity.*

Exercise 4.4.4.2 *Prove other tradeoff between parallel time, parallel complexity, and separators of networks.*

Exercise 4.4.4.3 *Prove versions of Theorems 4.4.3.1, 4.4.3.2, and 4.4.3.3 without the assumption that f_n essentially depends on all its input variables.*

4.4.5 Problems

Problem 4.4.5.1 *Investigate simulations between distinct interconnection networks of the same size and use communication arguments to prove some lower bounds on the increase of computational resources of networks needed for such simulations.*

4.5 Multilective VLSI circuits

4.5.1 Introduction and Definitions

The VLSI circuit model introduced in Section 4.1 is **semilective**, which means that each input variable enters the circuit exactly once via one input processor. The model is also **where-** and **when-determinate (oblivious)** which means that the pair (input processor, time unit) is fixed for each variable independently of the input assignments. A computing model is called **where (when)-indeterminate** if the entry (the time unit) via (in) which the value of an input variable is read depends on the values of input variables read before.

A circuit model is **multilective** if one allows any input variable to enter the circuit several times possibly via different input processors in different time units. In this section we want to show that multilectivity can bring an additional computational power to VLSI circuits, and to show that we can use the communication protocol model in a special way to get lower bounds on multilective VLSI circuits.

This section is organized as follows. Section 4.5.2 shows that the strong unclosure properties of communication complexity can be used to show large

differences between the computational power of (semilective) VLSI circuits and multilective VLSI circuits. In Section 4.5.3 we present an approach to prove lower bounds on A and AT^2 of multilective VLSI circuits by using slighty modified communication protocols. As usual, we conclude with exercises and research problems.

First, we give the formal definition of multilective VLSI circuits.

Definition 4.5.1.1 *Let k be a positive integer. We say that a general consistent program S over a set X of input variables and a set Y of output variables is a* ***k*-multilective VLSI program over X and Y** *if, for every $x \in X$, $1 \leq |\mathrm{In}_S(x)| \leq k$. A k-multilective VLSI program is called a* **multilective VLSI program** *for any $k \geq 2$.*

4.5.2 Multilectivity Versus Semilectivity

In Section 2.3.4 we showed that there exist two languages L_1 and L_2 such that $\mathrm{cc}(h_n(L_1)) = O(1)$, $\mathrm{cc}(h_n(L_2)) = O(1)$, and $\mathrm{cc}(h_n(L_1 \cup L_2)) = \Omega(n)$. The idea behind this was that $\mathrm{cc}(h_1(L_1), \Pi)$ was small exactly for those partitions Π for which $\mathrm{cc}(h_1(L_2), \Pi)$ was large, and $\mathrm{cc}(h_1(L_2), \Pi')$ was small exactly for those partitions Π' for which $\mathrm{cc}(h_1(L_1), \Pi')$ was large. To show similar results for some VLSI complexity measures one can extend this idea by considering some Boolean functions which are easy if one chooses a suitable placement of input processors and/or some special order in which their input variables enter the VLSI circuit, but which are hard if this special input processors placement and this special order of reading the input values are not allowed. Thus, if two Boolean functions f_1 and f_2 have very different requirements on the way in which their inputs are read, then $\{f_1, f_2\}$ (or $\{f_1 \vee f_2\}$) is probably hard. The following lemma shows such a specific example for the area complexity of VLSI circuits. For any $n = m^2$, $m \in \mathbb{N}$, let

$$g_n^1(x_{1,1}, \ldots, x_{1,m}, x_{2,1}, \ldots, x_{2,m}, \ldots, x_{m,1}, \ldots, x_{m,m}) = \bigwedge_{i=1}^{m} (x_{i,1} \equiv x_{i,2} \ldots \equiv x_{i,m})$$

and

$$g_n^2(x_{1,1}, \ldots, x_{1,m}, x_{2,1}, \ldots, x_{2,m}, \ldots, x_{m,1}, \ldots, x_{m,m}) = \bigwedge_{j=1}^{m} (x_{1,j} \equiv x_{2,j} \ldots \equiv x_{m,j}).$$

Lemma 4.5.2.1 *For every positive integer $n = m^2$, $m \in \mathbb{N}$, $m \geq 2$, the following three claims hold:*

(i) $\mathrm{A}(g_n^1) = \mathrm{A}(g_n^2) = O(1)$,

(ii) $\mathrm{A}(g_n^1 \vee g_n^2) = \Omega(\sqrt{n})$, *and*

(iii) there exists a 2-multilective VLSI program S computing $g_n^1 \vee g_n^2$ with $\mathrm{A}(S) = O(1)$.

Proof. To show *(i)* we give the idea how to construct a VLSI circuit S_1 computing g_n^1 only (the exact technical description of S_1 is left as an exercise to the reader). At the beginning S_1 reads $x_{1,1}$, saves the value of $x_{1,1}$ in a short cycle, and consecutively reading $x_{1,2}, \ldots, x_{1,m}$ checks whether the value of every variable from $\{x_{1,2}, \ldots, x_{1,m}\}$ is equal to the value of $x_{1,1}$ saved in the cycle. Then the values of $x_{1,m}$ and $x_{2,1}$ are read and compared. If the values are not equal then the value stored in the cycle is changed to its negation (else the value of the cycle is not changed). After that the values of $x_{2,2}, \ldots, x_{2,m}$ are sequentially compared with the value stored in the cycle. Generally, for every $i \in \{2, \ldots, m-1\}$, the values of $x_{i,m}$ and of $x_{i+1,1}$ are compared in order to save the value of $x_{i+1,1}$ in the cycle and to consecutively check whether $x_{i+1,1} \equiv x_{i+1,2} \equiv \ldots \equiv x_{i+1,m}$.

We observe that g_n^2 can be obtained from g_n^1 by permuting the names of the variables and so the same approach works to show $\mathrm{A}(g_n^2) = O(1)$.

To prove the claim *(ii)* we recall the proof of Lemma 3.3.4.2 providing $\mathrm{cc}(g_n^1 \vee g_n^2) \geq \sqrt{n}/2$. Note that this proof can be used without any change to obtain $\mathrm{acc}(g_n^1 \vee g_n^2) \geq \sqrt{n}/2$. Since $\mathrm{acc}_1(f) \geq \mathrm{acc}(f)$ for any Boolean function f, Theorem 4.3.2.1 provides $\mathrm{A}(g_n^1 \vee g_n^2) \geq \sqrt{n}/2$.

To prove the claim *(iii)* it is sufficient to consider a VLSI program S as the union of the above designed VLSI programs S_1 and S_2 for g_n^1 and g_n^2 respectively with a new processor computing the disjunction of the values of the output processors of S_1 and S_2. Obviously $\mathrm{A}(S) = O(1)$ because $\mathrm{A}(S_i) = O(1)$ for $i = 1, 2$. S is 2-multilective because each input variable enters S exactly twice via distinct input processors. □

We observe that the crucial point in the proof of Lemma 4.5.2.1 was in the fact that to compute g_n^1 in $O(1)$ area requires reading the inputs in the sequence $x_{1,1}, x_{1,2}, \ldots, x_{1,m}, x_{2,1}, \ldots, x_{2,m}, \ldots, x_{m,1}, x_{m,2}, \ldots, x_{m,m}$, while to compute g_n^2 in $O(1)$ area the inputs should be read in the order $x_{1,1}, x_{2,1} \ldots, x_{m,1}, x_{1,2}, \ldots, x_{m,2}$, $\ldots, x_{1,m}, x_{2,m}, \ldots, x_{m,m}$. Since these two orders of input variables are essentially different, there exists no sequence of input variables which is easy for both g_n^1 and g_n^2. Thus, to compute $\{g_n^1, g_n^2\}$ (or $g_n^1 \vee g_n^2$) is much harder than to compute one of these functions.

Lemma 4.5.2.1 shows that already 2-multilectivity can bring an essential increase of computational power to VLSI programs. If one wants to have a similar result for AT^2 complexity instead of A, then one has to deal with the requirements on the placement of the input processors (variables) instead of the order in which the input variables are read. This means that two Boolean functions with essentially different requirements on the placement of input variables have to be found. To do this is left to the reader as an advanced exercise.

4.5.3 Lower Bounds on Multilective VLSI programs

To prove $\mathrm{A}(f) \geq \mathrm{acc}_1(f)$ in Section 4.3.2 we have found a time unit corresponding to an almost balanced partition of input variables into the set of input variables read before the time unit t and the set of input vari-

ables read after the time unit. Proving a lower bound on the area complexity of r-multilective VLSI programs requires a deeper idea extending the simple approach mentioned above. The idea here is to find $2r$ time units $t_1, t_2, \ldots, t_{2r}$ such that the set X_1 of input variables read in time intervals $[0, t_1], [t_2 + 1, t_3], [t_4 + 1, t_5], \ldots, [t_{2r-2} + 1, t_{2r-1}]$ and set X_2 of input variables read in time intervals $[t_1 + 1, t_2], [t_3 + 1, t_4], \ldots, [t_{2r-1} + 1, t_{2r}]$ have "large" subsets $U = X_1 - X_2$ and $V = X_2 - X_1$. So, the configurations in the time units $t_1, t_2, \ldots, t_{2r}$ can be considered as messages flowing between two computers C_I and C_{II}, where C_I has obtained the values of input variables of X_1 and C_{II} has obtained the values of input variables of X_2. Since X_1 and X_2 need not to be disjoint we cannot speak about our standard protocol model with almost balanced partition of input variables. But we can speak about a protocol with an "overlapping" partition, where there are large sets U and V of input variables assigned to one of the computers C_I and C_{II} only. Thus, if we show the existence of the time units $t_1, t_2, \ldots, t_{2r}$ with the above properties, then the $2r$-rounds communication complexity according to "overlapping" partitions will provide lower bounds on the area of r-multilective VLSI programs. To reach this we give the following combinatorial lemma.

Lemma 4.5.3.1 *Let n, m and r be positive integers, $m \leq n/3^{2r}$, $r < \frac{1}{2}\log_2 n$. Let $X \supseteq \{u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_n\}$ be a set of at least $2n$ elements. Let $W = W_0, W_1, \ldots, W_d$ be a sequence of subsets of X with the properties $|W_i| \leq m$ for every $i = 1, \ldots, d$, and for every $x \in X$, x belongs to at most r sets of W. Then there exists $U \subseteq \{u_1, \ldots, u_n\}$,$V \subseteq \{v_1, \ldots, v_n\}$, $b \in \mathbb{N}$, and integers $t_0 = -1, t_1, \ldots, t_b$ such that the following five conditions hold:*

(i) $|U| \geq n/3^{2r}$, $|V| \geq n/3^{2r}$,

(ii) $b \leq 2r$, $t_a \in \{1, \ldots, d\}$ *for* $a = 1, 2, \ldots, b$, *and* $t_0 < t_1 < \ldots < t_b$,

(iii) *if* $U \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right) \neq \emptyset$ *for some* $i \in \{0, \ldots, b-1\}$, *then* $V \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right) = \emptyset$, *and*

(iv) *if* $V \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right) \neq \emptyset$ *for some* $i \in \{0, \ldots, b-1\}$, *then* $U \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right) = \emptyset$,

(v) $(U \cup V) \cap \left(\bigcup_{j=t_b+1}^{d} W_j\right) = \emptyset$.

Proof. To construct U and V we inductively define sets U_i, V_i, and the integers t_i for $i = 0, 1, \ldots, b$ in such a way that they satisfy the following properties:

(a) $U_{i+1} \subseteq U_i$ and $V_{i+1} \subseteq V_i$,

(b) $|U_i| \geq n/3^i$ and $|V_i| \geq n/3^i$,

(c) if $U_{i+1} \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right) \neq \emptyset$ then $V_{i+1} \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right) = \emptyset$,

(d) if $V_{i+1} \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right) \neq \emptyset$ then $U_{i+1} \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right) = \emptyset$.

At the beginning we set $U_0 = \{u_1, \ldots, u_n\}$, $V_0 = \{v_1, \ldots, v_n\}$, and $t_0 = -1$. We now describe the inductive step by determining U_{i+1}, V_{i+1}, and t_{i+1} if U_i, V_i, and $t_0, \ldots, t_i$ with the properties (a), (b), (c), (d) are known. We distinguish two cases according to the sets $W_{t_i+1}, W_{t_i+2} \ldots, W_d$.

(1) Let $|U_i \cap \left(\bigcup_{j=t_i+1}^{d} W_j\right)| < |U_i|/3$ and $|V_i \cap \left(\bigcup_{j=t_i+1}^{d} W_j\right)| < |V_i|/3$. We set $t_{i+1} = d$, $U_{i+1} = U_i - (U_i \cap \left(\bigcup_{j=t_i+1}^{d} W_j\right))$, and $V_{i+1} = V_i - \left(V_i \cap \left(\bigcup_{j=t_i+1}^{d} W_j\right)\right)$. Obviously all properties (a), (b), (c), and (d) hold for U_{i+1}. Further, we stop the process by setting $b = i + 1$.

(2) Let $|U_i \cap \left(\bigcup_{j=t_i+1}^{d} W_j\right)| \geq |U_i|/3$ or $|V_i \cap \left(\bigcup_{j=t_i+1}^{d} W_j\right)| \geq |V_i|/3$. We choose $t_{i+1} \in \{t_i + 1, t_i + 2, \ldots, d\}$ as the smallest integer with the property $|U_i \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right)| \geq |U_i|/3$ or $|V_i \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right)| \geq |V_i|/3$. Without loss of generality we assume $|U_i \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right)| \geq |U_i|/3$ (the other case can be handled analogously). Since for every $a \in \{1, \ldots, d\}$, $|W_a| \leq m \leq n/3^{2r} \leq |U_i|/3$ (and similarly $|W_a| \leq |V_i|/3$), we have

$$|U_i|/3 \leq \left|U_i \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right)\right| \leq |U_i|/3 + |W_{t_{i+1}}| \leq 2 \cdot |U_i|/3,$$

and

$$\left|V_i \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right)\right| \leq |V_i|/3 + |W_{t_{i+1}}| \leq 2 \cdot |V_i|/3.$$

We set $U_{i+1} = U_i \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right)$ and $V_{i+1} = V_i - (V_i \cap \left(\bigcup_{j=t_i+1}^{t_{i+1}} W_j\right))$. Clearly (a), (c), (d) hold for V_{i+1}, U_{i+1}, and t_{i+1}, and the condition (b) follows from the above inequalities.

If the process of the construction of U_i's and V_i's does not stop because of the case (1), we stop for $i = 2r$ and set $b = 2r$.

Now, the integers $t_0, t_1, \ldots, t_b$ are determined, and we set $U = U_b$ and $V = V_b$. Because of the property (b) of U_i's and V_i's, the condition *(i)* holds for U and V. Since $U \subseteq U_i$ ($V \subseteq V_i$) for every $i = 0, \ldots, b$ the conditions (c) and (d) imply *(iii)* and *(iv)*. Obviously t_i were chosen in such a way that *(ii)* holds.

It remains to show *(v)*, i.e., that $\bigcup_{j=t_b+1}^{d} W_j$ does not contain any element of U and V. If the inductive process of the construction of U_i's and V_i's halts because of the case (1), then $t_b = d$ (i.e., $\bigcup_{j=t_b+1}^{d} W_j = \emptyset$) and *(v)* holds. If this does not happen, then $b = 2r$. Moreover, all t_i's ($i = 1, \ldots, 2k$) were chosen according to the case (2). Let p be the number of indexes $e \in \{0, 1, \ldots, 2r - 1\}$ such that U_{e+1} was chosen as $U_e \cap (\bigcup_{j=t_e+1}^{t_{e+1}} W_j)$ and $V_{e+1} = V_e - (V_e \cap (\bigcup_{j=t_e+1}^{t_{e+1}} W_j))$. (Note that in this case every element of U occurs at least once in $\bigcup_{j=t_e+1}^{t_{e+1}} W_j$.) Let q be the number of indexes $h \in \{0, 1, \ldots, 2r - 1\}$ such that V_{h+1} was chosen as $V_h \cap (\bigcup_{j=t_h+1}^{t_{h+1}} W_j)$ and $U_{h+1} = U_h - (U_h \cap (\bigcup_{j=t_h+1}^{t_{h+1}} W_j))$.

(Note that in this case every element of V occurs at least once in $\bigcup_{j=t_h+1}^{t_{h+1}} W_j$.) We know that $p+q=b=2r$. Moreover, $p \leq r$ and $q \leq r$ because each element of $X \supseteq U \cup V$ can occur in at most r distinct sets from $W = W_1, \ldots, W_d$ and each of the p (q) choices of U_e (V_h) requires at least one occurrence of every element of U (V) in $\bigcup_{j=t_e+1}^{t_{e+1}} W_j$ ($\bigcup_{j=t_h+1}^{t_{h+1}} W_j$). Thus, $p=r$ and $q=r$ which means that no element from $U \cup V$ can be in $\bigcup_{j=t_{2k+1}}^{d} W_j$. This completes the proof of Lemma 4.5.3.1. □

We want to use Lemma 4.5.3.1 to prove lower bounds on complexity measures of multilective VLSI programs. The first idea is to define the sets W_i as the set of input variables read by a multilective VLSI program S in the i-th configuration. If $d = \mathrm{T}(S)$, $\mathrm{A}(S) \leq m$, and S is k-multilective, then all the assumptions of Lemma 4.5.3.1 are fulfilled. Then the sets U and V define an "overlapping" partition $\Pi = (\Pi_L, \Pi_R)$ of X with $U \subseteq \Pi_L$ and $V \subseteq \Pi_R$. We observe that we can construct a k-rounds protocol $\langle \Pi, \Theta \rangle$ computing the same function as S does and communicating the messages coding the t_j-th configurations of S for $j = 1, \ldots, b$. The trouble is that we cannot close our consideration at this point because we have never defined and investigated communication complexity measures over non-disjoint partitions of input variables. So, we now define a communication complexity measure suitable for proving lower bounds on multilective VLSI circuits. Note that we do not need to change our computing model (protocol) of two communicating computers, only to consider it for a broader class of partitions of input variables.

Definition 4.5.3.2 *Let $f: \{0,1\}^n \to \{0,1\}$ be a Boolean function with a set of variables $X = \{x_1, \ldots, x_n\}$. Let $U_0 \subseteq X$, $V_0 \subseteq X$, $|U_0| = |V_0|$ be two disjoint subsets of X. Let k be a positive integer. A pair $\Pi = (\Pi_L, \Pi_R)$ is called a* **(U_0, V_0, k)-overlapping partition of X** *if*

(i) $\Pi_L \cup \Pi_R = X$,

(ii) there exist $U \subseteq U_0 \cap \Pi_L$ and $V \subseteq V_0 \cap \Pi_R$ such that $U \cap \Pi_R = V \cap \Pi_L = \emptyset$ and $|U| \geq |U_0|/3^{2k}$, $|V| \geq |V_0|/3^{2k}$.

$\mathrm{Par}(X, U_0, V_0, k)$ *denotes the set of all (U_0, V_0, k)-overlapping partitions of X.*

For every $\Pi \in \mathrm{Par}(X, U_0, V_0, k)$ we define the **overlapping $2k$-rounds communication complexity of f according to Π** *as*

$$\mathbf{occ_{2k}}(f, \Pi) = \min\{\mathrm{cc}\langle \Pi, \Phi \rangle \mid \langle \Pi, \Phi \rangle \text{ is a } 2k\text{-round protocol computing } f \text{ for a communication function } \Phi\}.$$

For all disjoint subsets $U_0, V_0 \subseteq X$ we define the **overlapping $2k$-rounds communication complexity of f according to U_0 and V_0** *as*

$$\mathbf{occ_{2k}}(f, U_0, V_0) = \min\{\mathrm{occ}_{2k}(f, \Pi) \mid \Pi \in \mathrm{Par}(X, U_0, V_0, k)\}.$$

Finally, the **overlapping $2k$-rounds communication complexity of f** *is*

$$\mathbf{occ_{2k}}(f) = \max\{\mathrm{occ}_{2k}(f, U_0, V_0) \mid U_0 \subseteq X, V_0 \subseteq X, |U_0| = |V_0| \geq |X|/8, U_0 \cap V_0 = \emptyset\}.$$

Observation 4.5.3.3 *Let k be a positive integer. For every Boolean function $f \in B_2^n$, $n \in \mathbb{N}$,*

$$\mathrm{occ}_{2k}(f) \leq n/(2 \cdot 3^{2k}).$$

Proof. Let X, $|X| = n$, be the set of the input variables of f. For any choice of two disjoint subsets U_0 and V_0 of X, $|U_0| \leq n/2$ or $|V_0| \leq n/2$. Without loss of generality one can assume $|U_0| \leq n/2$. Then there is an overlapping partition $\Pi = (\Pi_L, \Pi_R) \in \mathrm{Par}(X, U_0, V_0, k)$ such that there are sets $U \subseteq U_0 \cap \Pi_R$, $V \subseteq V_0 \cap \Pi_L$ with $|U| = n/(2 \cdot 3^{2k})$, $\Pi_L = X - V$, and $\Pi_R = X - U$ (i.e., $\Pi_L \cap \Pi_R = X - (U \cup V)$). Obviously there is a one-way protocol $\langle \Pi, \Phi \rangle$ computing f by submitting the values assigned to the variables in U from the first computer to the second computer. Thus $\mathrm{occ}_{2k}(f, U_0, V_0) \leq |U| = n/(2 \cdot 3^{2k})$. □

Before using $\mathrm{occ}_{2k}(f)$ to get lower bounds on k-multilective circuits we have to show that we are able to prove high lower bounds on $\mathrm{occ}_{2k}(h_n(L))$ for specific languages L. Rather than proving a lower bound on a specific language we give a strategy how to prove $\Omega(n/3^{2k})$ lower bounds for so called "shift languages". The shift languages have been considered as languages with linear communication complexity or with linear s-communication complexity in Chapter 2. They can be specified as follows. Let $h_n(L)$ be defined over the set of input variables $X = X_1 \cup X_2 \cup X_3$, where $X_i \cap X_j = \emptyset$ for $i \neq j$, $i, j \in \{1, 2, 3\}$ and $|X_1| \geq |X|/4$, $|X_2| \geq |X|/4$. We say that L is a **shift language** if the values of variables in X_3 determine which pairs $(u, v) \in X_1 \times X_2$ must have the same values (have to be compared). An example of such a language is L_{choice} from Theorem 2.6.3.1. To prove $\mathrm{occ}_{2k}(h_n(L)) \geq n/(4 \cdot 3^{2k})$ for a shift language L we choose $U_0 = X_1$ and $V_0 = X_2$. Now, it is sufficient to prove $\mathrm{occ}_{2k}(h_n(L), X_1, X_2) \geq n/(4 \cdot 3^{2k})$, i.e., to prove $\mathrm{occ}_{2k}(h_n(L), \Pi) \geq n/(4 \cdot 3^{2k})$ for every $\Pi \in \mathrm{Par}(X, X_1, X_2, k)$.

Let $\Pi = (\Pi_L, \Pi_R)$ be an arbitrary (X_1, X_2, k)-overlapping partition of X. Then there exists $U \subseteq X_1 \cap \Pi_L$ and $V \subseteq X_2 \cap \Pi_R$ such that $U \cap \Pi_R = V \cap \Pi_L = \emptyset$ and $|U| \geq |X_1|/3^{2k}$, $|V| \geq |X_2|/3^{2k}$. Since $|X_i| \geq n/4$ for $i = 1, 2$ we have $|U| \geq n/(4 \cdot 3^{2k})$ and $|V| \geq n/(4 \cdot 3^{2k})$. Now, we can take an input assignment of variables in X_3 requiring $u \equiv v$ for $d = n/(4 \cdot 3^{2k})$ different pairs $(u_1, v_1), (u_2, v_2), \ldots (u_d, v_d)$ from $U \times V$ such that $|\{u_1, \ldots, u_d\}| = d$ and $|\{v_1, \ldots, v_d\}| = d$. Applying the fooling set method or the rank method in the standard way we obtain $\mathrm{occ}_{2k}(f, \Pi) \geq d = n/(4 \cdot 3^{2k})$. Thus, if k is a constant independent of n, then we have proved $\mathrm{occ}_{2k}(h_n(L)) = \Omega(n)$.

In what follows we apply overlapping $2k$-rounds communication complexity to get lower bounds on the area of k-multilective VLSI circuits.

Theorem 4.5.3.4 *Let k and n' be positive integers, $k < \frac{1}{2}\log_2 n' - 2$. Let $f \in B_2^{n'}$ be a Boolean function. Then, for every k-multilective VLSI program S computing f,*

$$A(S) \geq P(S) \geq occ_{2k}(f)/2k.$$

Proof. Let $\overline{X}$ be the set of input variables of f, and let U_0, V_0 be subsets of $\overline{X}$ such that $occ_{2k}(f) = occ_{2k}(f, U_0, V_0)$. Note that $|U_0| = |V_0| \geq |\overline{X}|/8$. Obviously it is sufficient to show that $P(S) \geq occ_{2k}(f, U_0, V_0)/2k$.

Let $|U_0| = |V_0| = n$. We set $X = U_0 \cup V_0$ and $W_i = \{x \in U_0 \cup V_0 \mid x$ is read by S in the i-th time unit$\}$ for $i = 0, 1, \ldots, T(S)$. We distinguish two possibilities according to the cardinalities of W_i's.

(1) Let there exist a $j \in \{0, 1, \ldots, T(S)\}$ such that $|W_j| > m = n/3^{2k}$. Then S has more than $n/3^{2k}$ input processors and so $P(S) \geq n/3^{2k}$. Following Observation 4.5.3.3 we see $occ_{2k}(f) \leq n/(2 \cdot 3^{2k})$ and so $P(S) \geq occ_{2k}(f)/2 \geq occ_{2k}(f)/2k$ for every positive integer k.

(2) Let, for every $j \in \{0, 1, \ldots, T(S)\}$, $|W_j| \leq m = n/3^{2k}$. We have $k \leq \frac{1}{2}\log_2 n$ because $n \geq n'/8$ and $k < \frac{1}{2}\log_2 n' - 2$. So $X, W_0, W_1, W_2, \ldots, W_{T(S)}$, k, m, and n fulfill all assumptions of Lemma 4.5.3.1 which implies the existence of $U \subseteq U_0$, $V \subseteq V_0$, $b \in \mathbb{N}$, $t_0 = -1, t_1, \ldots, t_b \in \{1, \ldots, T(S)\}$ satisfying the conditions *(i), (ii), (iii),(iv)* and *(v)* of Lemma 4.5.3.1. We use this to describe a b-round protocol D computing f. $D = \langle \Pi, \Phi \rangle$, where

(a) $\Pi = (\Pi_L, \Pi_R)$ for $\Pi_L = \overline{X} - V$ and $\Pi_R = \overline{X} - U$, and

(b) Φ works as follows. Without loss of generality we assume $U \subseteq \bigcup_{j=t_0+1}^{t_1} W_j$ (i.e., $V \cap \bigcup_{j=0}^{t_1} W_j = \emptyset$). (If $V \subseteq \bigcup_{j=0}^{t_1} W_j$ then we take $\Pi_L = \overline{X} - U$ and $\Pi_R = \overline{X} - V$ and the proof continues in the same way.) In the first round the first computer sends a word coding the t_1-th configuration of S working on a given input. Generally, for all $i = 1, \ldots, b-1$, if $U \subseteq \bigcup_{j=t_i+1}^{t_{i+1}} W_j$, then the first computer sends a word coding the t_{i+1}-th configuration to the second computer. If $V \subseteq \bigcup_{j=t_i+1}^{t_{i+1}} W_j$, then the second computer sends a word coding the t_{i+1}-th configuration to the first computer. We observe that if the first (second) computer knows the t_i-th configuration, and the values of all variables in $X - V \supseteq \bigcup_{j=t_i+1}^{t_{i+1}} W_j$ (all variables in $X - U \supseteq \bigcup_{j=t_i+1}^{t_{i+1}} W_j$), then it can unambiguously compute the t_{i+1}-th configuration of S on the given input.

In Section 4.3 we have already observed that one can code any configuration of S as a word of length $P(S)$. Since the communication consists of exactly $b \leq 2k$ messages, D is a $2k$-rounds protocol working within the communication complexity $b \cdot P(S) \leq 2k \cdot P(S)$. We observe that D communicating as described above can compute f because $(U \cup V) \cap \bigcup_{j=t_b+1}^{T(S)} W_j = \emptyset$ and every computer knowing the t_b-th configuration $C_{t_b}^{\alpha}$ and the values

of variables in $X - (U \cup V) \supseteq \bigcup_{j=t_b+1}^{T(S)} W_j$ of the input assignment α can compute $f(\alpha)$. Further, $\Pi \in \mathrm{Par}(\overline{X}, U_0, V_0, k)$ which together with the above facts imply

$$2k \cdot \mathrm{P}(S) \geq \mathrm{cc}(D) \geq \mathrm{occ}_{2k}(f, U_0, V_0).$$

□

We see that the idea of the proof of Theorem 4.5.3.4 was based on taking $W_0, W_1, \ldots, W_d$ as sets of variables read in the time units $0, 1, \ldots, d$ respectively. To get a lower bound on the AT^2 complexity of k-multilective VLSI programs it is sufficient to divide the corresponding VLSI circuit into some small areas and to choose W_i's as sets of variables read in these areas. The precise strategy is given in the proof of the following theorem.

Theorem 4.5.3.5 *Let k and n' be positive integers, $k < \frac{1}{2}\log_2 n' - 2$. Let $f \in B_2^{n'}$ be a Boolean function depending on all its n' variables. Then, for every k-multilective VLSI program S computing f,*

$$\mathrm{A}(S) \cdot (\mathrm{T}(S))^2 \geq (\mathrm{occ}_{2k}(f)/4k)^2.$$

Proof. Let $\overline{X}$ be the set of input variables of f, and let U_0, V_0 be such subsets of $\overline{X}$ that $\mathrm{occ}_{2k}(f) = \mathrm{occ}_{2k}(f, U_0, V_0)$. We show $\mathrm{A}(S) \cdot (\mathrm{T}(S))^2 \geq \mathrm{occ}_{2k}(f, U_0, V_0)$.

Let $|U_0| = |V_0| = n$. We set $X = U_0 \cup V_0$. Let G_S be the layout of S of the size $a \times d$, $a \geq d \geq 1$. Let $p_0, p_1, \ldots, p_{\mathrm{P}(S)-1}$ be the sequence of all processors of S lexicographically ordered according to its position in the layout G_S (i.e., the processor in the square (r_1, s_1) is before the processor on the position (r_2, s_2) if $s_1 < s_2$ or $[s_1 = s_2$ and $r_1 < r_2]$). For every $i \in \{0, 1, \ldots, \mathrm{P}(S) - 1\}$ we define W_i as the set of variables from X read by the processor p_i during the first $\mathrm{T}(S)$ steps of the computation of S. Similarly as in the proof of Theorem 4.5.3.4 we distinguish two possibilities according to the cardinalities of W_i's.

(1) Let there exist a $j \in \{0, 1, \ldots, P(S) - 1\}$ such that $|W_j| > m = n/3^{2k}$. Then $\mathrm{T(S)} \geq n/3^{2k}$ because p_j can read at most one input variable in one time unit. So, $(\mathrm{T(S)})^2 \geq (n/3^{2k})^2$. Since $\mathrm{occ}_{2k}(f) \leq n/(2 \cdot 3^{2k})$ [Observation 4.5.3.3], we have $(\mathrm{T(S)})^2 \geq (\mathrm{occ}_{2k}(f))^2$.

(2) Let, for every $j \in \{0, 1, \ldots, \mathrm{P(S)} - 1\}$, $|W_j| \leq m = n/3^{2k}$. Since $X, W_0, W_1, \ldots, W_{\mathrm{P(S)}-1}, k, m, n$ fulfill all assumptions of Lemma 4.5.3.1, there exist $U \subseteq U_0, V \subseteq V_0, b \in \mathbb{N}, t_0 = -1, t_1, \ldots, t_b \in \{0, \ldots, \mathrm{P(S)} - 1\}$ satisfying conditions $(i), (ii), (iii), (iv)$, and (v) of Lemma 4.5.3.1. Let, for $i = 0, 1, \ldots, b$,

$$X_i = \bigcup_{j=t_i+1}^{t_{i+1}} W_j.$$

We observe that one can partition the layout of S into $b+1$ parts $S_0, \ldots, S_b$ in such a way that S_i contains the processors reading the input variables from X_i and the border between S_i and S_{i+1} is of length at most $d+1$ [see Figure 4.3]. Now, we define an overlapping partition $\Pi = (\Pi_L, \Pi_R)$ with $\Pi_L = \overline{X} - V$ and $\Pi_R = \overline{X} - U$. Corresponding to Π we consider the partition of S into S_L and S_R, where S_L is the union of such $S_i's$ that $U \subseteq X_i$ (i.e., $V \cap X_i = \emptyset$). The number of edges flowing between S_L and S_R is so bounded by $(d+1) \cdot b \leq (d+1) \cdot 2k$. This means that at most $(d+1) \cdot 2k$ bits may flow between S_L and S_R in one time unit. We obtain

$$(d+1) \cdot 2k \cdot T(S) \geq occ_{2k}(f),$$

and so

$$2 \cdot A(S) \cdot (T(S))^2 \geq ((d+1) \cdot T(S))^2 \geq (occ_{2k}(f))^2/4k^2.$$

□

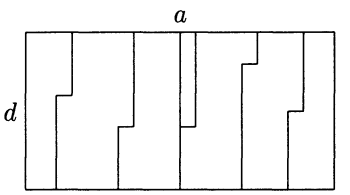

Fig. 4.3. A partition of the layout of a VLSI circuit

4.5.4 Exercises

Exercise 4.5.4.1 * *Find a sequence F of Boolean functions such that the AT^2 complexity of F is large but F is easy for 2-multilective VLSI circuits.*

Exercise 4.5.4.2 *Prove lower bounds and upper bounds on the overlapping communication complexity of the following languages:*

(a) $L = \{ww \mid w \in \{0,1\}^*\}$,
*(b)** L_Δ *from Theorem 2.3.3.2,*
(c) L_R *from Theorem 2.3.5.4,*
(d) L_{sm} *from Theorem 2.6.3.2, and*
(e) L_{dcf} *from Theorem 2.6.3.3.*

Exercise 4.5.4.3 * *Define one-way overlapping communication complexity and compare it with overlapping communication complexity. Find a Boolean function for which these two measures essentially differ.*

Exercise 4.5.4.4 *Prove an assertion about lower bounds on* $A(S)$ *analogous to Theorem 4.5.3.4 if the k-multilective VLSI program S fulfills the additional condition that all input processors lie on the border of the layout of S.*

Exercise 4.5.4.5 *Prove an assertion about lower bounds on* $A(S) \cdot (T(S))^2$ *analogous to Theorem 4.5.3.5 if the layout of the k-multilective VLSI program S fulfills the additional condition that all input processors lie on the border of the layout of S.*

Exercise 4.5.4.6 *Prove a theorem analogous to Theorem 4.5.3.5 for the three-dimensional layout of k-multilective VLSI programs and for the three-dimensional layout with the input processors on the layout border.*

Exercise 4.5.4.7 * *Prove, that for every constant k, there is a language* L_k *such that there is an exponential gap between* $occ_k(h_n(L_k))$ *and* $occ_{k+1}(h_n(L_k))$.

Exercise 4.5.4.8 *Define Las Vegas and Monte Carlo versions of overlapping communication complexity and relate them to k-multilective probabilistic VLSI programs.*

4.5.5 Problems

Problem 4.5.5.1 * *Find another method providing lower bounds on* A *and* AT^2 *complexity measures of multilective VLSI circuits which may yield higher lower bounds than Theorem 4.5.3.4 and 4.5.3.5 for some concrete Boolean functions.*

Problem 4.5.5.2 * *Find, for every* $k \in \mathbb{N}$, *a language* L_k *such that* $A(L_k)[A(L_k) \cdot (T(L_k))^2]$ *is essentially larger for k-multilective VLSI programs than for* $(k+1)$ *-multilective VLSI programs.*

4.6 Bibliographical Remarks

The fundamental reference on the design rules of VLSI circuits is Mead and Conway [MC80]. The basis for the formal mathematical model of VLSI circuits was given by Thompson [Th79, Th80], Brent and Kung [BK81], where the basic complexity measures time and area were introduced and investigated. Bilardi, Pracchi, and Preparata [BPP81] and Chazelle and Monier [CM81] introduced the length of wires as a complexity measure influencing the resulting time complexity of VLSI circuits. The notion of a VLSI program has been introduced in this book to achieve a formal description compatible with the description of Boolean circuits in Chapter 3.

It is a lot of work designing VLSI circuits for distinct computing problems. Because we are interested in the lower bounds here we have omitted the

presentation of them. Some nice examples for fundamental problems like integer multiplication, matrix multiplication, and Fourier transforms can be found for instance by Abelson and Andreae [AA80], Brent and Kung [BK80, BK81], Preparata and Vuillemin [PV80, PV81], and Savage [Sa84].

The first ideas for proving lower bounds on the complexity measures of VLSI circuits were presented by Vuillemin [Vu80] and Thompson [Th79, Th80]. Lipton and Sedgewick [LS81] generalized the ideas of Vuillemin and Thompson and developed the fooling set technique as the basic method for proving lower bounds on VLSI computations. The more general approach used in this book is based on communication complexity introduced by Yao [Ya81]. The comparison results between the distinct layout models considered here are based on Hromkovič [Hr88b] and the ideas used already for the layouts of Boolean circuits in Chapter 3. The three-dimensional layout of VLSI circuits was considered by Hromkovič [Hr88a], Leighton and Rosenberg [LR81], and Rosenberg [Ro81].

The use of communication complexity for proving special lower bounds for VLSI circuits with some restrictions on their topology was considered in [Hr88b]. The presented example of lower bounds for one-dimensional arrays is from Hromkovič and Procházka [HrP88, HrP94]. The presented lower bounds on interconnection networks with powerful processors are due to Hromkovič [Hr92].

Multilective VLSI circuits were first considered by Kedem and Zorat [KZ81, KZ81a]. In these papers and in [Sa84] effective multilective VLSI algorithms were designed. Lemma 4.5.2.1 showing that 2-multilective VLSI programs are more powerful than semilective VLSI programs have been proved by Gubáš and Waczulík [GW87]. Savage [Sa84] and Yao [Ya81] developed the first lower bounds for such algorithms. The lower bounds methods presented here are based on the approaches used by Ďuriš and Galil [DG93], Hromkovič, Krause, Meinel, and Waack [HKMW92], and Ullman [Ul84]. The presentation here consolidates these approaches by bounding them on the common combinatorial Lemma 4.5.3.1.

For more information about VLSI circuits, especially about VLSI algorithms and VLSI design tools, one has to consult the excellent monographs of Golze [Go96], Lengauer [Le90, Le90a], and Ullman [Ul84].

5. Sequential Computations

5.1 Introduction

In the previous two chapters we have shown several applications of communication complexity for proving lower bounds on parallel complexity measures. In this chapter we have chosen some examples illustrating the power of communication complexity method for proving lower bounds on complexity of sequential computations. Since the central topic of this book is the relation between communication complexity and parallel computing we do not try to give a complete overview of the relation between communication complexity and sequential complexity measures. The main goal of this chapter is to extend and deepen our view on the nature of communication complexity and its applicability in complexity theory.

The typical application of the communication complexity method in the previous chapters has been based on proving lower bounds on the amount of information which has to be exchanged between different processors of parallel computing models considered. In sequential information processing one has only one processor and thus the idea above cannot work. But one should not be surprised by the fact that communication complexity can be applied to obtain lower bound on sequential computations. In sequential computing models we have information flows between the input and the memory, between distinct parts of memory, between distinct parts of inputs, and between different time units of the sequential computation. All the well-known classical lower bound proof methods using entropy, Kolmogorov complexity, and the crossing sequence argument are in fact based on information-theoretic arguments, i.e., on some transfer of information during the computation. From this point of view communication complexity seems to be of the same nature as the methods mentioned above, and we show that in proving lower bounds it can be as successful as those methods.

This chapter is organized as follows. Section 5.2 presents a stronger connection between one-way communication complexity and finite automata than the connection presented in Section 2.3.5. Section 5.3 relates communication complexity to the time and space complexities of various Turing machine models. The results presented there show the connection between the crossing sequence argument for proving lower bounds and the communication complexity method. Section 5.4 is devoted to the relations between communication complexity and

the complexity measures of decision trees and branching programs. The bibliographical remarks of Section 5.4 conclude the chapter.

5.2 Finite Automata

5.2.1 Introduction

In this section we present a closer relation between one-way communication complexity and the number of states of finite automata than the one presented in the proof of Theorem 2.3.5.1 of Section 2.3.5. There we have seen that the one-way communication complexity of the Boolean function $h_n(L)$ for a regular language L provides a direct lower bound on the logarithm of the number of states of the minimal finite automaton recognizing L. This fact alone is not so important because one knows well-working lower bound methods for the size of the minimal finite automaton for a given language L. This size can be even computed in an algorithmic way. But this relation can be straightforwardly extended to the nondeterministic case. This is already of interest because we did not have any powerful general method for proving lower bounds on the size of the minimal nondeterministic automaton recognizing a regular language. Since we have good lower bound methods for nondeterministic communication complexity (for instance, 1-fooling sets), we win general methods for proving lower bounds on the size of minimal nondeterministic finite automata.

The considerations above show that, for every $n \in \mathbb{N}$, the one-way communication complexity of $h_n(L)$ provides lower bounds on the size of the minimal automaton for L. It may happen that one has trouble finding a suitable n such that $cc_1(h_n(L))$ is close to two to the number of states of the minimal finite automaton for L, or even that such an n does not exists. A good example is that of regular languages over a one-letter alphabet, for which the one-way communication complexity is zero or one. This trouble is caused by the fact that communication protocols are a nonuniform computing model (one uses an infinite number of them to recognize a language), whereas finite automata are a uniform (algorithmic) computing model. In order to establish a closer relation between communication complexity and finite automata, we introduce a uniform model of two-party communication protocols. We show that the number of distinct messages used by the optimal one-way uniform communication protocol recognizing a regular language L is equal to the number of states of the minimal finite automaton for L. The equality overcomes the trouble mentioned above and provides the possibility to generalize the use of the lower bound proof methods for communication complexity to prove lower bounds on the size of finite automata.

Section 5.2 is organized as follows. Section 5.2.2 provides the basic definitions of finite automata and related notions. Some basic well-known facts about finite automata are presented there, too. In Section 5.2.3 the relation between one-way (nondeterministic) communication complexity and the size of (nonde-

terministic) finite automata is explicitly formulated. In Section 5.2.4 a uniform model of deterministic and nondeterministic communication protocols is defined and applied to the size of finite automata. In the deterministic case we even show that, for every regular language L, the number of distinct messages of the optimal one-way uniform protocol for L is equal to the size of the minimal automaton for L. Section 5.2.5 present some exercises, and some research problems are formulated in Section 5.2.6.

5.2.2 Definitions

Despite the fact that we assume that the reader is familiar with basic formal language notions like regular languages and finite automata we give a formal definition of finite automata in order to fix our notation.

Definition 5.2.2.1 *A* **finite automaton (FA)** M *is a 5-tuple* $(Q, \Sigma, \delta, q_0, F)$, *where*

(i) Q *is a finite* **set of states**, *each element* $p \in Q$ *is called a* **state**,

(ii) Σ *is an* **input alphabet**,

(iii) $\delta : Q \times \Sigma \to Q$ *is a* **transition function**,

(iv) $F \subseteq Q$ *is a* **set of final states**, *each element* $q \in F$ *a* **final (accepting) state**,

(v) $q_0 \in Q$ *is the* **initial state**.

We define $\hat{\delta}_M : Q \times \Sigma^* \to Q$ *as*

1. $\hat{\delta}_M(q, \lambda) = q$ *for every* $q \in Q$, *and*
2. $\hat{\delta}_M(q, wa) = \delta(\hat{\delta}_M(q, w), a)$ *for every* $w \in \Sigma^*$ *and every* $a \in \Sigma$.

A word $x \in \Sigma^*$ *is said to be* **accepted by** M *if* $\hat{\delta}_M(q_0, x) \in F$. *The* **language accepted by** M, *designated* $L(M)$, *is the set* $\{x \mid \hat{\delta}_M(q_0, x) \in F\}$. *The* **size of** M *is* $\mathbf{s}(M) = |Q|$.

Definition 5.2.2.2 *A* **nondeterministic finite automaton (NFA)** *is a 5-tuple* $A = (Q, \Sigma, \delta, q_0, F)$, *where*

(i) Q *is a finite* **set of states**,

(ii) Σ *is an* **input alphabet**,

(iii) $\delta : Q \times \Sigma \to 2^Q$ (2^Q *is the power set of* Q) *is a* **transition function**,

(iv) $F \subseteq Q$ *is a* **set of final states**,

(v) $q_0 \in Q$ *is the* **initial state**.

We define $\hat{\delta}_A : Q \times \Sigma^* \to 2^Q$ *as*

1. $\hat{\delta}_A(q, \lambda) = \{q\}$ *for every* $q \in Q$, *and*
2. $\hat{\delta}_A(q, wa) = \{p \mid p \in \delta(r, a)$ *for some* $r \in \hat{\delta}_A(q, w)\}$ *for every* $q \in Q$ *and every* $a \in \Sigma$.

A word $x \in \Sigma^*$ *is said to be* **accepted by** A *if* $\hat{\delta}_A(q_0, x) \cap F \neq \emptyset$. *The* **language accepted by** A, $L(A)$, *is the set* $\{x \mid \hat{\delta}_A(q_0, x) \cap F \neq \emptyset\}$. *The* **size of** A *is* $\mathbf{s}(A) = |Q|$.

Definition 5.2.2.3 *A language* L *is called* **regular** *if there exists an* FA M *such that* $L = L(M)$. *For every regular language* L,

$$\mathbf{s}(L) = \min\{\mathrm{s}(M) \mid M \text{ is an FA such that } L(M) = L\},$$

and

$$\mathbf{ns}(L) = \min\{\mathrm{s}(A) \mid A \text{ is an NFA such that } L(A) = L\}.$$

An FA B *is called* **minimal for** L *if* $L(B) = L$ *and* $\mathrm{s}(L) = \mathrm{s}(B)$.

Definition 5.2.2.4 *Let* Σ *be an alphabet and let* L *be a language over* Σ. *An equivalence relation* R *on* Σ^* *is called* **right invariant** *if for all* $x, y \in \Sigma^*$,

$$x \; R \; y \text{ implies } xz \; R \; yz \text{ for all } z \in \Sigma^*.$$

We associate with L *a relation* R_L *on* Σ^* *defined as follows:*

$$x \; R_L \; y \quad \Leftrightarrow \quad \text{for each } z \text{ either both or neither of } xz \text{ and } yz \text{ is in } L.$$

Observation 5.2.2.5 *Let* Σ *be an alphabet. For every language* L *over* Σ, R_L *is a right invariant equivalence relation.*

In what follows we denote, for every equivalence relation R, the **index of** R (the number of equivalence classes) by $\mathbf{i}(R)$.

Lemma 5.2.2.6 *(A consequence of the* **Myhill–Nerode theorem***). For every regular language* L

$$\mathrm{s}(L) = \mathrm{i}(R_L).$$

Proof. Let L be a regular language over an alphabet Σ. First we show $\mathrm{i}(R_L) \leq \mathrm{s}(L)$. Let $M = (Q, \Sigma, \Delta, q_0, F)$ be a minimal FA for L. We define a relation R_M on Σ^* as follows:

$$x \; R_M \; y \quad \Leftrightarrow \quad \hat{\delta}_M(q_0, x) = \hat{\delta}_M(q_0, y).$$

R_M is a right invariant equivalence relation on Σ^* because, for any $z \in \Sigma^*$, if $\hat{\delta}_M(q_0, x) = \hat{\delta}_M(q_0, y)$, then $\hat{\delta}_M(q_0, xz) = \hat{\delta}_M(\hat{\delta}_M(q_0, x), z) = \hat{\delta}_M(\hat{\delta}_M(q_0, y), z) = \hat{\delta}_M(q_0, yz)$ [that R_M is reflexiv, symmetric and transitive is obvious].

The consequence is that xR_My implies xR_Ly, i.e., R_M is a refinement of R_L (every equivalence class of R_M is entirely contained in some equivalence class or R_L). Thus $\mathrm{i}(R_L) \leq \mathrm{i}(R_M) = \mathrm{s}(L)$.

Now, we prove $\mathrm{s}(L) \leq \mathrm{i}(R_L)$ by constructing an FA $A = (Q', \Sigma, \delta_A, q_0', F')$ such that $L(A) = L$ and $\mathrm{s}(A) = \mathrm{i}(R_L)$. We set Q' to be the finite set of the equivalence classes of R_L and denote by $[x]$ the element of Q' containing x. We define

$$\delta_A([x], a) = [xa]$$

for any $x \in \Sigma^*$ and any $a \in \Sigma$. This definition is consistent since R_L is right invariant (if xR_Ly then xaR_Lya, i.e. $[xa] = [ya]$). We set $q_0' = [\varepsilon]$ and $F' = \{[x] \mid x \in L\}$. Thus $L = L(A)$ and $\mathrm{s}(A) = |Q'| = \mathrm{i}(R_L)$. □

5.2.3 One-Way Communication Complexity and Lower Bounds on the Size of Finite Automata

In this short section we explicitly formulate the relations between one-way communication complexity and nondeterministic communication complexity on one side and $\mathrm{s}(L)$ and $\mathrm{ns}(L)$ on the other. In the deterministic case this relation has been already mentioned in the proof of Theorem 2.3.5.1. In what follows $X_n = \{x_1, \ldots, x_n\}$ and $\overline{\Pi}_n = (\{x_1, x_2, \ldots, x_{\lceil n/2 \rceil}\}, \{x_{\lceil n/2 \rceil + 1}, \ldots, x_n\})$ for every $n \in \mathbb{N}$.

Theorem 5.2.3.1 *For every regular language L over the alphabet $\{0,1\}$ and every $n \in \mathbb{N}$, $\mathrm{cc}_1(h_n(L), \overline{\Pi}_n) \leq \lceil \log_2(\mathrm{s}(L)) \rceil$.*

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a minimal finite automaton accepting L. Let $Q = \{q_0, q_1, \ldots, q_{k-1}\}$ and $d = \lceil \log_2 k \rceil$. For every $n \in \mathbb{N}$ we construct a protocol $D_n = \langle \overline{\Pi}_n, \Phi_n \rangle$ computing $h_n(L)$ as follows:

(i) $\overline{\Pi}_n = (\{x_1, x_2, \ldots, x_{\lceil n/2 \rceil}\}, \{x_{\lceil n/2 \rceil + 1}, \ldots, x_n\})$,

(ii) For every $x \in \{0,1\}^{\lceil n/2 \rceil}$: $\Phi_n(x, \lambda) = \mathrm{BIN}_d^{-1}(r)$, where $q_r = \hat{\delta}_A(q_0, x)$.
For every $y \in \{0,1\}^{\lfloor n/2 \rfloor}$ and every $c \in \{0,1\}^d$:
$\Phi_n(y, c) = \bar{1}$ if $\hat{\delta}_A(q_m, y) \in F$ and $m = \mathrm{BIN}(c)$,
$\Phi_n(y, c) = \bar{0}$ if $\hat{\delta}_A(q_m, y) \notin F$ and $m = \mathrm{BIN}(c)$.

Obviously D_n computes $h_n(L)$ and $\mathrm{cc}_1(h_n(L), \overline{\Pi}) \leq \mathrm{cc}_1(D_n) = d = \lceil \log_2 |Q| \rceil = \lceil \log_2(\mathrm{s}(L)) \rceil$. □

This assertion is not of great interest because one knows effective methods to construct the minimal FA for a regular language L and so to estimate $\mathrm{s}(L)$. The basic idea is to take an arbitrary FA $M = (Q, \Sigma, \delta, q_0, F)$ accepting L and

then to join any two states p and q of M if, for every $z \in \Sigma^*$, $\hat{\delta}_M(p,z) \in F \iff \hat{\delta}_M(q,z) \in F$. Lemma 5.2.2.6 shows that the minimal FA for L is an FA A for which no pair of states can be joined from the above reason.

For nondeterministic finite automata the situation is completely different. We do not know any efficient method for computing a minimal NFA for a given regular language L or for estimating $\mathrm{ns}(L)$. Nondeterministic communication complexity provides the first general method for proving lower bounds on $\mathrm{ns}(L)$ for regular languages.

Theorem 5.2.3.2 *For every regular language L over the alphabet $\{0,1\}$ and every $n \in \mathbb{N}$,*

$$\mathrm{ncc}_1(h_n(L), \overline{\Pi_n}) \leq \lceil \log_2(\mathrm{ns}(L)) \rceil.$$

Proof. The proof is almost the same as the previous one. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a NFA accepting $L = L(A)$. A nondeterministic protocol $D_n = \langle \overline{\Pi_n}, \Phi_n \rangle$ with communication complexity $\lceil \log_2 |Q| \rceil$ can be constructed as follows. For every input x of C_I, C_I nondeterministically chooses one of the states of $\hat{\delta}_A(q_0, x)$ and sends its binary code to C_{II}. If C_{II} receives the code of a state p and y is the input of C_{II}, then C_{II} computes $\bar{1}$ ($\bar{0}$) if and only if $\hat{\delta}_A(p,y) \cap F \neq \emptyset$ ($\hat{\delta}_A(p,y) \cap F = \emptyset$). □

We recall the fact that $\mathrm{ncc}_1(f, \Pi) = \mathrm{ncc}(f, \Pi)$ for every Boolean function f and every partition of their input variables Π. Thus, one can use the methods for proving lower bounds on nondeterministic communication complexity from Section 2.5.3 to get lower bounds on $\mathrm{ns}(L)$ of a given regular language L. The cover method can estimate $\mathrm{ncc}(h_n(L), \overline{\Pi})$ precisely, but it is not constructive and so usually very hard to use. The most practical method seems to be the nfool method. To get a lower bound on $\mathrm{ncc}(h_n(L), \overline{\Pi})$, it is sufficient to construct a 1-fooling set for $h_n(L)$ and $\overline{\Pi}_n$ only.

5.2.4 Uniform Protocols

For several regular languages Theorems 5.2.3.1 and 5.2.3.2 can provide good lower bounds on $\mathrm{s}(L)$ and $\mathrm{ns}(L)$, repectively. But there are many cases where they cannot help to get reasonable lower bounds. For instance, for any regular language L having only a few words of any fixed length, $\mathrm{cc}(h_n(L), \overline{\Pi}_n)$ is small but $\mathrm{s}(L)$ and $\mathrm{ns}(L)$ may be large. This drawback of our approach in the previous section is caused by the fact that the two-party protocols are a nonuniform computing model while finite automata are a uniform computing model. Here we overcome this trouble by defining one-way uniform protocols for the recognition of languages.

As we have seen in Chapter 2, the most convenient representation of a Boolean function f to be computed by a protocol was the Boolean matrix $M(f, \Pi)$. To represent the computing problem of the recognition of a language,

we shall use a representation of this problem by an infinite two-dimensional Boolean matrix defined as follows.

Definition 5.2.4.1 *Let $\Sigma = \{a_0, \ldots, a_{k-1}\}$ be an alphabet. For any two words $x, y \in \Sigma^*$ we say that x is before y according to the* **canonical order for Σ^*** *if*

(i) $|x| < |y|$ *or*

(ii) $|x| = |y|$, $x = zx_1x'$, $y = zy_2y'$, *where* $z, x', y' \in \Sigma^*$ *and* $x_1 = a_i$, $y_2 = a_j$ *for some* $i < j \leq k-1$.

An example of the canonical order of words for the alphabet $\{0,1,2\}$ is $\lambda, 0, 1, 2, 00, 01, 02, 10, 11, 12, 20, 21, 22, 000, 001, \ldots$.

Definition 5.2.4.2 *Let Σ be an alphabet and let $w_1, w_2, w_3, \ldots$ be the canonical order of words from Σ^*. Let $L \subseteq \Sigma^*$. We define the infinite Boolean matrix* $\boldsymbol{M(L, \Sigma)} = \{a_{ij}\}_{i,j \geq 1}$ *in such a way that* $a_{ij} = 1 \iff w_i w_j \in L$.

One can observe that, for each word $x \in L$, $M(L, \Sigma)$ contains $|x|+1$ elements claiming $x \in L$. So, this representation of L contains many repetitions. But this representation is suitable for our purposes because we shall require from a uniform protocol that it recognizes $x \in L$ for every partition of x into such z and y that $x = zy$.

Informally, we define a one-way uniform protocol as follows. As before, we consider a protocol consisting of two computers C_I and C_{II}, each one having a part of the input. Let the input of C_I be x, and let the input of C_{II} be y. C_I starts the computation by sending a binary message to C_{II}. C_{II} receiving the message decides whether $xy \in L$ or not. Obviously, we have to take care over consistency – if $xy = uv$, then the one-way uniform protocol may not take a different decision on xy than on uv.

Definition 5.2.4.3 *Let Σ be an alphabet and let $L \subseteq \Sigma^*$. A* **one-way uniform protocol over Σ** *is a pair $D = \langle \Phi, \varphi \rangle$, where:*

(i) $\Phi : \Sigma^* \longrightarrow \{0,1\}^*$ *is a function having the prefix freeness property, and*

(ii) $\varphi : \Sigma^* \times \{0,1\}^* \longrightarrow \{\bar{0}, \bar{1}\}$ *is a function.*

We say $\boldsymbol{D = \langle \Phi, \varphi \rangle}$ **accepts** $\boldsymbol{L}$, $\boldsymbol{L(D) = L}$, *if for all* $x, y \in \Sigma^*$: $\varphi(y, \Phi(x)) = \bar{1} \iff xy \in L$.

The **message complexity of the protocol** $\boldsymbol{D}$ *is* $|\{\Phi(x) \mid x \in \Sigma^*\}|$, *denoted* $\mathbf{mc(D)}$. *If* $\mathrm{mc}(D)$ *is finite, we define the* **communication complexity of the protocol** $\boldsymbol{D}$ *as* $\mathbf{cc(D)} = \max\{|\Phi(x)| \mid x \in \Sigma^*\}$.

Using the same idea as in the proof of Theorem 5.2.3.1, we observe that for every regular language L, there is a uniform protocol D accepting L within a finite message complexity.

Lemma 5.2.4.4 *Let L be a regular language over an alphabet Σ. Then there exists a uniform protocol D accepting L with* $\mathrm{mc}(D) = \mathrm{s}(L)$.

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a minimal FA for L. For every $x \in \Sigma$, C_I submits the code of the state $\hat{\delta}_A(q_0, x)$ to C_{II}. After this C_{II} having an input y computes $\bar{1}$ if and only if $\hat{\delta}_A(\hat{\delta}_A(q_0, x), y) \in F$.

□

In what follows we consider only uniform protocols with finite message complexity.

Definition 5.2.4.5 *Let Σ be an alphabet and let L be a regular language over Σ. The* **message complexity of L** *is* $\mathbf{mc(L)} = \min\{\mathrm{mc}(D) \mid D \text{ is a one-way uniform protocol accepting } L\}$. *The* **one-way uniform communication complexity of L** *is*

$$\mathbf{ucc_1(L)} = \min\{\mathrm{cc}(D) \mid D \text{ is a one-way uniform protocol accepting } L\}.$$

Observation 5.2.4.6 *For every regular language L*

$$\mathrm{ucc}_1(L) = \lceil \log_2(\mathrm{mc}(L)) \rceil.$$

Proof. For every uniform protocol using k messages one can encode the messages as binary strings of the same length $\lceil \log_2 k \rceil$. □

Observation 5.2.4.7 *For every regular language $L \subseteq \Sigma^*$ and every $n \in \mathbb{N}$,*

$$\mathrm{cc}_1(h_n(L)) \leq \mathrm{ucc}_1(L).$$

Proof. Let, for every $n \in \mathbb{N}$, $\overline{\Pi}_n$ denote the partition

$$(\{x_1, \ldots, x_{\lceil n/2 \rceil}\}, \{x_{\lceil n/2 \rceil + 1}, \ldots, x_n\}).$$

Since $M(h_n(L), \overline{\Pi}_n)$ is a submatrix of the matrix $M(L, \Sigma)$, the assertion follows.

□

The main goal of this section is to obtain the following strong relation between one-way uniform protocols and finite automata.

Theorem 5.2.4.8 *For every regular language L,*

$$\mathrm{s}(L) = \mathrm{mc}(L).$$

Proof. $\mathrm{mc}(L) \leq \mathrm{s}(L)$ has been already shown in Lemma 5.2.4.4. To prove $\mathrm{s}(L) = \mathrm{mc}(L)$ for every regular language L, we first show that $\mathrm{mc}(L)$ is equal

to the number of different rows of the matrix $M(L, \Sigma)$, and then that $s(L)$ is equal to the number of different rows of $M(L, \Sigma)$, too. In Section 2.4.3 we have shown $2^{cc_1(f,\overline{\Pi})}$ is equal to the number of different rows of $M(f, \overline{\Pi})$. The idea of this proof is the same. Let $D = \langle \Phi, \varphi \rangle$ be a one-way uniform protocol accepting a regular language $L \subseteq \Sigma^*$. Let $x, y \in \Sigma^*$ be two words such that the row corresponding to x in $M(L, \Sigma)$ differs from the row corresponding to y in $M(L, \Sigma)$. Obviously $\Phi(x) \neq \Phi(y)$ because there exists $z \in \Sigma^*$ such that either ($xz \notin L$ and $yz \in L$) or ($xz \in L$ and $yz \notin L$). Thus, the number of different messages of D is at least the number of different rows of $M(L, \Sigma)$. On the other hand one can construct a one-way uniform protocol $D' = \langle \Phi, \varphi' \rangle$ in such a way that $\Phi'(x) = \Phi'(y)$ if and only if the rows of $M(L, \Sigma)$ corresponding to the words x and y are equal. So, $mc(L)$ is equal to the number of different rows of the matrix $M(L, \Sigma)$.

Now we observe that, for any FA $M = (Q, \Sigma, \delta, q_0, F)$ accepting L, $\hat{\delta}_M(q_0, x) \neq \hat{\delta}_M(q_0, y)$ if there exists a word z such that $xz \in L$ and $yz \notin L$. In other words if the rows of $M(L, \Sigma)$ corresponding to x and y are different, then $\hat{\delta}_M(q_0, x)$ is another state than $\hat{\delta}_M(q_0, y)$. Thus, $|Q|$ is at least the number of distinct rows of $M(L, \Sigma)$.

To show that $s(L)$ is at most the number of different rows of $M(L, \Sigma)$, it is sufficient to define a right invariant equivalence relation R on Σ^* such that $i(R)$ is equal to the number of different rows of $M(L, \Sigma)$. This can be simply done by saying xRy if and only if the rows of $M(L, \Sigma)$ corresponding to x and y are the same. Obviously, R is reflexive, symmetric, and transitive. For every $z \in \Sigma^*$, xRy implies $xzRyz$ because the rows corresponding to xz and yz must be equal. (Note that the infinite row for xz (yz) can be obtained by choosing those elements from the row x (y) which correspond to columns $zw_1, zw_2, zw_3, \ldots$, where $w_1, w_2, w_3, \ldots$ is the canonical order of the words in Σ^*). □

Thus, Theorem 5.2.4.8 provides a new method for estimating $s(L)$. Despite of the fact we have effective algorithms estimating $s(L)$ (even constructing the minimal FA for L), Theorem 5.2.4.8 can be useful. We say this because, for some regular languages L, the use of Theorem 5.2.4.8 results in a very quick estimation of $s(L)$ and in the process of computing $s(L)$ we do not need to construct any FA for L. We illustrate it on the following example. Let $L_k = \{w \in \{0,1\}^* \mid |w| \mod k \equiv 0\}$. One can immediately observe that the k rows $\lambda, 0, 0^2, \ldots, 0^{k-1}$ of $M(L_k, \{0,1\})$ are different and that $M(L_k, \{0,1\})$ contains only such rows. Thus $s(L_k) = k$. The languages L_k are also a good example for the comparison of Theorem 5.2.3.1 and Theorem 5.2.4.8. Theorem 5.2.3.1 does not provide any reasonable lower bound because $cc_1(h_n(L_k), \overline{\Pi}) = 0$ for all $k, n \in \mathbb{N}$, whereas Theorem 5.2.4.8 estimates $s(L_k) = k$ in a simple way. The typical use of Theorem 5.2.4.8 for proving lower bounds on $s(L)$ may look as follows. One chooses a "suitable" finite submatrix M' of $M(L, \Sigma)$ and estimates the number of different rows of M'. Obviously, if $M(L, \Sigma)$ has m different rows, then there exists a finite submatrix of $M(L, \Sigma)$ having m different rows, too.

Thus, the crucial point in proving a good lower bound on $s(L)$ is the choice of this finite submatrix. The principal advantage of this method over Theorem 5.2.3.1 is that the submatrix chosen need not correspond to rows (columns) with the same input length, which is the case for the matrices $M(h_n(L), \overline{\Pi})$.

As we have already mentioned, the main open problem is to find an effective algorithm estimating $ns(L)$ for any regular language L. We are not able to solve this problem here, but we can improve the lower bound method of Theorem 5.2.3.2 in a similar way as we have improved the method of Theorem 5.2.3.1 in the deterministic case.

Definition 5.2.4.9 *Let Σ be an alphabet, and let L be a regular language over Σ. A* **one-way uniform nondeterministic protocol over Σ** *is a pair $D = \langle \Phi, \varphi \rangle$, where:*

(i) $\Phi : \Sigma^* \to 2^{\{0,1\}^*}$ is a function fulfilling the following properties:

(i.1) Φ has the "prefix freeness property" (i.e., if $z \in \Phi(x)$ and $u \in \Phi(y)$ then neither u is a proper prefix of z nor z is a proper prefix of u),

(i.2) for every $x \in \Sigma^*$, $\Phi(x)$ is a finite set, and

(i.3) the set $\{\Phi(x) \mid x \in \Sigma^*\}$ is finite;

(ii) $\varphi : \Sigma^* \times \{0,1\}^* \to \{\bar{0}, \bar{1}\}$ is a function.

A **computation** *of D on a word $x = x_1x_2$ is a word $u\$r$, where $u \in \Phi(x_1)$ and $r = \varphi(x_2, u)$. In what follows we call $u\$r$ a* **computation of D on the partition x_1, x_2 of the word x**, *too. A computation $u\$r$ is called* **accepting (rejecting)** *if $r = \bar{1}$ ($\bar{0}$). The* **message complexity of the protocol D** *is* **$\mathrm{nmc}(D) = |\{u \in \{0,1\}^* \mid u \in \Phi(x)$ for some $x \in \Sigma^*\}|$.** *We say* **$D = \langle \Phi, \varphi \rangle$ accepts L, $L(D) = L$,** *if, for all $x, y \in \Sigma^*$, there exists an accepting computation of D on the partition x, y of the word $xy \Leftrightarrow xy \in L$.*

The **nondeterministic message complexity of L** *is*

$$\mathbf{nmc}(L) = \min\{\mathrm{nmc}(D) \mid D \text{ is a one-way uniform nondeterministic protocol accepting } L\}.$$

The **communication complexity of D** *is*

$$\mathbf{cc}(D) = \max\{|u| \mid u \in \Phi(x) \text{ for some } x \in \Sigma^*\}.$$

The **one-way uniform nondeterministic communication complexity of L** *is*

$$\mathbf{ucc_1}(L) = \min\{\mathrm{cc}(D) \mid D \text{ is a one-way uniform nondeterministic protocol accepting } L\}.$$

As in Theorem 5.2.3.2, one can establish the following relation.

Theorem 5.2.4.10 *For every regular language L*

$$\mathrm{nmc}(L) \leq \mathrm{ns}(L).$$

Proof. It is sufficient to show that, for every NFA $A = (Q, \Sigma, \delta, q_0, F)$, there exists a one-way uniform nondeterministic protocol $D = \langle \Phi, \varphi \rangle$ such that $L(A) = L(D)$ and $\mathrm{nmc}(D) = |Q|$. Let $k = \lceil \log_2 |Q| \rceil$ and let code : $Q \to \{0,1\}^k$ be an injective function. Then D can be defined as follows:

(i) $\forall x \in \Sigma^* \; : \; \Phi(x) = \{\mathrm{code}(p) \mid p \in \hat{\delta}_A(q_0, x)\}$, *and*

(ii) $\forall y \in \Sigma^* \; : \; \varphi(y, \mathrm{code}(p)) = \bar{1}$ if $\hat{\delta}_A(p, y) \cap F \neq \emptyset$,
$\varphi(y, \mathrm{code}(p)) = \bar{0}$ if $\hat{\delta}_A(p, y) \cap F = \emptyset$.

□

Now, as in the uniform deterministic case, to prove a lower bound on $\mathrm{ns}(L)$ one can choose a finite submatrix M' of $M(L, \Sigma)$ and prove that M' possesses a large 1-fooling set. The advantage of this approach over Theorem 5.2.4.2 is again in the fact that M need not to be a matrix whose columns (rows) correspond to words of the same length. To illustrate this advantage, consider again the language $L_k \subseteq \{0,1\}^*$ for any $k \in \mathbb{N}$. Let us consider the $k \times k$ submatrix M' of $M(L_k, \{0,1\})$ lying on the intersection of the rows $0^1, 0^2, \ldots, 0^k$ and the columns $\lambda, 0^1, 0^2, \ldots, 0^{k-1}$. One can simply observe that this matrix M' has 1's only on the diagonal from the lower-left corner to the upper-right corner. Thus we can say that the communication problem represented by M' possesses a "generalized" 1-fooling set $\mathcal{A} = \{(0^k, \lambda), (0^{k-1}, 0^1), (0^{k-2}, 0^2), \ldots, (0, 0^{k-1})\}$. Generalized means that $\mathcal{A}$ contains pairs (x, y) with $xy \in L$ instead of words $xy \in L$ as in Definition 2.2.2.14. If (x_1, x_2) and (y_1, y_2) are in a generalized 1-fooling set, then $x_1x_2 \in L$, $y_1y_2 \in L$, and one of the words x_1y_2, y_1x_2 is not in L. So, a generalized fooling set may contain several partitions of the same word. For instance $\mathcal{A}$ contains k partitions of 0^k. This is impossible for the standard definition of fooling sets because each input word may be at most once in a fooling set. Obviously, the generalized interpretation of the notion fooling set does not change anything on the fact that the cardinality of any fooling set for $M(L, \Sigma)$ is a lower bound on $\mathrm{nmc}(L)$.

Another method for proving lower bounds on nondeterministic communication complexity in Section 2.5.3 was the cover method. Using the same idea as in the proof of Theorem 2.5.3.5 one can show that $\mathrm{nmc}(L)$ is exactly $\mathrm{cov}(M(L, \Sigma))$ — the size of the minimal cover of all 1's of $M(L, \Sigma)$ by 1-monochromatic submatrices (see Definition 2.5.3.4). Again, to apply this method one chooses a finite submatrix M' and then it remains to prove some lower bound on $\mathrm{cov}(M')$. This method may work very well if it can be easily proved that M' does not contain any large 1-monochromatic submatrix.

5.2.5 Exercises

Exercise 5.2.5.1 *Construct the minimal* FA *and the minimal* NFA *for any language* $U_k = \{x \in \{0,1\}^* \mid \#_0(x) \bmod k \equiv 0\}$.

Exercise 5.2.5.2 *Prove that every language* L *with a finite* $\mathrm{mc}(L)$ *is regular.*

Exercise 5.2.5.3 *Construct a one-way uniform protocol* D *which does not accept any language.*

Exercise 5.2.5.4 *Construct a one-way uniform protocol accepting:*

(i) $U_k = \{x \in \{0,1\}^* \mid \#_0(x) \bmod k \equiv 0\}$, *for any* $k \in \mathbb{N}$;

(ii) $L_1 = \{x001100y \mid xy \in \{0,1\}^*\}$; *and*

(iii) $L = \{wcw \mid w \in \{0,1\}^*\}$.

Exercise 5.2.5.5 *Construct a one-way uniform nondeterministic protocol accepting:*

(i) $U_k = \{x \in \{0,1\}^* \mid \#_0(x) \bmod k \equiv 0\}$ *for any* $k \in \mathbb{N}$; *and*

(ii) $L' = \{x0110y1001z \mid x,y,z \in \{0,1\}^*\}$.

Exercise 5.2.5.6 *Find a regular language* L *such that there exists* $n \in \mathbb{N}$ *such that* $\log_2(\mathrm{s}(L)) = \mathrm{ncc}_1(h_n(L), \overline{\Pi}_n)$

Exercise 5.2.5.7 *Find an infinite sequence of regular languages* $V_1, V_2, \ldots$, *such that* $\mathrm{s}(V_k) = k$ *for all* $k \in \mathbb{N}$, *and* $\mathrm{cc}_1(h_n(V_j), \overline{\Pi}_n) \leq 1$ *for all* $n, j \in \mathbb{N}$.

Exercise 5.2.5.8 *Find an infinite sequence* $Z_1, Z_2, \ldots$ *of regular languages such that, for any* $k \in \mathbb{N}$, *there exists an* $n_k \in \mathbb{N}$ *with*

$$\log_2(\mathrm{ns}(Z_k)) = \mathrm{ncc}_1(h_{n_k}(Z_k), \overline{\Pi}_{n_k}) = k.$$

Exercise 5.2.5.9 * *Find, for every* $k \in \mathbb{N}$, *a regular language* L_k *with* $\mathrm{s}(L_k) = 2^k$ *and* $\mathrm{ns}(L_k) = k$.

Exercise 5.2.5.10 *Give an infinite Boolean matrix that differs from any* $M(L, \{0,1\})$ *for any language* L.

Exercise 5.2.5.11 *Prove that if a one-way uniform nondeterministic protocol* D *accepts a language* L, *then* L *is regular.*

Exercise 5.2.5.12 *Prove that every one-way uniform protocol D accepting $L = \{wcw \mid w \in \{0,1\}^*\}$ has an infinite* $\mathrm{mc}(D)$.

Exercise 5.2.5.13 * *Find an infinite sequence of regular languages $F_1, F_2, \ldots$ such that* $\mathrm{cc}_1(h_n(F_k), \overline{\Pi}_n) = 2$ *for all* $n, k \in \mathbb{N}$ *and* $\mathrm{mc}(F_k) = 2^k$ *for every* $k \in \mathbb{N}$.

Exercise 5.2.5.14 *Prove that for every regular language L,*

$$\mathrm{nmc}(L) = \mathrm{Cov}(M(L, \Sigma_L)).$$

Exercise 5.2.5.15 *Prove, that there is a regular language L such that* $\mathrm{nmc}(L)$ *essentially differs from the size of the minimal nondeterministic automaton for L.*

5.2.6 Research Problems

Problem 5.2.6.1 * *Find an efficient algorithm for an approximate estimation of* $\mathrm{ns}(L)$ *for any regular language L.*

5.3 Turing Machines

5.3.1 Introduction

This section is devoted to the complexity measures of Turing machine models and so to general sequential time and space complexity measures. Note that this is not for the first time we handle the time complexity measure in this book. The combinational complexity of Boolean circuits (the length of straight-line programs) is the number of elementary operations which have to be executed in order to compute a given Boolean function, and so it is a nonuniform sequential time complexity measure. From Chapter 3 we already know that nobody has been able to prove a superlinear lower bound on the combinational complexity of a concrete Boolean function. The same situation appears for uniform time complexity considered as the number of elementary operations of general algorithmic models of sequential computations. We do not have any superlinear lower bound on the time complexity of the recognition of a concrete language on multitape Turing machines or on register machines. On the other hand we know thousands of languages for which one even supposes the existence of superpolynomial lower bounds.

The lower bound proof methods using the crossing sequence argument or Kolmogorov complexity have been successful in proving some superlinear lower bounds on the time complexity of restricted models of sequential computations or on some time-space tradeoffs of general models of sequential computations.

In this section we show that one can use communication complexity in order to get this kind of lower bound too.

Section 5.3 is organized as follows. In Section 5.3.2 it is shown that the nondeterministic communication complexity squared provides a lower bound on the time complexity of the classical Turing machine consisting of one infinite tape only. In Section 5.3.3 we consider on-line and off-line multitape Turing machines. For on-line multitape Turing machines it is shown that one-way communication complexity provides lower bounds on the space complexity. For off-line multitape Turing machines it is shown that the nondeterministic communication complexity squared provides lower bounds on some time-space tradeoff. Section 5.3.4 involves some exercises and some open problems are formulated in Section 5.3.5.

In the whole Section 5.3 we assume that the reader is familiar with the basic Turing machine models and so we omit formal definitions of them.

5.3.2 Time Complexity of Classical Turing Machines

In this section we consider the original model of Turing machines, called simply Turing machines in what follows. A Turing machine, for short TM, consists of a finite state control, an infinite tape, and a read/write head. At the beginning of any computation of a TM A an input word is written on the tape, the head is positioned on the first symbol of the input word and the Turing machine A is in its initial state. In one computation step, A reads the tape symbol positioned under the head and (1) may change its state, (2) rewrites the symbol read, and (3) moves its head to the left or to the right. A accepts the input if it halts in an accepting state. If A halts in a rejecting state on an input w or if the computation of A on w is infinite, then w is not accepted by A. The language accepted by A is denoted by $L(A)$.

If $x \in L(A) \subseteq \Sigma^*$, then the **time complexity of the computation of A on x**, denoted $\mathbf{T_A(x)}$, is the number of steps of this computation. The **time complexity of A** is the function $T_A : \mathbb{N} \to \mathbb{N}$ defined as follows:

$$\forall n \in \mathbb{N},\ \mathbf{T_A(n)} = \max\{T_A(x) \mid x \in L(A) \cap \Sigma^n\}.$$

A nondeterministic Turing machine, shortly NTM, is a nondeterministic version of the Turing machine described above. A nondeterministic TM B accepts an input x if there exists an accepting computation of B (a computation finishing in an accepting state of B) on x. The **time complexity of the NTM B on x** is the number of steps (the length) of the shortest accepting computation of B on x. The **time complexity of B** is a function $T_B : \mathbb{N} \to \mathbb{N}$ defined as follows:

$$\forall n \in \mathbb{N},\ \mathbf{T_B(n)} = \max\{T_B(x) \mid x \in L(A) \cap \Sigma^n\}.$$

In what follows we show the connection between the crossing sequence argument and communication complexity in order to prove quadratic lower

bounds on the time complexity of nondeterministic Turing machines. This classical crossing sequence argument is very close to the communication complexity approach because communication complexity measures (in some sense) the amount of information which has to be exchanged between two parts of the input in a Turing machine computation. In order to use it we first define crossing sequences.

Definition 5.3.2.1 *Let A be an* NTM, *and let $x = x_1x_2\dots x_n, x_i \in \Sigma$ for some alphabet Σ, be an input word. For every i, the position of the tape of A containing x_i at the beginning of the computation of A on x is called the* **i-th position of the tape**. *Let B be a finite computation of A on x. For every $i \in \{1,2,\dots,n\}$, the* **i-th crossing sequence of the computation B on x, $\mathrm{CS}(B,i,x)$**, *is the sequence of states of A in which A moves its head either from the i-th position of the tape to the $(i+1)$-st position of the tape or from the $(i+1)$-st position of the tape to the i-th position of the tape in the computation B on x. $|\mathrm{CS}(B,i,x)|$ denotes the length of the sequence $\mathrm{CS}(B,i,x)$.*

Observation 5.3.2.2 *Let A be an* NTM *and let $x \in L(A), |x| = n$. Then*

$$\mathrm{T}_A(x) \geq \min\{\sum_{i=1}^{n} |\mathrm{CS}(B,i,x)| \quad | \ B \text{ is an accepting computation of } A \text{ on } x\}.$$

One can see that to prove a lower bound on $\mathrm{T}_A(x)$, it is sufficient to prove that the sum of the lengths of crossing sequences of any accepting computation on x is large enough. We observe that a crossing sequence $\mathrm{CS}(B,i,x)$ may be viewed as the information exchanged between the input parts $x_1x_2\dots x_i$ and $x_{i+1}\dots x_n$ in the computation B. Any state coming into $\mathrm{CS}(B,i,x)$ due to the movement of the head to the right from the $(i-1)$-th position to the i-th position can be considered as a communication from C_I with the input $x_1\dots x_i$ to C_{II} with the input $x_{i+1}\dots x_n$. Similary, the states of $\mathrm{CS}(B,i,x)$ connected with the movement of the head from the $(i+1)$-th position to the i-th position may be considered as a communication from C_{II} to C_I. Formalizing this idea we prove the following theorem.

Theorem 5.3.2.3 *Let $L \subseteq \{0,1\}^*$. For any $X_n = \{x_1, x_2, \dots, x_n\}$, let $\overline{\Pi}_n = (\{x_1, x_2, \dots, x_{\lceil n/2 \rceil}\}, \{x_{\lceil n/2 \rceil}, \dots, x_n\})$. For each Turing machine A accepting L,*

$$\mathrm{T}_A(n) = \Omega((\mathrm{ncc}(h_n(L), \overline{\Pi}_n))^2).$$

Proof. Let A be a TM accepting L, and let Q be the set of states of A. For every $n \in \mathbb{N}$ we consider the following. Let, for every $x \in L \cap \{0,1\}^n$, B_x be a computation of A on x such that $\mathrm{T}_A(x)$ is the number of steps of B_x. Let $c_n = \mathrm{ncc}(h_n(L), \overline{\Pi}_n)$, and let $q = \lceil \log_2 |Q| \rceil$. For every $x \in L \cap \{0,1\}^n$, let $\mathrm{T}_A(x) \leq r_n c_n$ for some positive real number r_n (i.e., $\mathrm{T}_A(n) \leq r_n c_n$). Then, for every $x \in L \cap \{0,1\}^n$,

$$r_n \cdot c_n \geq \sum_{i=1}^{n} |\mathrm{CS}(B_x, i, x)| \geq \sum_{i=\lceil n/2 \rceil - \lceil c_n/2 \rceil}^{\lceil n/2 \rceil} |\mathrm{CS}(B_x, i, x)|.$$

Thus, for every $x \in L \cap \{0,1\}^n$, there exists an $i_x \in \{1, \ldots, n\}$ such that

$$\mathrm{CS}(B_x, i_x, x)| \leq r_n/2.$$

Now, we describe a one-way nondeterministic protocol $D_n = \langle \overline{\Pi}_n, \bar{\Phi}_n \rangle$ computing $h_n(L)$ within communication complexity

$$\log_2(c_n/2) + q \cdot r_n/2 + c_n/2 + 1.$$

D_n computes as follows. For every $x \in L \cap \{0,1\}^n$, C_I nondeterministically guesses an $i \in \{\lceil n/2 \rceil - \lceil c_n/2 \rceil, \ldots, \lceil n/2 \rceil\}$ and a sequence C of states. Then C_I checks whether C is a good candidate for the i-th crossing sequence from its $(x_1 \ldots x_i)$ point of view (i.e., C_I checks whether all odd elements of C are correct if the states on even positions in C are given). If yes, then C_I submits $\lceil n/2 \rceil - i, C$, and $x_{\lceil n/2 \rceil - \lceil c_n/2 \rceil} \ldots x_{\lceil n/2 \rceil}$ to C_{II}. The length of this message is at most

$$\log_2(c_n/2) + q \cdot |C| + c_n/2 + 1.$$

Now, C_{II} checks whether the states on even positions of C are achievable if the states on odd positions of C are given. If yes, then clearly C is the i-th crossing sequence of a computation of A on x. C_{II} accepts if this computation finishes in an accepting state. Thus, for every $x \in L(A) \cap \{0,1\}^n$, there exists an accepting computation of D_n on x with the message of length

$$\log_2(c_n/2) + q \cdot |\mathrm{CS}(B_x, i_x, x)| + c_n/2 + 1 \leq \log_2(c_n/2) + q \cdot r_n/2 + c_n/2 + 1.$$

Since $\mathrm{cc}(D_n) \geq \mathrm{ncc}(h_n(L), \overline{\Pi}_n) = c_n$, we obtain:

$$\log_2(c_n/2) + q \cdot r_n/2 + c_n/2 + 1 \geq c_n$$

$$q \cdot r_n/2 + 1 \geq \tfrac{c_n}{2} - \log_2(c_n/2)$$

$$r_n \geq \tfrac{c_n}{q} - \tfrac{2}{q} \cdot \log_2 c_n.$$

Thus,

$$r_n = \Omega(c_n) \text{ and so } \mathrm{T}_A(n) = \Omega((c_n)^2) = \Omega((\mathrm{ncc}(h_n(L), \overline{\Pi}_n))^2).$$

□

We have already mentioned that one has proved quadratic lower bounds on the time complexity of the recognition of concrete languages on Turing machines by the crossing sequence argument. To achieve these results some lower bounds on the lengths of the crossing sequences were proved, usually, by the fooling set method. So, one advantage of Theorem 5.3.2.3 over the previous approaches is its generality. We apply the lower bounds on nondeterministic communication complexity directly for all languages instead of writing a special lower bound proof for each specific language. Further, we do not depend so much on the fooling set method only, because we have a more general machinery to handle nondeterministic communication complexity than the pure application of the fooling set method.

5.3.3 Sequential Space and Time-Space Complexity

In this section we consider very general models of sequential computations covering multitape Turing machines and register machines. A **sequential machine** is a machine consisting of one input tape with one read-only head and an infinite memory. The infinite memory consists of cells containing binary values. In one step a sequential machine A reads one symbol from the input tape and, depending on the symbol read and the content of the memory, changes a finite part of the memory and moves the head on the input tape. A **configuration of A** is a triple $C = (¢w\$, i, \alpha)$, where

(i) w is the input word,

(ii) $i \in \{0, 1, ..., |w| + 1\}$ is the position of the head on the input tape, and

(iii) $\alpha \in \{0, 1\}^*$ is the content of the memory.

For any configuration $C = (¢x\$, r, \beta)$, β is called the **internal configuration of C**. For any input w, the **initial configuration of A on w** (i.e., the configuration in which A starts the computation on w) is $(¢w\$, 1, \lambda)$.
A **computation of A on an input w** is a finite sequence $C_0, C_1, C_2, ..., C_k$ of configurations of A such that

(i) $C_0 = (¢w\$, 1, \lambda)$, and

(ii) C_i is reachable from C_{i-1} in one step for $i = 1, 2, ..., k$.

Some of the configurations of A are accepting and A **accepts** an input word if an accepting configuration has been reached. For any sequential machine M, **$L(M)$** denotes the set of words accepted by M.

We observe that sequential machines are a very general model of sequential computations. It is sufficient to specify the work with the memory by some restrictions and one obtains multitape (multidimensional) Turing machines or register machines. Moreover, the sequential machine model is non-uniform because one may change an arbitrarily large part of the memory in one simple computation step. Thus, lower bounds proved for our sequential machine model remain very largely valid.

In what follows we shall also consider **nondeterministic sequential machine** as a nondeterministic version of sequential machines.

Depending on the allowed direction of the head movement on the input tape we distinguish two versions of sequential machines. If the head may move to the right only we speak about an **on-line sequential machine**. If the head may move in both directions in the area between ¢ and \$ we speak about an **off-line sequential machine**.

Let A be a sequential machine, and let $C_0, C_1, \ldots, C_k$ be a computation on a word x: the **time complexity of the computation of A on x** is $\mathrm{T}_A(x) = k$. If $x \in L(A)$ and A is nondeterministic, then $\mathrm{T}_A(x) = \min\{|C| - 1 \mid C$ is an accepting computation of A on $x\}$. The **time complexity of A** is the

function $T_A : \mathbb{N} \to \mathbb{N}$ defined by $\mathbf{T_A(n)} = \max\{T_A(x) \mid x \in L(A) \cap \Sigma^n\}$ for any $n \in \mathbb{N}$.

Let A be a sequential machine and let $C = (\not\vdash x\$, 1, \lambda), (\not\vdash x\$, i_1, \alpha_1), \ldots, (\not\vdash x\$, i_k, \alpha_k)$ be a computation of A on x. The **space complexity of C** is $\mathrm{Space}(C) = \max\{|\alpha_i| \mid i = 1, \ldots, k\}$. If A is deterministic, then the **space complexity of the computation of A on x** is $\mathbf{S_A(x)} = \mathrm{Space}(C)$. If A is nondeterministic and $x \in L(A)$, then the **space complexity of A on x** is $\mathbf{S_A(x)} = \min\{\mathrm{Space}(C) \mid C$ is an accepting computation of A on $x\}$. The **space complexity of A** is the function $S_A : \mathbb{N} \to \mathbb{N}$ defined by $\mathbf{S_A(n)} = \max\{S_A(x) \mid x \in L(A) \cap \Sigma^n\}$ for any $n \in \mathbb{N}$.

First, we show that one-way communication complexity provides lower bounds on the space complexity of on-line sequential machines. The argument for this claim is almost the same as in the case of finite automata.

Theorem 5.3.3.1 *For every language L over the alphabet $\{0,1\}$, and every on-line sequential machine A accepting L,*

$$cc_1(h_n(L), \overline{\Pi}_n) \leq S_A(n) + 1$$

for any $n \in \mathbb{N}$.

Proof. The proof is almost the same as the proof of Theorem 5.2.3.1. One can construct a one-way protocol whose messages code the internal configurations of A after reading the first $\lceil n/2 \rceil$ bits of the input. □

The same argument as above yields the same result for the nondeterministic case.

Theorem 5.3.3.2 *For every language L over the alphabet $\{0,1\}$, and every on-line nondeterministic sequential machine M accepting L,*

$$ncc(h_n(L), \overline{\Pi}_n) \leq S_M(n) + 1$$

for any $n \in \mathbb{N}$.

The following result is an extension of Theorem 5.3.2.3 for sequential machines. It shows that nondeterministic communication complexity provides lower bounds on the tradeoff $T \cdot S$ of off-line sequential computations.

Theorem 5.3.3.3 *Let $L \subseteq \{0,1\}^*$, and let A be an off-line nondeterministic sequential machine accepting L. Then*

$$T_A(n) \cdot S_A(n) = \Omega((ncc(h_n(L), \overline{\Pi}_n))^2).$$

Proof. The argument is the same as in the proof of Theorem 5.3.2.3. Instead of considering the crossing sequences as sequences of states we consider the

crossing sequences as sequences of internal configurations here. Thus, instead of $q = \lceil \log_2(\text{the number of states}) \rceil$ in the proof of Theorem 5.3.2.3 one has $S_A(n)$ for the inputs of the length n. Assuming $T_A(x) \leq r_n c_n$ (r_n and c_n have the same meaning as in the proof of Theorem 5.3.2.3) one can construct a one-way nondeterministic protocol D_n computing $h_n(L)$ within the communication complexity

$$\log_2(c_n/2) + S_A(n) \cdot r_n/2 + c_n/2 + 1.$$

Since $cc(D_n) \geq ncc(h_n(L), \overline{\Pi}_n) = c_n$, we obtain

$$\log_2(c_n/2) + S_A(n) \cdot r_n/2 + c_n/2 + 1 \geq c_n$$

$$r_n \geq \frac{c_n}{S_A(n)} - \frac{2}{S_A(n)} \cdot \log_2 c_n.$$

Thus $r_n = \Omega(c_n/S_A(n))$ and so

$$T_A(n) \cdot S_A(n) = \Omega(\frac{c_n}{S_A(n)} \cdot c_n \cdot S_A(n)) = \Omega((c_n)^2).$$

□

5.3.4 Exercises

Exercise 5.3.4.1 *Explain why the proof of Theorem 5.3.2.3 does not work in the deterministic case.*

Exercise 5.3.4.2 * *Prove* $T_A(n) = \Omega(n)$ *for every nondeterministic Turing machine accepting the language* $\{ww | w \in \{0,1\}^*\}$ *without using the communication complexity argument (Theorem 5.3.2.3).*

Exercise 5.3.4.3 *For any language* $L \subseteq \{0,1\}^*$, *let* $L_{insert} = \{x2^i y \mid xy \in L, i \in \mathbb{N}\}$. *Prove that every nondeterministic Turing machine* M *accepting* L_{insert} *fulfills*

$$T_M(n) = \Omega(n \cdot ncc(h_n(L), \overline{\Pi}_n)).$$

Exercise 5.3.4.4 *Generalize Theorem 5.3.2.3 by allowing* $\overline{\Pi}_n$ *to be exchanged by an arbitrary almost balanced partition.*

Exercise 5.3.4.5 *Give a formal description of the protocol mentioned in the proof of Theorem 5.3.3.1. Note that one cannot directly use the internal configurations as messages because of the prefix-freeness property.*

Exercise 5.3.4.6 *Give a formal definition of crossing sequences of sequential machines, and give the full formal proof of Theorem 5.3.3.3.*

Exercise 5.3.4.7 *For any language* $L \subseteq \{0,1\}^*$, *let* $L_{insert} = \{x2^i y \mid xy \in L, i \in \mathbb{N}\}$. *Prove that every off-line nondeterministic sequential machine* M *accepting* L_{insert} *fulfills*

$$\mathrm{T}_M(n) \cdot \mathrm{S}_M(n) = \Omega(n \cdot \mathrm{ncc}(h_n(L), \overline{\Pi}_n)).$$

Exercise 5.3.4.8 * *For any language* $L \subseteq \{0,1\}^*$, *let* $L_{insert} = \{x2^i y \mid xy \in L,\ i \in \mathbb{N}\}$. *Prove that every deterministic Turing machine* M *accepting* L_{insert} *fulfills*

$$\mathrm{T}_M(n) = \Omega(n \cdot \mathrm{cc}(h_n(L), \overline{\Pi}_n)).$$

Note that the lower bound idea of Theorem 5.3.2.3 (Exercise 5.3.4.3) essentially uses nondeterminism and cannot be straightforwardly used to get lower bounds in the deterministic case.

5.3.5 Research Problems

Problem 5.3.5.1 *Can Las Vegas (Monte Carlo) communication complexity be used to get a lower bound on the time complexity of probabilistic Turing machines?*

Problem 5.3.5.2 ** *Prove a lower bound* $\Omega(n^2 f(n))$ *for an unbounded function* f *on the time complexity of Turing machines accepting a specific language* $L \in \mathrm{NP}$ *(*NP *is the class of languages accepted by nondeterministic polynomial time Turing machines).*

Problem 5.3.5.3 ** *Prove a superlinear lower bound on the time complexity of a general sequential machine model (multitape Turing machines, register machines) for the recognition of a specific languages* $L \in \mathrm{NP}$.

5.4 Decision Trees and Branching Programs

5.4.1 Introduction

Decision trees and branching programs are standard non-uniform computing models used to measure the time and space complexity of general sequential computations. In this section we show that communication complexity can be used to get lower bounds on the depth of these computing models and on the size of some restricted branching programs. This is of interest because the depth of decision trees (branching programs) corresponds to the sequential time complexity and the size of branching programs corresponds to the sequential space complexity.

This section is organized as follows. Section 5.4.2 is devoted to the definitions of decision trees, branching programs, and their complexity measures. Section 5.4.3 relates communication complexity to the size of some models of branching programs. Section 5.4.4 relates communication complexity to the depth of decision trees. As usual, the last two sections are devoted to exercises and research problems.

5.4.2 Definitions

We start with the definition of decision trees and their complexities.

Definition 5.4.2.1 *Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A* **decision tree T over X** *is a labeled, directed, rooted, binary tree with the following properties:*

(i) the edges are directed from the root to the leaves,

(ii) every internal node has outdegree two,

(iii) every internal node v is labeled by a variable label$(v) \in X$, and one of the edges outcoming from v is labeled by 1 and the other one by 0,

(iv) every leaf is labeled by 0 or 1.

The **Boolean function f_T: $\{0,1\}^n \to \{0,1\}$ computed by the decision tree T** *is defined such that, for every input $\alpha = \alpha_1\alpha_2 \ldots \alpha_n \in \{0,1\}^n$, T computes the function value $f_T(\alpha)$ in the following way. The computation starts at the root of T. If the computation has reached an internal node v, then the computation proceeds via the edge labeled by α_i if label$(v) = x_i$. Once the computation reaches a leaf, the computation ends and $f_T(\alpha)$ is defined to be the label of that leaf. The* **depth of the decision tree T, depth(T)**, *is the length of the longest path from the root to a leaf. The* **size of T, size(T)**, *is the number of nodes of T.*

We observe that decision trees can simulate the computations of a very general model of sequential machines on the inputs of a fixed length in the following sense. A node v of a decision tree corresponds to an internal configuration of the sequential machine and label(v) says which input variable is read in this configuration. Note that several different nodes of the decision tree may correspond to the same configuration of the sequential machine. So, one step of the sequential machine corresponds to the move from a node of the decision tree to one of its sons. We call attention to the fact that decision trees can simulate even more general sequential machines than the sequential machines considered in Section 5.3. The machines of Section 5.3 have input tapes and after reading the value x_i they must read one of the values of x_{i-1}, x_i, or x_{i+1}. But the decision trees may read the input in an arbitrary order, which corresponds to sequential machines that may jump on the input tape (or with a random access to the input tape).

In what follows we define generalized decision trees that can evaluate a function defined on the input variables instead of reading a value of one input variable. This may correspond to very powerful sequential computations, where in one step a function over several input values may be evaluated.

Definition 5.4.2.2 *Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables, and let $\mathcal{F} \subseteq \{f \mid f : \{0,1\}^n \to \{0,1\}\}$. A* **general decision tree over X and $\mathcal{F}$** *is a labeled, directed, rooted binary tree with the following properties:*

(i) the edges are directed from the root to the leaves,

(ii) every internal node has outdegree two,

(iii) every internal node is labeled by a function label(v) $\in \mathcal{F}$ and one of the edges outcoming from v is labeled by 1 and the other one by 0,

(iv) every leaf is labeled by 0 or 1.

The **Boolean function f_T: $\{0,1\}^n \to \{0,1\}$ computed by the general decision tree T** *is defined such that for every input $\alpha = \alpha_1\alpha_2 \ldots \alpha_n \in \{0,1\}^n$, T computes the function value $f_T(\alpha)$ in the following way. The computation starts at the root of T. If the computation has reached an internal node v and label(v) $= g$ then the computation proceeds via the edge labeled by $g(\alpha_1, \alpha_2, \ldots, \alpha_n)$. Once the computation reaches a leaf, the computation ends and $f_T(\alpha)$ is defined to be the label of that leaf. The* **depth of the general decision tree T, depth(T)**, *is the length of the longest path from the root to a leaf.*

Now, we define the branching programs. Informally, one can obtain a branching program B from a decision tree T by joining the nodes of T with the same meaning (corresponding to the same configuration of sequential computations) into one node. Thus, B is an acyclic graph if T does not contain any cyclic sequential computation. Since the number of nodes of the branching program B is exactly the number of configurations used in all sequential computations on inputs of the given length, branching programs are used to measure the space complexity of sequential computations.

Definition 5.4.2.3 *Let $G = (V, E)$ be an acyclic, directed graph. A* **source of G** *is any node from V having indegree 0. A* **sink of G** *is any node from V having outdegree 0.*

Definition 5.4.2.4 *Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of Boolean variables. A* **branching program B over X** *is a labeled, directed, acyclic graph with the following properties:*

(i) B has exactly one source,

(ii) every non-sink (internal) node of B has outdegree two,

(iii) *every non-sink node v is labeled by a variable label$(v) \in X$, and one of the edges outcoming from v is labeled by 1 and the other one by 0,*

(iv) *every sink node is labeled by 0 or 1.*

The **Boolean function f_B: $\{0,1\}^n \to \{0,1\}$ computed by the branching program B** *is defined such that, for every input $\alpha = \alpha_1\alpha_2\ldots\alpha_n \in \{0,1\}^n$, the computation starts at the only source of B. If the computation has reached an internal node v, then the computation of B on α proceeds via the edge labeled by α_i if $x_i = label(v)$. Once the computation reaches a sink of B, the computation ends and $f_B(\alpha)$ is defined to be the label of that sink. The* **depth of the branching program B, depth(B)**, *is the length of the longest directed path in B. The* **size of B, size(B)**, *is the number of nodes of B. The* **capacity of B is capacity(B)**$= \lceil \log_2(\text{size}(B)) \rceil$.
For any Boolean function f,

$$\textbf{size}(f) = \min\{\text{size}(B) \mid B \text{ is a branching program computing } f\},$$

and

$$\textbf{capacity}(f) = \min\{\text{capacity}(B) \mid B \text{ is a branching program computing } f\}.$$

Observation 5.4.2.5 *Let L be a language over the alphabet $\{0,1\}$. For every sequential machine A accepting L*

$$S_A(n) \geq \text{capacity}(h_n(L)) - \log_2 n.$$

Proof. Every $S(n)$ space-bounded sequential machine uses at most $2^{S(n)}$ internal configurations in the computations on the inputs of the length n. Thus, it can be simulated by a branching program of $n \cdot 2^{S(n)}$ nodes, where n indicates the position of the head on the input tape. □

In what follows we shall consider several restricted versions of branching programs corresponding to distinct sequential computing models.

Definition 5.4.2.6 *Let B be a branching program over a set of variables X. The* **i-th level of B**, *denoted* **level$_i(B)$**, *is the set of all nodes of B with distance i from the root of B (The distance of a node v from the root x of B is the length of the shortest directed path from x to v). Let B have $k+1$ levels. B is called* **leveled**, *if every edge of B goes from level$_i(B)$ to level$_{i+1}(B)$ for some $0 \leq i \leq k-1$. If, for every $m \in \{0,1,\ldots,k\}$, all vertices of level$_m(B)$ are labeled by the same input variable, B is called* **oblivious**. *The* **width of the branching program B**, *denoted* **width(B)**, *is the maximum of the cardinalities of the levels of B.*

Oblivious branching programs simulate sequential computations for which a fixed order of reading the input variables is given. Examples are on-line machines, where the variables are read in the order $x_1, x_2, \ldots$. In the literature, any computing model is called **oblivious** if the order of input values read is the same for each input of given length (the order does not depend on the actual values read). Examples for the oblivious computing model are VLSI circuits.

Definition 5.4.2.7 *Let B be a branching program over a set of variables X. For any $k \in \mathbb{N}$, the branching program B is said to be* ***k*-time-only** *if for each variable $x \in X$ and each directed path P of B there are at most k nodes of P labeled by the variable x.*

Observation 5.4.2.8 *Let $k \in \mathbb{N}$ and $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. For any k-time-only branching program over B*

$$\text{depth}(B) \leq k \cdot n.$$

We note that 1-time-only branching programs correspond to erasing machines. Erasing machines have one read-only tape on which they may jump. If the content of a position of the tape has been read, then it is erased and this position cannot be read any more.

5.4.3 Capacity of Branching Programs

One of the central problems in complexity theory is to prove that a specific language from NP is not in DLOGSPACE, i.e., that the language cannot be accepted by any off-line multitape Turing machine A with $S_A(n) = O(\log_2 n)$. One possibility to reach such a result is to prove a superpolynomial lower bound on the size of branching programs computing $h_n(L)$'s for such a language L. Recently, we do not know any lower bound of this kind. The highest known lower bound on the size of branching programs grows slower than n^2. This situation changes if one considers one-time-only branching programs for which exponential lower bounds are known. In what follows we show that one-way communication complexity can help to obtain exponential lower bounds on the size of oblivious one-time-only branching programs.

Theorem 5.4.3.1 *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function of n variables, $n \in \mathbb{N}$. For every oblivious 1-time-only branching program B computing f*

$$\text{width}(B) \geq 2^{\text{cc}_1(f)-1}.$$

Proof. Since B is 1-time-only one can consider that B consists of at most n levels $\text{level}_0(B), \text{level}_1(B), \ldots, \text{level}_{n-1}(B)$. Since B is oblivious we may assume, for every $m \in \{0, \ldots, n-1\}$, that all nodes of $\text{level}_m(B)$ are labeled by the same

variable x_{i_m}. Now, one can construct a one-way protocol $D = \langle \Pi, \Phi \rangle$ computing f in the following way. $\Pi = (\{x_{i_0}, x_{i_1}, \ldots, x_{i_{\lceil n/2 \rceil - 1}}\}, \{x_{i_{\lceil n/2 \rceil}}, \ldots, x_{i_{n-1}}\})$ and, for any input, C_I submits the code of the node of $\text{level}_{\lceil n/2 \rceil - 1}(B)$ in which the branching program is after reading the values of the input corresponding to the variables in Π_L. So, D uses exactly $|\text{level}_{\lceil n/2 \rceil}(B)|$ different messages which can be coded in length $\lceil \log_2 |\text{level}_{\lceil n/2 \rceil - 1}(B)| \rceil$ in a prefix-free manner. □

Corollary 5.4.3.2 *Let f be a Boolean function. For every oblivious 1-time-only branching program B computing f*

$$\text{capacity}(B) \geq \text{cc}_1(f).$$

Thus, Theorem 5.4.3.1 provides many exponential lower bounds on the size of 1-time-only oblivious branching programs computing specific Boolean functions in a straightforward way. More elaborate techniques are used to prove exponential lower bounds on the size of (non-oblivious) 1-time-only branching programs. We do not see any possibility of using communication complexity to get these lower bounds.

Interestingly we do not know any exponential lower bound on the size of 2-times-only branching programs. But exponential lower bounds have been achieved for oblivious k-times-only branching programs. Using a similar approach as for k-multilective VLSI circuits we show that such results can be achieved by using the communication complexity approach, too.

Theorem 5.4.3.3 *Let k and m be positive integers, $k < \frac{1}{2} \cdot \log_2 m - 2$. Let $f \in B_2^m$. For every oblivious k-times-only branching program B computing f*

$$\text{capacity}(B) \geq \text{occ}_{2k}(f)/2k.$$

Proof. Let $\bar{X} = \{x_1, \ldots, x_m\}$ be the set of input variables of f, and let U_0, V_0 be subsets of $\bar{X}$ such that $\text{occ}_{2k}(f, U_0, V_0) = \text{occ}_{2k}(f)$. Note that $|U_0| = |V_0| \geq m/8$. Let B be an oblivious k-time-only branching program computing f. It is sufficient to show that $\text{capacity}(B) \geq \text{occ}_{2k}(f, U_0, V_0)/2k$.

Let $|U_0| = |V_0| = n$. We set $X = U_0 \cup V_0$ and $W_i = \{\text{label}(v) \in X \mid v \in \text{level}_i(B)\}$ for $i = 0, 1, \ldots, \text{depth}(B)$. Since B is oblivious, $|W_i| \leq 1$ for every $i \in \{0, 1, \ldots, \text{depth}(B)\}$. We observe that $k \leq \frac{1}{2} \log_2 n$ because $m \geq n/8$ and $k < \frac{1}{2} \log_2 m - 2$. Since B is k-times-only, $X, W_0, W_1, \ldots, W_{depth(B)}, k, n$, and the cardinalities of W_i's fulfill the assumption of Lemma 4.5.3.1. Thus, there exist sets $U \subseteq U_0$, $V \subseteq V_0$, and integers $b \in \mathbb{N}$, $t_0 = -1$, $t_1, \ldots, t_b \in \{1, 2, \ldots, depth(B)\}$ satisfying the conditions (i), (ii), (iii), (iv), and (v) of Lemma 4.5.3.1. Using this we describe a b-round protocol $D = \langle \Pi, \Phi \rangle$ computing f. We set $\Pi = (\Pi_L, \Pi_R)$ for $\Pi_L = \bar{X} - V$ and $\Pi_R = \bar{X} - U$. Φ can be described as follows. Without loss of generality we assume $U \cap (\bigcup_{j=t_0+1}^{t_1} W_j) \neq \emptyset$. Then in the first round the left computer C_I sends a message coding the node $v \in \text{level}_{t_1}(B)$

reached by B working on the given input. Generally, for any $i \in \{1, \ldots, b-1\}$, if $U \cap (\bigcup_{j=t_i+1}^{t_{i+1}} W_j) \neq \emptyset$, then C_I sends a message coding the node of the $\text{level}_{t_{i+1}}(B)$ reached by B computing on the given input. If $V \cap (\bigcup_{j=t_i+1}^{t_{i+1}} W_j) \neq \emptyset$, then C_{II} sends a message coding the node of the $\text{level}_{t_{i+1}}(B)$ reached by the computation of B on the given input. We observe that if C_I (C_{II}) knows the node reached in the t_i-th level of B, and $\bigcup_{j=t_i+1}^{t_{i+1}} W_j \subseteq \Pi_L = \bar{X} - V$ ($\bigcup_{j=t_i+1}^{t_{i+1}} W_j \subseteq \Pi_R = \bar{X} - U$), then C_I (C_{II}) can determine the node reached in the t_{i+1}-th level in the computation of B on the given input.

Since $b \leq 2k$ according to (ii) of Lemma 4.5.3.1, D is a $2k$-round protocol. If one codes every non-sink node of B by another message, then D can be constructed in such a way that every message has the length $\lceil \log_2(\text{size}(B)) \rceil = \text{capacity}(B)$. Since D is a $2k$-round protocol, $\text{cc}(D) \leq 2k \cdot \text{capacity}(B)$. □

As we have mentioned in Section 4.5 we know languages with linear $\text{occ}_{2k}(f)$. Due to Theorem 5.4.3.3 we see that each such language requires oblivious k-times-only branching programs of exponential size. In the next theorem we strengthen this result further by removing the k-times-only property. More precisely, we exchange this property for a restriction on the depth of the branching program.

Theorem 5.4.3.4 *Let k and m be positive integers, $k < \frac{1}{4}\log_2 m - 1$. Let $f \in B_2^m$ and $\bar{X}$ be the set of input variables of f. Let B be an oblivious branching program computing f. If $\text{depth}(B) \leq k \cdot m$, then*

$$4k \cdot \text{capacity}(B) \geq \min\{\text{occ}_{4k}(f, U_0, V_0) \mid U_0 \subseteq \bar{X},\ V_0 \subseteq \bar{X},\ |U_0| = |V_0| = |\bar{X}|/4,\ U_0 \cap V_0 = \emptyset\}.$$

Proof. Since the depth of B is bounded by $k \cdot m$, there exists a set $X \subseteq \bar{X}$, $|X| \geq |\bar{X}|/2 = m/2$, such that each $x \in X$ is read at most $2k$ times in B. Taking any balanced partition (U_0, V_0) of X one can complete the proof in the same way as in the proof of Theorem 5.4.3.3. □

Corollary 5.4.3.5 *Let M be an oblivious sequential machine accepting a language L. Let $k(n) : \mathbb{N} \to \mathbb{N}$ be a function with $k(n) \leq \frac{1}{4}\log_2 n - 1$ for any $n \in \mathbb{N}$. Let $\text{occ}_{4k(n)}(h_n(L), U_0, V_0) = \Omega(n)$ for every U_0, V_0 with $U_0 \cap V_0 = \emptyset$ and $|U_0| = |V_0| = n/4$. If $\text{T}_M(n) \leq k(n) \cdot n$, then*

$$\text{S}_M(n) = \Omega(n/k(n)).$$

The last application of the combinatorial Lemma 4.5.3.1 presented here enables to exchange obliviousness from Theorem 5.4.3.4 for a restriction on the width of branching programs.

Theorem 5.4.3.6 *Let k, m, and d be positive integers, $k < \frac{1}{2}\log_2 m - 2$, $d \leq m/8 \cdot 3^{2k}$. Let $f \in B_2^m$. For every k-times only levelled branching program*

B computing f with width $(B) \leq d$,

$$\text{capacity}(B) \geq \text{occ}_{2k}(f)/2k.$$

Proof. The proof is almost the same as the proof of Theorem 5.4.3.6. The only one difference is that the cardinality of $W_j = \{label(v) \in X \mid v \in level_j(B)\}$ is not 1 but restricted by d. This does not change anything on the fact that all assumption of Lemma 4.5.3.1 are fulfilled.

□

We observe that, for small constants k, Theorem 5.4.3.6 provides exponential lower bounds on the size of k-times-only branching programs with some linear restriction on their width.

5.4.4 Lower Bounds on the Depth of Decision Trees

It is obvious that for any Boolean function $f \in B_2^n$, $n \in \mathbb{N}$, one can construct a decision tree T (branching program) computing f with $\text{depth}(T) \leq n$. Thus, one cannot use decision trees to try to prove superlinear lower bounds on sequential computations. It seems to be much more interesting to prove lower bounds on the depth of general decision trees than on the basic decision tree model because, for general decision trees, sublinear lower bounds may be of interest, too. In what follows we show how communication complexity can be used to get lower bounds on the depth of decision trees.

Theorem 5.4.4.1 *Let* $X = \{x_1, \ldots, x_n\}$ *be a set of Boolean variables, and let* Bas $\subseteq B_2^n$. *Let* T *be a general decision tree over* X *and* Bas *computing a function* $f \in B_2^n$. *Then*

$$\text{depth}(T) \geq \frac{\max\{\text{cc}(f, \Pi) \mid \Pi \in \text{Abal}(X)\}}{\text{ucc}(\text{Bas}, n)}.$$

Proof. Recall that $\text{ucc}(g) = \max\{\text{cc}(g, \Pi) \mid \Pi \text{ is a partition of } X\}$ (Definition 3.5.2.1) and that $\text{ucc}(\text{Bas}, n) = \max\{\text{ucc}(g) \mid g \in \text{Bas}\} + 1$ (Definition 3.5.2.5). A protocol D simulating the computation of T on a given input α can be constructed in the following way. Let the root be labeled by a function $g \in$ Bas. Then D computes $g(\alpha)$ within communication complexity $\text{ucc}(g)$ and uses one additional communication bit to secure that both C_I and C_{II} know the value $g(\alpha)$. Thus, both C_I and C_{II} know the next node (one of the sons of the root) of the computation of T on α. In every non-leaf node D proceeds in the same way as in the root. The result $f(\alpha)$ is the label of the leaf reached in this way. The communication complexity of D is at most $\text{depth}(T) \cdot \text{ucc}(\text{Bas}, n)$. □

Thus, for instance, for the class of functions *Threshold* we have proved in Section 3.5.2 $\text{ucc}(Threshold, n) \leq \lceil \log_2 n \rceil + 1$. This yields the lower bound

$\max\{cc(f, \Pi) \mid \Pi \in \text{Abal}(X)\}/(\lceil \log_2 n \rceil + 1)$ on the depth of any general decision tree over *Threshold* computing f.

5.4.5 Exercises

Exercise 5.4.5.1 *Prove that for every $f \in B_2^n$, $n \in \mathbb{N}$, there is a decision tree (branching program)T computing f within depth n.*

Exercise 5.4.5.2 *Prove that for every branching program there exists an equivalent branching program having exactly two sinks.*

Exercise 5.4.5.3 *Prove that almost all Boolean functions of n variables require the depth n to be computed on decision trees (branching programs).*

Exercise 5.4.5.4 *Let B be a branching program computing a Boolean function f. Prove that $\text{CC}(f) \leq 3 \cdot \text{size}(B)$.*

Exercise 5.4.5.5 *Let B be a branching program computing a Boolean function f. Prove that $\text{D}(f) \leq 2 \cdot \text{depth}(B)$.*

Exercise 5.4.5.6 *Prove that for every $f \in B_2^n$, there is a branching program of size $O(2^n/n)$ computing f.*

Exercise 5.4.5.7 *Prove that almost all functions from B_2^n have their optimal branching programs of size at least $2^n/3n$.*

Exercise 5.4.5.8 *Prove that every Boolean function can be computed by a 1-time-only branching program.*

Exercise 5.4.5.9 *Prove that every Boolean function can be computed by a branching program of width 2.*

Exercise 5.4.5.10 *Find a Boolean function $f \in B_2^n$ having the following two properties:*

(i) f essentially depends on all its input variables, and

(ii) there exists a branching program of depth $\lceil \log_2 n \rceil + 1$ computing f.

Exercise 5.4.5.11 ** *Consider the threshold function $T_k^n \in B_2^n$ defined by $T_k^n(\alpha_1, \ldots, \alpha_n) = 1$ iff $\alpha_1 + \alpha_2 + \ldots \alpha_n \geq k$. Prove that there exists a k such that every branching program computing T_k^n has size $\Omega(n \cdot (\log \log_n)/\log \log \log_n)$.*

Exercise 5.4.5.12 *Prove the following claim. Let f be a symmetric Boolean function computed by a 1-time-only branching program B_1. Then there exists an oblivious 1-time-only branching program B_2 such that*

(i) *B_2 computes f, and*

(ii) $\text{size}(B_2) \leq \text{size}(B_1)$.

Exercise 5.4.5.13 * *Prove, for a specific language L, that every 1-time-only branching program computing $h_n(L)$ has size $2^{\Omega(n)}$.*

Exercise 5.4.5.14 * *Prove the following claim. Let $L \subseteq \{0,1\}^*$. There is for some constant k a sequence of branching programs computing $h_n(L)$ with polynomial size and width k if and only if there is a sequence of Boolean circuits computing $h_n(L)$ with polynomial size and depth $O(\log_2 n)$.*

Exercise 5.4.5.15 *Define nondeterministic branching programs and relate their complexity measures to nondeterministic communication complexity.*

5.4.6 Research Problems

Problem 5.4.6.1 * *Prove an exponential lower bound on the size of 2-times-only branching programs computing a specific Boolean function.*

Problem 5.4.6.2 * *Prove an exponential lower bound on the size of oblivious branching programs computing a specific Boolean function with depth $n \cdot f(n)$, where $\log_2 n = o(f(n))$.*

Problem 5.4.6.3 ** *Prove a quadratic (or even an exponential) lower bound on the size of branching programs computing a specific Boolean function.*

Problem 5.4.6.4 ** *Prove or disprove:*

(i) *deterministic logarithmic space is a proper subset of nondeterministic logarithmic space,*

(ii) *deterministic logarithmic space is a proper subset of deterministic polynomial time.*

5.5 Bibliographical Remarks

The simple relation between finite automata and communication complexity formulated in Theorem 5.2.3.1 comes from Hromkovič [Hr86c]. The one-way

uniform protocols have been introduced by Hromkovič and Schnitger [HrS96], where Theorem 5.2.4.8 has been proved too. The introduction of nondeterministic uniform protocols and their relation to nondeterministic finite automata is a contribution of this book. The claim of Exercise 5.2.5.15 showing that the nondeterministic message complexity of a regular language L can essentially differ from the size of the minimal nondeterministic finite automaton for L has been recently established by Klauck and Schnitger[KlS96].

Time complexity as a complexity measure of off-line multitape Turing machines was introduced by Hartmanis and Stearns in [HS65]. One of the most challenging problems of complexity theory is to prove a superlinear lower bound on sequential time complexity. Unfortunately, we do not have any such result for powerful models of sequential computations like multitape Turing machines or register machines. The strongest Turing machine model for which superlinear lower bounds have been established is the one-tape Turing machine with an additional read-only two-way input tape (see Maas, Schnitger, Szemeredi and Turán [MSS87, MSST90]). The crossing sequence argument for proving lower bounds was introduced by Cobham [Co66], and used in many papers about lower bounds (see, for instance, some more advanced applications in Ďuriš and Galil [DG84], Ďuriš, Galil, Paul, and Reischuk [DGPR83], Hromkovič [Hr83, Hr85a, Hr85b, Hr89, Hr91b], Janiga [Ja81], and Rivest and Yao [RY78]). The quadratic lower bounds on the time complexity of classical nondeterministic Turing machines were presented by Freivalds [Fr84], Maas [Ma84], and Ming Li [Li84]. The lower bound method based on communication complexity and presented in Section 5.3.2 was observed by Kalyanasundaram [Ka88], Dietzfelbinger [Di93], and perhaps others too. Section 5.3.3 contains a straightforward extension of this idea. The communication complexity argument was further developed by Kalyanasundaram and Schnitger [KS92], and Paturi and Simon [PaS83] to get lower bounds for probabilistic Turing machines. Recently, Dietzfelbinger [Di96] has proved the assertion of Exercise 5.3.4.8 extending his idea from [Di93] about the use of deterministic communication complexity to get lower bounds on time complexity of (deterministic) Turing machines.

Decision trees are a classical non-uniform model for the measurement of the time complexity of non-uniform programs using the operations "if then else" and "go to" (see the surveys by Bollobás [Bo84] and Kahn, Saks, and Sturtevant [KSS84]). The main interest in the study of decision trees was not in the standard model testing the values of Boolean variables, but in different generalized models testing the values of functions over integers, rationals, and reals (see, e.g., Dobkin and Lipton [DL78, DL79], Dietzfelbinger and Maas [DM88], Klein and Meyer auf der Heide [KM83], Meyer auf der Heide [MadH85], Snir [Sn82, Sn85], Yao [Ya77], and Yao and Rivest [YR80]).

Branching programs may seem to be more interesting than decision trees because they simultaneously measure time and space of non-uniform sequential computations. This relation between branching programs and sequential computations was proved by Cobham [Co66] and Pudlák and Žák [PZ83]. Moreover, Wegener [We84] showed a closed relation between branching programs

and Boolean circuits (see Exercises 5.4.5.4 and 5.4.5.5). Branching program are also called binary decision diagrams and used as data structures for representing Boolean functions. They are used for symbolic Boolean manipulation, which has been successfuly applied to a wide variety of practical tasks, particulary in computer-aided VLSI design. A nice survey of these applications can be found in [Mi96]. We also call attention to the fact that branching programs can be viewed as a restricted version of the contact schemes (Π-schemes) intensively investigated already in the Russian-language literature for three decades. An exhaustive survey of the study of this computing model can be found in the nice monograph by Nigmatulin [Ni83].

Rivest and Vuillemin [RV76] proved that almost all Boolean functions of n variables require the maximal depth n to be computed by branching programs.

The highest lower bounds $\Omega(n^2/(\log_2 n)^2)$ on the size of branching programs computing concrete Boolean functions can be achieved by applying the method of Nechiporuk [Ne66]. The highest lower bound on the size of branching programs computing a symmetric Boolean function (Exercise 5.4.5.10) was proved by Pudlák [Pu84].

1-time-only branching programs were introduced by Masek [M76]. Wegener [We84] proved that there is no difference between the size of 1-time-only branching programs and the size of oblivious 1-time-only branching programs for symmetric Boolean functions. Exponential lower bounds on the size of 1-time-only branching programs computing concrete sequences of Boolean functions have been developed by Ajtai, Babai, Hajnal, Komlós, Pudlák, Szemeredi, and Turán [ABH86], Dunne [Du85], Kriegel and Waack [KW86], Wegener [We84a, We86], and Žák [Za84, Za86, Za95]. One of the main open problems connected with branching programs is to prove high lower bounds at least on the size of 2-times-only branching programs. It is interesting that we do not have any lower bound on this 2-times-only restricted model of branching programs higher than the lower bounds on the size of the general model of branching programs [We87]. More information about the study of branching programs can be found in the excellent monograph by Wegener [We87].

The relation between communication complexity and k-times-only branching programs presented in Section 5.4.5 has been developed in this book. It is a simple extension of the ideas described in the papers by Hromkovič and Procházka [HrP88], Hromkovič [Hr91a], and Hromkovič, Krause, Meinel, and Waack [HKMW92] relating the size of branching programs to the area of VLSI circuits. The results presented here can be simply extended to the nondeterministic case (for the definition of nondeterministic branching programs see Meinel [Me87, Me88]).

As we have already noted, Section 5 does not provide a complete survey of the relation between the communication protocol model and the complexity of sequential computing. From the research direction not involved in Section 5 we call attention to the following three applications of communication complexity.

The study of the connections between randomness and computation is an important theme in contemporary theoretical computer science. This is because

randomized algorithms using a source of independent unbiased bits can be used to speed up the computations of important computing problems like optimization problems, data encryption, etc. Yet whether or not a source of independent unbiased bit can be constructed is an open problem. The major difficulty is that making any measurement disturbs the systems in such a way that the bit-sequence resulting from series of measurements might be correlated. Santha and Vazirani [SV86] defined **semi-random sources** and introduced a general mathematical framework for the study of imperfect and correlated sources of randomness coresponding to physical sources like Zener diodes and Geiger counters. Vazirani [Va87] has used the communication protocol model to efficiently generate high quality random sequences (called **quasi-random bit sequences**) from two independent semi-random sources. Vazirani's new concept is strongly related to communication complexity theory. It led to a definition of **strong** communication complexity of a Boolean function and brought new impulses in the investigation of communication complexity.

The second application we present is a relation between cryptography and the communication complexity of the following number-theoretic problem. Let the first computer C_I have a prime number x and the second computer C_{II} have a composite number y, where $x, y < 2^n$. The protocol has to compute a prime number p, $p < 2n$, such that $x \not\equiv y (\mathrm{mod})p$ (The existence of such a small prime p is guaranted by the prime number theorem and the Chinese remainder theorem). We require that after the communication of the protocol both C_I and C_{II} know p. Let $c(n)$ denote the communication complexity of this problem. Obviously $c(n) \leq n + \lceil \log_2 n \rceil$ because the trivial protocol can work as follows. C_I submits its whole input (n bits) to C_{II}, which computes p and sends it ($\log_2 n$ bits) back to C_I. The trivial lower bound is $c(n) = \Omega(\log_2 n)$. At present, these trivial upper and lower bounds are the best known. What is important is that this communication problem encodes the computational difficulty of primality testing. Answering it is important for computational number theory (cryptography, coding) as well as for complexity theory from the following reason. If $c(n) = O(\log_2 n)$, then for every n there is a polynomial size (in n) Boolean formula deciding whether the n input bits code a prime integer. This would mean that primality can be tested highly efficiently, both sequentially and in parallel. If $c(n) \neq O(\log_2 n)$, the primality function has no such formulas, and it would be the first explicit example of such a function. For more details about this problem one can consult Wigderson [Wi91].

The last application we would like to mention is the problem of whether the real time storage modification machine of Shönhage [Sho80] is more powerful than the Kolmogorov–Uspenskii machine introduced by Kolmogorov and Uspenskii [KU63]. Both machines are powerful models of sequential computations with dynamically changing storage structures (for instance, Shoenhage's machine can compute integer multiplication in linear time). Kalyanasundaram and Schnitger have reduced this known open problem to a special communication problem on our two-party communication protocols in [KS92].

These last examples of the application of communication complexity show that several open problems in complexity theory may be elucidated in terms of communication (information transfer). The elegance of the communication protocol model, and the existence of non-trivial machinery to handle it, give hope that the approach based on communication complexity will be one of the instrumental approaches for solving several of these open problems. Note that this book does not handle the multiparty protocol model, which has several interesting applications too (see, for instance, Babai, Nisan, and Szegedy [BNS89] for the relation to time-space tradeoffs and branching programs).

References

[AA80] Abelson, H. Andreae, P., Information transfer and area-time tradeoffs for VLSI multiplication. *Comm. ACM* 23 (1980), 20-23.

[AB87] Alon, A., Boppana, R.; The monotone circuit complexity of Boolean functions. *Combinatorica* 7 (1987), 1-22.

[ABH86] Ajtai, M. Babai, L., Hajnal, A., Komlós, Pudlák, P., Szemeredi, A. Turán, G., Two lower bounds for branching programs. *Proc. 18th ACM STOC*, ACM 1986, pp. 30-38.

[AFR85] Alon, N., Frankl, P., Rödl, V., Geometrical realization of set systems and probabilistic communication complexity. *Proc. 26th IEEE FOCS*, IEEE 1985, pp. 277-280.

[AG87] Ajtai, M., Gurevich, Y., Monotone versus positive. *J. of Association for Computing Machinery* 34 (1987), 1004-1015.

[AM85] Alon, N., Milman, V.D., λ_1, Isoperimetric inequalities for graphs, and superconcentrators. *J. Combinatorial Theory B* 38 (1985), 73-88.

[AUY83] Aho, A.V., Ullman, J.D., Yannakakis, M., On notions of informations transfer in VLSI circuits. *Proc. 15th ACM STOC*, ACM 1983, pp. 133-139.

[Ab78] Abelson, H., Lower bounds on information transfer in distributed computations, *Proc. 19th IEEE FOCS*, IEEE 1978, pp. 151-158.

[Ab78a] Abelson, H., Towards a theory of local and global in computation. *Theoretical Computer Science* 6 (1978), 41-67.

[Ab80] Abelson, H., Lower bounds on information transfer in distributed computations. *J. of Association for Computing Machinery* 27 (1980), 384-392.

[Abl93] Ablayer, F., Lower bounds for one-way probabilistic communication complexity. *Proc. ICALP '93, Lecture Notes in Computer Science* 700, Springer-Verlag, Berlin 1993, pp. 241-252.

[Abl96] Ablayev, F., Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theoretical Computer Science* 157 (1996), 139-159.

[Al78] Albrecht, A., On circuits of lattice elements. *Problemy kibernetiky* 33 (1978), 209-214 (in Russian).

[Al86] Alon, N., Eigenvalues and expanders. *Combinatorica* 6 (1986), 85-95.

[All89] Allender, E., A note on the power of threshold circuits. *Proc. 30th IEEE FOCS*, IEEE 1989, pp. 580-584.

[Am96] Ambainis, A., Communication complexity in a 3-computer model. *Algorithmica* 16 (1996), 298-301.

[An85] Andreev, A.E., On a method for obtaining lower bounds on the complexity of individual monotone functions. *Dokl. Akad. Nauk SSSR* 282 (1985), 1033-1037 (in Russian) [English translation in: *Sov. Math. Dokl. 31 (1985), 530-534*].

[An87] Andreev, A.E., On a method for proving higher than quadratic effective lower bounds on the complexity of π-schemes. *Vest. MGU* 1 (1987), 70-73 (in Russian).

[BFS86a] Babai, L., Frankl, P., Simon, J., Complexity classes in communication complexity theory. *Proc. 27th IEEE FOCS*, IEEE 1986, pp. 337-347.

[BFS86b] Babai, L., Frankl, P., Simon, J., BPP and the polynomial time hierarchy in communication complexity. Technical report TR-86-03, Dept. of Computer Science, University of Chicago 1986.

[BG80] Brent, R.P., Goldschlager, L.M., Some area-time tradeoffs for VLSI. *Proc. 12th ACM STOC*, ACM 1980, pp. 190-200.

[BG82] Brent, R.P., Goldschlager, L.M., Some area-time tradeoffs for VLSI. *SIAM J. Computing* 11 (1982), pp. 737-747.

[BGH82] Borodin, A., von zur Gathen, J., Hopcroft, J., Fast parallel matrix and GCD computations. *Proc. 23rd ACM STOC*, ACM 1982, pp. 65-71.

[BK80] Brent, R.P., Kung, H.T., The chip complexity of binary arithmetic. *Proc. 12th ACM STOC*, ACM 1980, pp. 190-200.

[BK81] Brent, R.P., Kung, H.T., The area-time complexity of binary multiplication. *J. Association for Computing Machinery* 28 (1981), pp. 521-534.

[BNS89] Babai, L., Nisan, N, Szegedy, M., Multiparty protocols and logspace-hard pseudorandom sequences. *Proc. 21st ACM STOC*, ACM 1989, pp. 1-11

[BPP81] Bilardi, G., Pracchi, M., Preparata, F.P., A critique and appraisal of VLSI models of computation. In: *VLSI Systems and Computations* (Kung, H.T., Sproull, R., Steele, G. - eds.) Computer Science Press 1981, pp. 81-88.

[BS90] Boppana, R., Sipser, M., The complexity of finite functions. *Handbook of Theoretical Computer Science A* (J. van Leeuwen, ed.), Elsevier Science Publishers B.V., 1990, pp. 759-804.

[Bl84] Blum, N., A Boolean function requiring 3n network size. *Theoretical Computer Science* 28 (1984), pp. 337-345.

[Bo84] Bollobás, B., . *Extremal Graph Theory,* Academic Press, 1984.

[Br41] Brooks, R. L., On coloring the nodes of network. *Proc. Cambrigde Philos. Soc.* 37 (1941), pp. 194-197.

[Bs89] Bshouty, N.H., On the extended direct sum conjecture. *Proc. 21st ACM STOC*, ACM 1989, pp. 177-185.

[CFL83] Chandra, A.K., Furst, M.L., Lipton, R.J., Multi-party protocols. *Proc. 15th ACM STOC*, ACM 1983, pp. 94-99.

[CG85] Chor, B., Goldreich, O., Unbiased bits from sources of weak randomness and probabilistic communication complexity. *Proc. 26th IEEE FOCS*, IEEE 1985, pp. 429-442.

[CG90] Canetti, R., Goldreich, O., Bounds on tradeoffs between randomness and communication complexity. *Proc. 31st IEEE FOCS*, IEEE 1990, pp. 766-775.

[CM81] Chazelle, B.M., Monier, L.M., Model of computation for VLSI with related complexity results. *Proc. 13th ACM STOC*, ACM 1981, pp. 318-325.

[Ch71] Chrapchenko, V.M., A method for determining lower bounds for the complexity of π-schemes. *Math. Notes Acad. Sci. USSR* (1971), 474-479.

[Ch71a] Chrapchenko, V. M., On the complexity of the realization of linear functions in the class of π-schemes. *Matem. Zametki* 9 (1971), 35-40 (in Russian).

[Ch71b] Chrapchenko, V. M., About a method for obtaining lower bounds on the complexity of π-schemes. *Matem. Zametki* 10 (1971), 83-92 (in Russian).

[Ch84] Chrapchenko, V. M., Lower bounds on the complexity of schemes of functional elements (A survey). *Kiberneticeskij sbornik* 21 (1984), 3-54.

[Co64] Cobham, A., The intrinsic computational difficulty of functions. *Proc. 1964 Congress for Logic, Mathematics, and Philosophy of Science*, North Holland, Amsterdam 1964, pp. 24-30.

[Co66] Cobham, A., The recognition for perfect squares. *Proc. 7th Annual IEEE Symp. on SWAT*, IEEE 1966, pp. 78-87.

[DF92] Dolev, D., Feder, T., Determinism vs. nondeterminism in multiparty communication complexity. *SIAM J. Computing* 21 (1992), pp. 889-893.

[DG84] Ďuriš, P., Galil, Z., A time-space tradeoff for language recognition. *Mathematical Systems Theory* 17(1984), 3-12.

[DG93] Ďuriš, P., Galil, Z., On the power of multiple reads in chip. *Information and Computation* 104 (1993), 277-287.

[DGPR83] Ďuriš, P., Galil, Z., Paul, W., Reischuk, R., Two nonlinear lower bounds. *Proc. 15th ACM STOC*, ACM 1983, pp. 127-132.

[DGS84] Ďuriš, P., Galil, Z., Schnitger, G., Lower bounds on communication complexity. *Proc. 16th ACM STOC*, ACM 1984, pp. 81-91.

[DHS94] Dietzfelbinger, M., Hromkovič,J., Schnitger,G., A comparison of two lower bounds methods for communication complexity. Proc. *19th MFCS '94, Lecture Notes in Computer Science* 841, Springer-Verlag 1994, pp. 423-432. (also *Theoretical Computer Science* 168 (1996), 39-51).

[DHSR97] Ďuriš, P., Hromkovič, J., Schnitger, G., Rolim, Las Vegas versus determinism for one-way communication complexity, finite automata, and polynomial-time computations. *Proc. STACS' 97, Lecture Notes in Computer Science*, Springer-Verlag 1997, to appear.

[DKMW92] Damm, C., Krause, K., Meinel, Ch., Waack, S., Separating counting communication complexity classes. *Forschungsbericht Nr. 92-01*, 1992, Universität Trier.

[DL78] Dobkin, D.P., Lipton, R.J., A lower bound of $n^2/2$ on linear search programs for the knapsack problem. *J. of Computer and System Sciences* 16 (1978), 413-417.

[DL79] Dobkin, D.P., Lipton, R.J., On the complexity of computations under varying sets of primitives. *J. of Computer System Sciences* 18 (1979), 86-91.

[DM88] Dietzfelbinger, M., Maas, W., Lower bounds arguments with "inaccessible" numbers. *J. of Computer System Sciences* 36 (1988), 313-335.

[Di93] Dietzfelbinger, M., personal communication.

[Di96] Dietzfelbinger, M., The linear-array problem in communication complexity resolved. *Forschungsbericht* Nr. 632, University of Dortmund, 1996.

[Du85] Dunne, P. E., Lower bounds on the complexity of 1-time-only branching programs. *Proc. FCT'85, Lecture Notes in Computer Science* 199, Springer-Verlag 1985, pp. 90-99.

[Du88] Dunne, P. E., *The Complexity of Boolean Networks.* Academic Press 1988.

[FF81] Frauld, P., Füredi, Z., A short proof of a theorem of Harper about Hamming spheres. *Discrete Math.* 34 (1981),pp. 311-313.

[FJM90] Fleischer, M., Jung, H., Mehlhorn, K., A time-randomness tradeoffs for communication complexity. *Proc. 4th Int. Workshop on Distr. Alg.*, 1990.

[FKNN91] Feder, T., Kushilevitz, E., Naor, M., Nisan, N., Amortized communication complexity. *Proc. 32nd IEEE FOCS*, IEEE 1991, pp. 299-304 (to appear in *SIAM J. Computing*).

[FMP82] Fischer, M.J., Meyer, A.R., Paterson, M.S., $\Omega(nlogn)$ lower bounds on length of Boolean formulas. *SIAM J. Computing* 11 (1982), 416-427.

[Fr77] Freivalds, R., Probabilistic machines can use less running time. *Information Processing 1977, IFIP*, North Holland 1977, pp. 839-842.

[Fr84] Freivalds, R., Quadratic lower bound for nondeterministic Turing machines. Unpublished communication at the 11th MFCS'84, Prague 1984.

[Fue87] Fürer, M., The power of randomness for communication complexity. *Proc. 19th ACM STOC*, ACM 1987, pp. 178-191.

[GG81] Gaber, Z., Galil, Z., Explicit constructions of linear-sized superconcentrators. *J. of Computer and System Sciences* 22 (1981), 407-420.

[GH92] Goldmann, M., Håstad, J.; A lower bound for monotone clique using a communication game. *Information Processing Letters* 41 (1992), 221 - 226.

[GHR92] Goldmann, M., Håstad, J., Razborov, A., Majority gates vs. general weighted threshold gates. *Proc. 7th Structure in Complexity Theory*, 1992.

[GHW92] Gubáš, X., Hromkovič, J., Waczulík, J., A nonlinear lower bound on the practical combinatorial complexity. *Proc. 9th STACS '92, Lecture Notes in Computer Science* 577, Springer-Verlag 1992, pp. 281-292. (Also *Theoretical Computer Science* 143 (1995), 335-342)

[GMW87] Goldreich, D., Micali, S., Wigderson A., How to play any mental game, *Proc. 19th ACM STOC,* ACM 1987, pp. 218-229.

[GW86] Gubáš, X, Waczulík, J., Closure properties of the families of languages defined by communication complexity. *Švoč 1986/1, section Theoretical Cybernetics and Mathematical Informatics*, Comenius University, Bratislava 1986 (in Slovak) 26p.

[GW87] Gubáš, X., Waczulík, J., Closure properties of the complexity measures A and AT^2, *Švoč 1987/1, section VLSI and computer graphics.* Comenius University, Bratislava 1987 (in Slovak) 25p.

[Ga78] Gashkov, The depth of Boolean Functions. *Problemi Kibernetiki* 34 (1978), 265-268 (in Russian).

[Go96] Golze, U., *VLSI Chip Design with the Hardware Description Language VERILOG.* Springer-Verlag Berlin Heidelberg, 1996.

[Gor73] Gorelik, E.S., On the complexity of the realization of elementary conjunctions and disjunctions in the base $[x/y]$. *Problemi Kibernetiki* 26 (1973), 27-36 (in Russian).

[HG91] Håstad, J., Goldmann, M., On the power of small-depth threshold circuits. *Commputational Complexity* 1 (1991), 113-129.

[HHK91] Hofmeister, T., Houberg, W., Köling, S., Some notes on threshold circuits and multiplication in depth 4. *Information Processing Letters* 39 (1991), 219-225.

[HHS75] Harper, L.H., Hsieh, W.N., Savage, J.E., A class of Boolean functions with linear combinatorial complexity, *Theoretical Computer Science* 1 (1975), pp. 161-183.

[HKMW92] Hromkovič, J., Krause, M., Meinel, Ch., Waack, S., Branching programs provide lower bounds on the area of multilective deterministic and nondeterministic VLSI circuits. *Information and Computation* 95 (1992), 117-128.

[HLR92] Hromkovič,J., Ložkin, S.A., Rybko, A. I., Sapoženko, A. A., Škalikova, A. N., Lower bounds on the area complexity of Boolean circuits. *Theoretical Computer Science* 97 (1992), 285-300.

[HMPST87] Hajnal, A., Maas, W., Pudlák, P., Szegedy, M., Turán, G., Threshold circuits of bounded depth. *Proc. 28th IEEE FOCS*, IEEE 1987, pp. 99-110.

[HMT88] Hajnal, A., Maass, W., Turán, G., On the communication complexity of graph properties. *Proc. 20th ACM STOC*, ACM 1988, pp. 186-191.

[HP88] Hromkovič, J., Pardubská, D., Some complexity aspects of VLSI computations. Part 3. On the power of input bit permutation in the tree and trellis automata. *Computers and Artificial Intelligence* 7 (1988), 397-412.

[HP89] Hromkovič, J., Pardubská, D., Some complexity aspects of VLSI computations. Part 4. VLSI circuits with program. *Computers and Artificial Intelligence* 8 (1989), No. 2, 169-188.

[HR88] Halstenberg, B., Reischuk, R., On different modes of communication. *Proc. 20th ACM STOC*, ACM 1988, pp. 162-172.

[HR92] Halstenberg, B., Reischuk, R., Über den Nutzen von Orakelfragen bei nicht deterministischen Kommunikationsprotokollen.*Festschrift zum 60. Geburtstag von Günter Hotz*, Teubner-Verlag 1992, pp. 169-183 (in German).

[HS65] Hartmanis, I., Stearns, R. E., On the computational complexity of algorithms. . *Trans. AMS* 117, 285-306.

[HSa73] Harper, L. H., Savage, J. E., Complexity made simple. *Proc. Int. Symp. on Combinatorial Theory*, Rome 1973, pp. 2-15.

[HU79] Hopcroft, J.E., Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publ. Comp. Inc., 1979.

[Ha86] Håstad, J., Improved lower bounds for small depth circuits. *Proc. 18th ACM STOC,* ACM 1986, pp. 6-20.

[HoS68] Hodes, L., Specker, E., Lengths of formulas and elimination of quantifiers. In: *Contributions to Mathematical Logic*, North-Holland, Amsterdam 1968, pp. 175-188.

[Hr83] Hromkovič, J., One-way multihead deterministic finite automata. *Acta Informatica.* 19 (1983), 377-384.

[Hr84] Hromkovič, J., Normed protocol and communication complexity. *Computers and Artificial Intelligence* 3 (1984), No. 5, 415-422.

[Hr85] Hromkovič, J., Linear lower bounds on unbounded fan-in Boolean circuits. *Information Processing Letters* 21 (1985), 71-74.

[Hr85a] Hromkovič, J., Fooling a two-way automaton with reversal number restriction. *Acta Informatica* 22 (1985), 589-594.

[Hr85b] Hromkovič, J., On the power of alternation in automata theory. *J. Computer and System Sciences* 31 (1985), 28-39.

[Hr86a] Hromkovič, J., A new approach to defining the communication complexity for VLSI, *Proc. 12th MFCS'86, Lecture Notes in Computer Science* 233, Springer-Verlag, 1986, pp. 431-439.

[Hr86b] Hromkovič, J., Communication complexity hierarchy. *Theoretical Computer Science* 48 (1986), pp. 109-115.

[Hr86c] Hromkovič, J., Relation between Chomsky hierarchy and communication complexity hierarchy. *Acta Math. Univ. Com.*, Vol. 48-49 (1986), 311-317.

[Hr88a] Hromkovič, J., Some complexity aspects of VLSI computations. Part 1. A framework for the study of information transfer in VLSI circuits. *Computers and Artificial Intelligence* 7 (1988), pp. 229-252.

[Hr88b] Hromkovič, J., Some complexity aspects of VLSI computations. Part 2. Topology of circuits and information transfer. *Computers and Artificial Intelligence* 7 (1988), 289-302.

[Hr88c] Hromkovič, J., The advantages of a new approach to defining the communication complexity for VLSI. *Theoretical Compututer Science* 57 (1988), 97-111.

[Hr89] Hromkovič, J., Tradeoffs for language recognition on alternating machines. *Theoretical Computer Science* 63 (1989), 203-221.

[Hr89a] Hromkovič, J., Some complexity aspects of VLSI computations. Part 5. Nondeterministic and probabilistic VLSI circuits. *Computers and Artificial Intelligence* 8 (1989), 162-188.

[Hr89b] Hromkovič, J., Some complexity aspects of VLSI computations. Part 6. Communication complexity, *Computers and Artificial Intelligence* 8 (1989), 209-225.

[Hr91] Hromkovič, J., Nonlinear lower bounds on the number of processors of circuits with sublinear separators. *Infomation and Computation* 95 (1991), 117-128.

[Hr91a] Hromkovič, J., Branching programs provide lower bounds on the area of VLSI circuits. *Kybernetika* 27 (1991), pp. 542-550.

[Hr91b] Hromkovič, J., Reversals - space - parallelism tradeoffs for language recognition. *Mathematica Slovaca* (1991), 121-136.

[Hr92] Hromkovič, J., Topology of parallel networks and computational complexity. *Proc. 18th WG'92, Lecture Notes in Computer Science* 657, Springer Verlag, 1993, pp. 70-77.

[HrP88] Hromkovič, J., Procházka, J., Branching programs as a tool for proving lower bounds on VLSI computations and optimal algorithms for systolic arrays. *Proc. 13th MFCS'88, Lecture Notes in Computer Science* 324, Springer-Verlag 1988, pp. 360-370.

[HrP94] Hromkovič, J., Procházka, J., Lower bounds on systolic array computations and the optimality of Kung's convolution algorithm. *Mathematical Aspects of Natural and Formal Languages* (G. Paun - ed.), *Series of Computer Science*, Vol. 43, World Scientific 1994, pp. 151-164.

[HrS95] Hromkovič, J., Schnitger, G., Determinismus versus Las Vegas. Unpublished manuscript, 1995.

[HrS96] Hromkovič, J., Schnitger, G., Nondeterministic communication with a limited number of advice bits. *Proc. 28th ACM STOC*, ACM 1996, pp. 551-560.

[HrSu90] Hromkovič, J., Šuster, B., On the gap between the complexities of two types of planar Boolean circuits. *Diskretnaja matematika* 2 (1990), pp. 121-126 (in Russian).

[Hue93] Hühne, M., personal communication.

[JPS84] Ja'Ja, J., Prassanna Kamar, V.K, Simon, J., Information transfer under different sets of protocols. *SIAM J. Computing* 13 (1984), 840-849.

[Ja81] Janiga, L., Real-time computations of two-way multihead finite automata. *Proc. FCT '79*, Academic Verlag, Berlin 1979, pp.214-219.

[K79] Kung,H.T., Let's design algorithms for VLSI systems. *Proc. of Caltech Conf. on VLSI: Architecture, Design, Fabrication*, 1979, pp. 65-90.

[KA86] King F. Pang, Abhasel Gamal, Communication complexity of computing the Hamming distance. *SIAM J. Computing* 15 (1986), 932-946.

[KKN92] Karchmer, M., Kushilevitz, E., Nisan, N., Fractional covers and communication complexity. *Proc. 7th IEEE Structure in Complexity Theory*, IEEE 1992, pp. 262-274 (to appear in *SIAM J. Discrete Mathematics*).

[KL78] Kung, H.T., Leiserson, P.E., Algorithms for VLSI processor arrays. *Proc. Symposium on Sparse Matrix Compulations*, Knuxville 1978.

[KL83] Kramer, M., van Leeuwen, J., The VLSI complexity of Boolean functions, *Lecture Notes in Computer Science* 171, Springer-Verlag, Berlin 1983, pp. 397-407.

[KLO96] Kushilevitz, E., Linial, N., Ostrovsky, R., The linear-array conjecure in communication complexity is false. *Proc. 28th ACM STOC*, ACM 1996.

[KM83] Klein, P., Meyer auf der Heide, F., A lower bound for the knapsack problem on random access machines. *Acta Informatica* 19 (1983), 385-395.

[KMa65] Klass, B. M., Malyshev, V. A.A., Bounds on complexity of some classes of functions. *Vestnik Moskov. Univ. and Sev. Mat. Mech.* 4 (1965), 44-51 (in Russian).

[KRW91] Karchmer, M., Raz, R., Wigderson, A., On proving super-logarithmic depth lower bounds via the direct sum in communication complexity. *Proc. 6th IEEE Structures in Complexity Theory*, IEEE 1991, pp. 299-304.

[KS87] Kalyanasundaram, B., Schnitger, G., The probabilistic communication complexity of set intersection. *Proc. 2nd Annual Conference on Structure in Complexity Theory*, 1987, pp. 41-47.

[KS92] Kalyanasundaram, B., Schnitger, G., Communication complexity and lower bounds for sequential machines. In: *Festschrift zum 60.Geburtstag von Günter Hotz*, Teubner Verlag, 1992, pp. 840-849.

[KSS84] Kahn, J., Saks, M. Sturtevant. D., A topological approach to evasiveness. *Combinatorica* 4 (1984), 297-306.

[KU63] Kolmogorov, A. N., Uspenskii, V. A., On the definition of an algorithm. *Russian Math. Surveys* 30 (1963), 217-245.

[KW86] Kriegel, Waack, S., Lower bounds on the complexity of real-time branching program. Tech. Rep., Akademie der Wissenschaften, Berlin 1986.

[KWi88] Karchmer, M., Wigderson, A., Monotone circuits for connectivity require superlogarithmic depth. *Proc. 20th ACM STOC*, ACM 1988, pp. 539-550 (also *SIAM J. Discrete Mathematics* 3 (1990), 718-727).

[KZ81] Kedem, Z.M., Zorat, A., Replication of inputs may save computational resources in VLSI. *VLSI Systems and Computations* (H.T. Kung, B.Sproull, G. Steele, Eds.), Computer Sci. Press Rockville 1981.

[KZ81a] Kedem, Z. M., Zorat, A., On relations between input and communicational computation in VLSI. *Proc. 22nd ACM STOC*, ACM 1981, pp. 37-41.

[Ka88] Kalyanasundaram, B., Lower bounds on time, space and communication. Ph. D. Dissertation, The Pensylvania State University, 1988.

[Kar89] Karchmer, M., Communication complexity. A new approach to circuit depth. MIT Press 1989.

[KlS96] Klauck, H., Schnitger, G., Nondeterministic finite automata versus nondeterministic uniform communication complexity. Unpublished manuscript, University of Frankfurt, 1996.

[Ko65] Komlós, J., On the determinant of (0,1)-matrices. *Studia Sci. Math. Hungar.* 2 (1965), 7-21.

[Ko68] Komlós, J., On the determinant of random matrices. *Studia Sci. Math. Hungar.* 3 (1968), 378-399.

[Kr67] Kravcov, S.S., On the realization of switching functions in a class of circuits of lattice. *Problemy Kibernetiki* 19, (1967), 274-284 (in Russian).

[Kri64] Krichevskij, R.E., On the complexity of parallel contact schemes realizing a sequence of Boolean functions. *Problemy Kibernetiki* 12 (1964), 45-55 (in Russian).

[Kri65] Krichevskij, R.E., A minimal scheme of switching contacts for a Boolean function of a variables. *Diskretnyj Analiz* 5 (1965), 89-92 (in Russian).

[Ku85] Kurcabová, V., Communication complexity. Master's thesis, Dept. of Theoretical Cybernetics, Comenius University, Bratislava 1985 (in Slovak).

[Ku89] Kumičáková-Jirásková G., Chomsky hierarchy and communication complexity, J. *Inform. Process. Cybernet.EIK* 25 (1989), 157-164.

[Kus92] Kushilevitz, E., Privacy and communication complexity. *SIAM J. Discrete Mathematics* 5 (1992), 273-284.

[LR81] Leighton, F.T., Rosenberg, A.L., Three dimensional circuit layouts. Unpublished memorandum, MIT, Cambridge 1981, Mass.

[LRSH88] Ložkin, C.A., Rybko, A.N., Sapoženko, A.A.,Hromkovič, J., Škalikova, N.A., On an approach to a bound on the area complexity of combinational circuits. In: *Mathematical Problems in Computation Theory*, Banach Center Publications, Warsaw 1988, pp. 501-510 (in Russian).

[LS81] Lipton, R.J., Sedgewick, R., Lower bounds for VLSI. *Proc. 13th ACM STOC*, ACM 1981, pp. 300-307.

[LS88] Lovász, L., Saks, M., Lattices, Möbius functions and communication complexity. *Proc. 29th IEEE FOCS*, IEEE 1988, pp. 81-90.

[LT79] Lipton, R.J., Tarjan, R.E., A separator theorem for planar graphs. *SIAM J. Applied Mathematics* 36 (2) (1979), 177-189.

[LT80] Lipton, R.J., Tarjan, R.E., Applications of a planar separator theorem. *SIAM J. Computing* 9 (1980), 615-627.

[LTT92] Lam, T., Tiwari, P., Toma, M., Trade-offs between communication and space. *J. of Computer and System Sciences* 45 (1992), 296-315.

[LaR81] Lam, T.W., Ruzzo, W.L., Results on communication complexity classes. *Technical Report 89-03-05,* University of Washington 1981(also *J. of Computer and System Sciences* 44 (1992), 324-342).

[Le81] Leighton, F.T., New lower bound techniques for VLSI. *Proc. 22nd IEEE FOCS*, IEEE 1981, pp. 1-12.

[Le82] Leighton, F.T., A layout strategy for VLSI which is provably good. *Proc. 14th ACM STOC*, ACM 1982, pp. 85-98

[Le90] Lengauer, Th., VLSI Theory. In: *Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity*, Elsevier 1990, pp. 835-868.

[Le90a] Lengauer, Th., *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley - Teubner Series in Computer Science, Wiley, J.W., Teubner, B.G., Chichester - Stuttgart 1990.

[Lei80] Leiserson, C.E., Area efficient graph algorithms (for VLSI). *Proc. 21st IEEE FOCS*, IEEE 1980, pp. 270-281.

[Li84] Li, M., On one tape versus two stacks. Technical Report 84-591, January 1984, Dept. of Computer Science. Cornell University, Ithaca, New York.

[Lo89] Lovász, L., Communication Complexity: A Survey. *Techn. Report CS-TR-204-89*, Princeton University, 1989 (also in: *Paths, Flows and VLSI Layout* (Korte, Lovász, Promel, and Schrijver, eds.), Springer-Verlag, Berlin 1990, pp. 235-266).

[Lu58] Lupanov, O.B., Ob odnom methode sinteza skhem. *Izv. Vyssh. Uchebn. Zaved.* 1, 1958, pp. 120-140 (in Russian).

[Lu63] Lupanov, O.B., On the synthesis of some classes of control systems. *Problemy Kibernetiki* 10 (1963), 63-97 (in Russian).

[M76] Masek, A fast algorithm for the string editing problem and decision graph complexity. M. Sc.-thesis, MIT 1976.

[MC80] Mead, C.A., Conway, L.A., *Introduction to VLSI Systems.* Addison-Wesley 1980, Reading Mass.

[MP87] McColl, W.F., Paterson, M.S., The planar realization of Boolean functions. *Information Processing Letters* 24 (1987), 165-170.

[MS82] Mehlhorn, K., Schmidt,E., Las Vegas is better than determinism in VLSI and distributed computing. *Proc. 14th Annual ACM Symposium on Theory of Computing*, San Francisco, ACM 1982, pp. 330-337.

[MSS87] Maas, W., Schnitger, G., Szemeredi, E., Two tapes are better than one for off-line Turing machines. *Proc. 19th Annual ACM Symposium on Theory of Computing*, ACM 1987, pp. 94-100.

[MSST90] Maas, W., Schnitger, G. Szemeredi, D., Turán, G., Two tapes are better than one for off-line Turing machines. Tech. Report CS-90-30, Dept. of Computer Science, The Pensylvania State University, 1990.

[MW91] Meinel, Ch., Waack, S., Upper and lower bounds for certain graph-accessibility-problems on bounded alternating ω-branching programs, *Proc. MFCS'91, Lecture Notes in Computer Science* 520, Springer-Verlag 1991.

[MW92] Meinel, Ch., Waack, S., Lower bounds for the probabilistic communication complexity of various graph-accessibility problems. *Preprint*, 1992.

[Ma84] Maas, W., Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines. *Proc. 16th ACM STOC,* ACM 1984, pp. 401-408.

[MadH84] Meyer auf der Heide, F., A polynomial linear search algorithm for the h-dimensional knapsack problem. *J. of Association for Computing Machinery* 31 (1984), 668-676.

[MadH85] Meyer auf der Heide, F., Lower bounds for solving linear diophantine equations on random access machines. *J. of Association for Computing Machinery* 32 (1985), 929-937.

[MadH85a] Meyer auf der Heide, F., Nondeterministic versus probabilistic linear search algorithms. *Proc. 26th IEEE FOCS*, IEEE 1985, pp. 65-73.

[Mc85a] McColl, W.F., On the planar monotone computation of threshold functions. *Proc. 2nd STACS'85, Lecture Notes in Computer Science* 182, Springer-Verlag, Berlin, 1985, pp. 219-230.

[Mc85b] McColl, W.F., Planar circuits have short specifications. *Proc. 2nd STACS'85, Lecture Notes in Computer Science* 182, Springer-Verlag, Berlin 1985, pp. 231-242.

[McP87] McColl, W. F., Paterson, M. S., The planar realization of Boolean functions. *Information Processing Letters* 24 (1987), 165-170.

[Me87] Meinel, Ch., The power of nondeterminism in polynomial-size bounded-nidth branching programs. *Proc. FCT'87, Lecture Notes in Computer Science* 278, Springer-Verlag 1987 pp. 302-309.

[Me88] Meinel, Ch., *Modified Branching Programs and Their Computational Power. Lecture Notes in Computer Science* 370, Springer-Verlag 1989.

[Me92] Meinel, Ch., A note on Möbius functions and the communication complexity of the graph-accessibility problem. Technical report, University of Trier 1992.

[Mi96] Minato, S., *Binary Decision Diagrams and Applications for VLSI CD* , Kluwer Academic Publishers 1996.

[NW91] Nisan, N., Wigderson, A., Bounds in communication complexity revised. *32nd ACM STOC*, ACM 1991, pp. 419-429 (also *SIAM J. Computing* 22 (1993), 211-219).

[NW94] Nisan, N., Wigderson, A., On ranks vs. communication complexity. *Proc. 35th IEEE FOCS*, IEEE 1994, pp. 831-836 (also *Combinatorica* 15 (1995), 557-565).

[Ne66] Nechiporuk, E. I., A Boolean function. *Sov. Math. Dokl.* 7 (1966), 999-1000.

[New91] Newman, I., Private versus common random bits in communication complexity. *Information Processing Letters* 39 (1991), 67-71.

[Ni83] Nigmatulin, A.N., *The Complexity of Boolean Functions.* The Press of Kazan University, Kazan, 1983 (in Russian).

[Nis93] Nisan, N., The communication complexity of threshold gates. *Combinatorics* 1 (1993), 301-315.

[OG88] Orlitsky, A., El Gamal, A., Communication Complexity, In: *Complexity in Information Theory* (Y. Abu-Mostafa, ed.), Springer-Verlag 1988.

[PS82] Papadimitriou, Ch., and Sipser, M., Communication complexity. *Proc. 14th ACM STOC*, San Francisco, ACM 1982, pp. 196-200.

[PS84a] Papadimitriou, Ch. - Sipser, M., Communication complexity. *J. of Computer and System Sciences* 28 (1984), 260-269.

[PV80] Preparata, F.P., Vuillemin, J.E., Area-time optimal VLSI networks for multiplying matrices. *Information Processing Letters* 11 (1980), 70-80.

[PV81] Preparata, F.P., Vuillemin, J.E., Area-time optimal VLSI networks for computing integer multiplication and discrete Fourier transform. *Proc. 8th ICALP*, Springer-Verlag 1981, pp. 29-40.

[PVa76] Paterson, M., Valiant, L.G., Circuit size is non-linear in depth. *Theoretical Computer Science* 2 (1976), 397-400.

[PZ83] Pudlák, P., Žák, S., Space complexity of computations. Techn. Report, Prague 1983.

[Pa77] Paul, W.J., A $2.5n$-lower bound on the combinational complexity of Boolean functions, *SIAM J. Computing* (1977), 427-443.

[PaS83] Paturi, R., Simon, J., Lower bounds on the time of probabilistic on -line simulations. *Proc. 24th IEEE FOCS*, IEEE 1983, pp. 343-350.

[PaS84] Paturi, R., Simon, J., Probabilistic communication complexity. *Proc. 25th IEEE FOCS*, IEEE 1984, pp. 118-126 (also *J. of Computer and System Sciences* 33 (1986), 106-123).

[Pu84] Pudlák, P., A lower bound on the complexity of branching programs. *Proc. 11th MFCS'84*, *Lecture Notes in Computer Science* 176, Springer-Verlag 1984, pp. 480-489.

[Pu84a] Pudlák, P., Bounds for Hodes-Specker theorem. *Lecture Notes in Computer Science* 171, Springer-Verlag, Berlin 1984, pp. 421-445.

[RS93] Raz, R., Spiker, B., On the Log-Rank conjecture in communication complexity. *Proc. 34th IEEE FOCS* , IEEE 1993, pp. 168-176 (also *Combinatorica* 15 (1995), 567-588).

[RV76] Rivest, R.L., Vuillemin, J., On recognizing graph properties from adjacency matrices. *Theoretical Computer Science* 3 (1976), 371-384.

[RW89] Raz, R., Wigderson, A., Probabilistic communication complexity of Boolean relations. *Proc. 30th IEEE FOCS*, IEEE 1989.

[RW90] Raz, R., Wigderson, A., Monotone circuits for matching require linear depth. *Proc. 22nd ACM STOC*, ACM 1990, pp. 287-292. (also *J. of Association for Computing Machinery* 39 (1992), 736-744).

[RY78] Rivest, R.L., Yao, A.C., k+1 heads are better than k. *J. of Association for Computing Machinery* 25 (1978), 337-340.

[Ra85] Razborov, A.A., Lower bounds for the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR* 281 (1985), 798-801 (in Russian) [English transl. in: *Sov. Math. Dokl.* 31 (1985), 354-357].

[Ra85a] Razborov, A.A., Lower bounds on the monotone network complexity of the logical permanent. *Mat. Zametki* 37 (1985), 887-900 (in Russian) [English transl. in *Math. Notes of the Academy of Sciences of USSR* 37 (1985), 485-493].

[Ra89] Razborov, A. A., On the method of approximations. *Proc. 21st ACM STOC*, ACM 1989, pp. 167-176.

[Ra90] Razborov, A.A., On the distributed complexity of disjointness, *Proc. 17th ICALP* , Springer-Verlag 1990, pp. 249-253. (also *Theoretical Computer Science* 106 (1992), 385-390.)

[Ra90b] Razborov, A. A., Applications of matrix methods for the theory of lower bounds in computational complexity. *Combinatorica* 10 (1990), 81-93.

[Re70] Reďkin, N.P., The proof of the optimality of some schemes of funtional elements. *Problemi Kibernetiki* 23 (1970), 83-101 (in Russian).

[Re81] Reďkin, N. P., Minimal realization of a binary adder. *Problemi Kibernetiki* 38 (1981), 181-216 (in Russian).

[Ro81] Rosenberg, A.L., Three-dimensional integrated circuitry. In *VLSI Systems and Computations* (Kung,H.T., Sproull, R., Steele, G., Eds.), Computer Science Press 1981, pp. 69-79.

[Ry85] Rychkov, K.L., A modification of the method of Chrapchenko and its application for estimations of the complexity of π-schemes for code functions. *Diskretnyj Analiz* 42 (1985), 91-98 (in Russian).

[SB91] Siu, K.I., Bruck, J., On the power of threshold circuits with small weights. *SIAM J. Discrete Mathematics* 4 (1991), 423-435.

[SS96] Schmetzer, Ch., Schnitger G., Kommunikationskomplexität. Lecture notes, University of Frankfurt, 1996 (in German).

[SV86] Sántha, M., Vazirani, U.V., Generating quasi-random sequences from semi-random sources. *J. of Computer and System Sciences* 33 (1986), 75-87.

[SZ97] Savický, P., Žák, S., A lower bound for branching programs reading some bits twice. *Theoretical Computer Science*, to appear.

[Sa76] Savage, J.E., *The Complexity of Computing,* Wiley, New York, 1976.

[Sa81] Savage, J.E., Area-time tradeoffs for matrix multiplication and related problems in VLSI models. *J. of Computer and System Sciences* 20 (1981), 230-242.

[Sa84] Savage, J.E., Multilective VLSI algorithms. *J. of Computer and System Sciences* (1984), 243-273.

[Sc74] Schnorr, G.P., Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen. *Computing* 13 (1974), 155-171.

[Sc76] Schnorr, G.P., The combinational complexity of equivalence. *Theoretical Computer Science* 1 (1976), 289-295.

[Sc80] Schnorr, G.P., A $3 \cdot n$ lower bound on the network complexity of Boolean functions. *Theoretical Compututer Science* 10 (1980), 83-92.

[Sh49] Shannon, C., The synthesis of two-terminal switching circuits. *Bell Syst. Techn. J.* 28 (1949), 59-98.

[Sho80] Schönhage, A.A., Storage modification machines. *SIAM J. Computing* 9 (1980), 490-508.

[Sk76] Škalikova, N.A., On the complexity of the realization of some functions by lattice circuits. *Sbornik rabot po matematičeskoj kibernetike 1*, 1976, pp. 102-115 (in Russian).

[Sk82] Škalikova, N.A., On the relation between the area complexity and space complexity of Boolean circuits. *Metody diskretnogo analiza v ocenkach zložnosti upravljajuščich sistem* 38, 1982, pp. 87-107 (in Russian).

[Sm90] Smolensky, R., On interpolation by analytical functions with special properties and some weak lower bounds on the size of circuits with symmetrie gates. *Proc. 31st IEEE FOCS*, IEEE 1990, pp. 628-631.

[Sn82] Snir, M., Comparisons between linear functions may help. *Theoretical Computer Science* 19 (1982), 321-330.

[Sn85] Snir, M., Lower bounds on probabilistic decision trees. *Theoretical Computer Science* 38 (1985), 69-82.

[So65] Soprunenko, E. P., Minimal realization of functions by circuits using functional elements. *Problemy Kibernetiki* 15 (1965), 117-134 (in Russian).

[St76] Stockmeyer, L.J., On the combinational complexity of certain symmetric functions. *Mathematical System Theory* 10 (1976), 323-336.

[Su61] Subbotovskaja, B.A., On the realization of linear functions by formulas in the base $\vee, \wedge, \neg$. *Dokl. Akad. Nauk. SSSR* 136 (1961), 553-555 (in Russian).

[Ta88] Tardos, E., The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica* 8 (1988), 141-142.

[Th79] Thompson, C.D., Area-time complexity for VLSI. *Proc. 11th ACM STOC*, ACM 1979, pp. 81-88.

[Th80] Thompson, C.D., A complexity theory for VLSI, Doctoral dissertation. CMU-CS-80-140, Computer Science Department, Carnagie-Mellon University, Pittsburgh, August 1980, 131 p.

[Ti87] Tiwari, P., Lower bounds on communication complexity in distributed computer networks. *J. of Association for Computing Machinery* 34 (1987), pp. 921-938.

[Tu89] Turán, G. (1989), Lower bounds for synchronous circuits and planar circuits. *Information Processing Letters* 130, pp. 37-40.

[Ul84] Ullman, J.D., *Computational Aspects of VLSI.* Computer Science Press, 1984, 495 p.

[Va87] Vazirani, U.V., Strong communication complexity or generating quasi-random sequences from two communicating semi-random sources. *Combinatora* 7 (1987), No. 4, 375-392.

[Vu80] Vuillemin, J., A combinatorial limit to the computing power of VLSI circuits. *Proc. 21st ACM STOC*, ACM 1980, pp. 294-300.

[We84] Wegener, I., Optimal decision trees and one-time-only branching programs for symmetric Boolean functions. *Information and Computation* 62 (1984), 129-143.

[We84a] Wegener, I., On the complexity of branching programs and decision trees for clique functions. Techn. Rep., University of Frankfurt 1984 (also *J. of Association for Computing Machinery* 35 (1988), 461-471).

[We86] Wegener, I., Time-space trade-offs for branching programs. *J. of Computer and System Sciences* 32, 91-96.

[We87] Wegener, I., *The Complexity of Boolean Functions.* Wiley-Teubner Series in Computer Science, John Wiley and Sons Ltd., and Teubner, B.G., Stuttgart, 1987.

[Wi91] Wigderson, A., Information theoretic reasons for computational difficulty or communication complexity for circuit complexity. *Proc. of the Inter-*

national Congress of Mathematicians, Kyoto, Japan, 1990, The Mathematical Society of Japan 1991, pp. 1537-1548.

[YR80] Yao, A.C., Rivest, R., On the polyhedual decision problem. *SIAM J. Computing* 9 (1980), 343-347.

[Ya77] Yao, A.C., Probabilistic computations: Towards a unified measure of complexity. *Proc. 18th IEEE FOCS*, IEEE 1977, pp. 222-227.

[Ya79] Yao, A.C., Some complexity questions related to distributive computing. *Proc. 11th ACM STOC*, ACM 1979, pp. 209-213.

[Ya81] Yao, A.C., The Entropic Limitations on VLSI computations. *Proc. 13th ACM STOC*, ACM 1981, pp. 308-311.

[Ya82] Yao, A.C., Protocols for secure computations. *Proc. 23rd IEEE FOCS*, IEEE 1982, pp. 160-164.

[Ya83] Yao, A.C., Lower bounds by probabilistic arguments. *Proc. 25th ACM STOC*, ACM 1983, pp. 420-428.

[Ya86] Yao, A.C., How to generate and exchange secrets. *Proc. 27th IEEE FOCS*, IEEE 1986, pp. 162-167.

[Yan88] Yannakakis, M., Expressing combinatorial optimization problems by linear programs. *Proc. 20th ACM STOC*, ACM 1988, pp. 223-228.

[Za84] Žák, S., An exponential lower bound for one-time-only branching programs. *Proc. 11th MFCS' 84, Lecture Notes in Computer Science* 176, Springer-Verlag 1984, pp. 562-566.

[Za86] Žák, S., An exponential lower bound for real-time branching programs. *Information and Control* 71 (1986), 87-94.

[Za95] Žák, S., A superpolynomial lower bound for $(1 + k(n))$-branching programs. *Proc. MFCS'95, Lecture Notes in Computer Science* 969, Springer-Verlag 1995, pp. 319-325.

Index

Texts in Theoretical Computer Science – An EATCS Series

J. L. Balcázar, J. Díaz, J. Gabarró
Structural Complexity I
2nd ed. (see also overleaf, Vol. 22)

M. Garzon
Models of Massive Parallelism
Analysis of Cellular Automata and Neural Networks

J. Hromkovič
Communication Complexity and Parallel Computing

A. Leitsch
The Resolution Calculus

A. Salomaa
Public-Key Cryptography
2nd ed.

K. Sikkel
Parsing Schemata
A Framework for Specification and Analysis of Parsing Algorithms

Monographs in Theoretical Computer Science – An EATCS Series

C. Calude
Information and Randomness
An Algorithmic Perspective

K. Jensen
Coloured Petri Nets
Basic Concepts, Analysis Methods and Practical Use, Vol. 1
2nd ed.

K. Jensen
Coloured Petri Nets
Basic Concepts, Analysis Methods and Practical Use, Vol. 2

A. Nait Abdallah
The Logic of Partial Information

Former volumes appeared as EATCS Monographs on Theoretical Computer Science

Vol. 5: W. Kuich, A. Salomaa
Semirings, Automata, Languages

Vol. 6: H. Ehrig, B. Mahr
Fundamentals of Algebraic Specification 1
Equations and Initial Semantics

Vol. 7: F. Gécseg
Products of Automata

Vol. 8: F. Kröger
Temporal Logic of Programs

Vol. 9: K. Weihrauch
Computability

Vol. 10: H. Edelsbrunner
Algorithms in Combinatorial Geometry

Vol. 12: J. Berstel, C. Reutenauer
Rational Series and Their Languages

Vol. 13: E. Best, C. Fernández C.
Nonsequential Processes
A Petri Net View

Vol. 14: M. Jantzen
Confluent String Rewriting

Vol. 15: S. Sippu, E. Soisalon-Soininen
Parsing Theory
Volume I: Languages and Parsing

Vol. 16: P. Padawitz
Computing in Horn Clause Theories

Vol. 17: J. Paredaens, P. DeBra, M. Gyssens, D. Van Gucht
The Structure of the Relational Database Model

Vol. 18: J. Dassow, G. Páun
Regulated Rewriting in Formal Language Theory

Vol. 19: M. Tofte
Compiler Generators
What they can do, what they might do, and what they will probably never do

Vol. 20: S. Sippu, E. Soisalon-Soininen
Parsing Theory
Volume II: LR(k) and LL(k) Parsing

Vol. 21: H. Ehrig, B. Mahr
Fundamentals of Algebraic Specification 2
Module Specifications and Constraints

Vol. 22: J. L. Balcázar, J. Díaz, J. Gabarró
Structural Complexity II

Vol. 24: T. Gergely, L. Úry
First-Order Programming Theories

R. Janicki, P. E. Lauer
Specification and Analysis of Concurrent Systems
The COSY Approach

O. Watanabe (Ed.)
Kolmogorov Complexity and Computational Complexity

G. Schmidt, Th. Ströhlein
Relations and Graphs
Discrete Mathematics for Computer Scientists

S. L. Bloom, Z. Ésik
Iteration Theories
The Equational Logic of Iterative Processes

Springer and the environment

At Springer we firmly believe that an international science publisher has a special obligation to the environment, and our corporate policies consistently reflect this conviction.

We also expect our business partners – paper mills, printers, packaging manufacturers, etc. – to commit themselves to using materials and production processes that do not harm the environment. The paper in this book is made from low- or no-chlorine pulp and is acid free, in conformance with international standards for paper permanency.

Springer