



---

Basic Research in Computer Science

BRICS NS-94-3

S. Skyum (ed.): Complexity Theory - Present and Future

## Complexity Theory - Present and Future

15–18 August 1994, Aarhus, Denmark

Sven Skyum (editor)

BRICS Notes Series

NS-94-3

---

ISSN 0909-3206

September 1994

**See back inner page for a list of recent publications in the BRICS Notes Series. Copies may be obtained by contacting:**

**BRICS  
Department of Computer Science  
University of Aarhus  
Ny Munkegade, building 540  
DK - 8000 Aarhus C  
Denmark  
Telephone: +45 8942 3360  
Telefax: +45 8942 3255  
Internet: BRICS@brics.dk**

**BRICS publications are in general accessible through WWW and anonymous FTP:**

`http://www.brics.dk/  
ftp ftp.brics.dk (cd pub/BRICS)`

# Complexity Theory - Present and Future

15-18 August 1994 — Aarhus, Denmark

Sven Skyum (editor)

## Preface

These “proceedings” contain slides, overviews and papers on which the conference talks were based.

The conference was a byproduct of a longer meeting for a relatively small number of researchers in complexity theory, hosted by BRICS, which took place in Aarhus during the months of August and September, 1994.

On Friday, August 12, two preconference lectures were given, aimed at those who were not complexity theory experts, and introducing the listener to areas, methods and concepts of the field.

During the actual conference (August 15-18) there were two main talks each day followed by a session with talks on more specific subjects.

We would like to thank Shmuel Safra, Hebrew University, Jerusalem, who took the initiative to the meeting and with whom we co-organized the meeting. Finally we would like to thank Karen Kjær Møller for extra-ordinary engagement in organizing the meeting and the conference.

Sven Skyum

# Complexity theory - Present and Future

## Preconference Lectures

**Friday, August 12**

*page*

10.15-12.00    *Noam Nisan, The Hebrew University, Jerusalem, Israel*    1-11

### Communication Complexity: an Introduction

Abstract:

Yao's model of two-party communication complexity aims to capture, in the simplest way, a situation where communication plays a role. We will define the model and then concentrate on how to analyse complexity in this model. We will present basic techniques for lower bounds and will study the power of nondeterminism and of randomization. We will also give applications to Turing machines and to circuits.

The talk will be of a tutorial nature; it will require no prior knowledge, but will assume a mathematically-oriented audience.

14.15-15.15    *Avi Wigderson, The Hebrew University, Jerusalem, Israel*    13-27

### The wonders of the digital envelope - a crash course in modern cryptography

Abstract:

The "One-way function" (or "digital envelope") was suggested 15 years ago as means for solving the most basic cryptographic tasks - secret communication. Since then it was gradually discovered that this simple device is in fact universal, and can be used to solve essentially ANY cryptographic task with given secrecy and fault-tolerant constraints. In this talk I will try to survey the key ideas that led to this understanding, including (naturally) a "zero-knowledge proof" demonstration.

# Complexity Theory - Present and Future

## Program

### Monday, August 15

page

10.30	Opening	
10.35-12.00	<i>Allan Borodin, Toronto University</i> Trade offs between time and space	29-41
12.00-13.30	Lunch	
13.30-15.00	<i>Adi Shamir, Weizmann Institute, Rehovot</i> Open problems in cryptocomplexity	43
15.00-15.30	Coffee	
15.30-	<i>Dexter Kozen, Cornell University, Ithaca</i> Efficient average-case algorithms for the modular group	45-55

### Tuesday, August 16

10.30-12.00	<i>Noam Nisan, The Hebrew University, Jerusalem</i> Direct sums, products, and help bits in circuits and decision trees	57-69
12.00-13.30	Lunch	
13.30-15.00	<i>Alexander Razborov, Steklov Institute, Moscow</i> Independence results in bounded arithmetic and natural proofs	71-112
15.00-15.30	Coffee	
15.30-	<i>Russell Impagliazzo, Univ. of California at San Diego</i> Hard core distributions for somewhat hard functions	113-119
	<i>Amnon Ta-Shma, The Hebrew University, Jerusalem</i> Symmetric Log-space is closed under complement	121-139

**Wednesday, August 17**

page

10.30-12.00	<i>Peter Bro Miltersen, BRICS</i> On cell probe complexity	141-166
12.00-13.30	Lunch	
13.30-15.00	<i>Michael Ben-Or, The Hebrew University, Jerusalem</i> On algebraic complexity theory	167
15.00-15.30	Coffee	
15.30-	<i>Christoph Meinel, University of Trier</i> Modular communication complexity of UCON	169-184
	<i>Søren Riis, BRICS</i> Complexity of counting principles	185-189

**Thursday, August 18**

10.30-12.00	<i>Mark Jerrum, University of Edinburgh</i> Approximation via semidefinite programming relaxations	191-205
12.00-13.30	Lunch	
13.30-15.00	<i>Avi Wigderson, The Hebrew University, Jerusalem</i> On rank and communication complexity	207-213
15.00-15.30	Coffee	
15.30-	Discussions	

**Lecturer: Noam Nisan**

**Title: Communication Complexity:  
an Introduction**

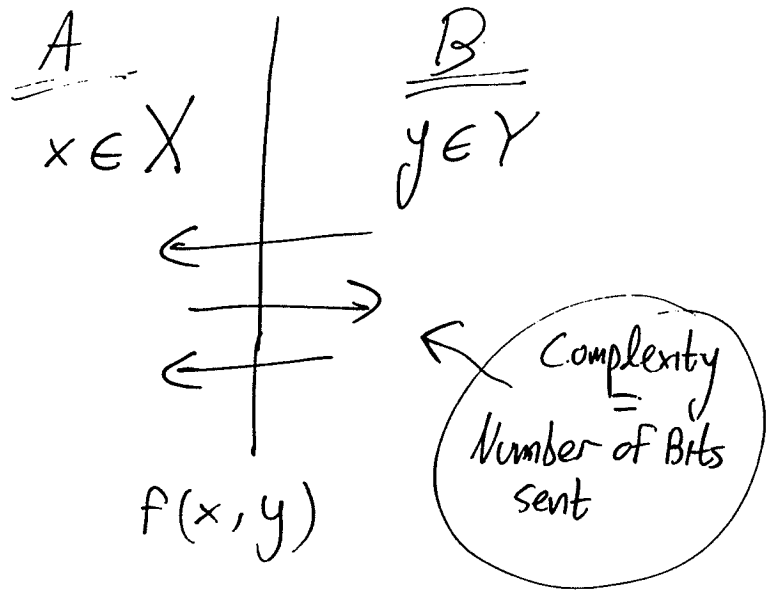
**Material: Slides**

Institute of Computer Science  
Hebrew University  
Jerusalem, Israel  
E-mail: noam@CS.HUJI.AC.IL



# COMMUNICATION COMPLEXITY

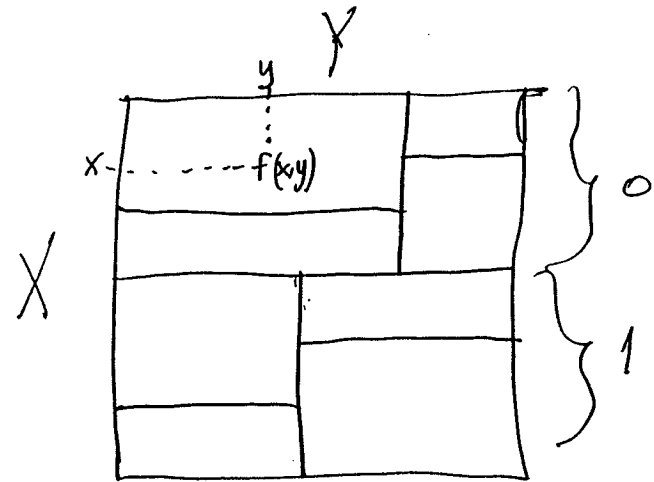
Yao '79



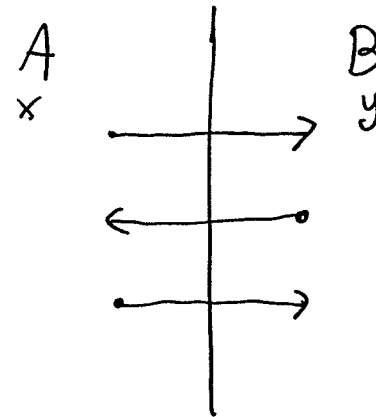
## PROTOCOL:

- (1) Who Speaks at any Point.
- (2) What is sent.  
(If A speaks: function of  $x$  and conversation so far. If B speaks: function of  $y$  and conversation so far)
- (3) When it terminates, and what the answer is (Depends only on the conversation)

## MATRIX OF $f$



## Protocol



# RECTANGLES

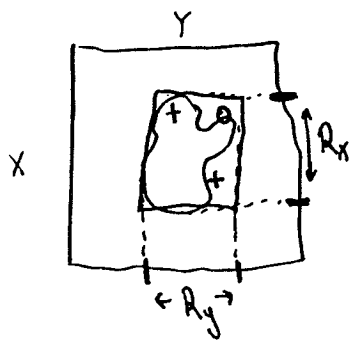
DEFINITION:  $R \subseteq X \times Y$  IS A RECTANGLE  
IF  $R = A \times B$ ,  $A \subseteq X$ ,  $B \subseteq Y$ .

LEMMA: T.F.A.E.

- 1)  $R$  IS A RECTANGLE
- 2)  $\langle x_1, y_1 \rangle \in R, \langle x_2, y_2 \rangle \in R \implies \langle x_1, y_2 \rangle \in R$
- 3)  $R = R_x \times R_y$ , where

$$R_x = \{x \mid \exists y \langle x, y \rangle \in R\}$$

$$R_y = \{y \mid \exists x \langle x, y \rangle \in R\}$$



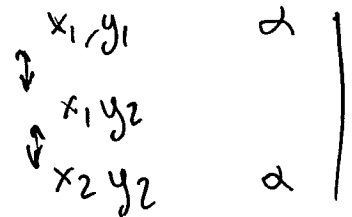
$\langle x, y \rangle \in R_x \times R_y$   
 $\langle x, y' \rangle \in R$   
 $\langle x', y \rangle \in R$

# CC AND RECTANGLES.

LET  $P$  BE A PROTOCOL  
COMPUTING  $f$ . LET  $\alpha$  BE  
TRANSCRIPT OF  $P$ .

FACTS:

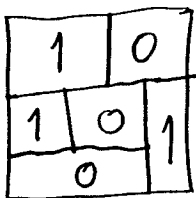
- ①  $R_\alpha = \{ \langle x, y \rangle \mid \text{ON INPUT } \langle x, y \rangle, \text{ THE TRANSCRIP. OF } P \text{ IS } \alpha \}$   
IS A RECTANGLE.
- ②  $f(x, y)$  IS THE SAME  $\forall \langle x, y \rangle \in R_\alpha$ .
- ③  $\{R_\alpha\}_{\alpha \in \text{POSSIBLE TRANSCRIPTS}}$  IS A PARTITION  
OF  $X \times Y$ .



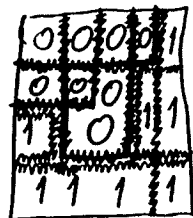
# COVER NUMBERS.

## DEFINITIONS:

- \*  $C(f) =$  MINIMUM NUMBER OF  $f$ -MONOX RECTANGLES NEEDED TO DISJOINTLY COVER  $X \times Y$
- \*  $C^N(f) = \dots$  NOT NECESSARILY DISJOINT COVER...
- \*  $C^1(f) = \dots$  TO COVER THE 1'S OF  $f$
- \*  $C^0(f) = \dots$  TO COVER THE 0'S OF  $f$



$C(f) = 6$



$C^1(f) = 4$   
 $C^0(f) = 3$   
 $C^N(f) = 7$

## FACTS:

- \*  $C(f) \geq C^N(f) = C^1(f) + C^0(f)$
- \*  $D(f) = \text{COMMUNICATION COMPLEXITY OF } f \geq \lceil \lg_2 C(f) \rceil$

# FOOLING SETS

DEFINITION:  $\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \dots, \langle x_t, y_t \rangle$  IS

A FOOLING SET (FOR  $f$ ) IF

- (1)  $\forall i \quad f(x_i, y_i) = 1$
- (2)  $\forall i \neq j \quad f(x_i, y_j) = 0 \text{ OR } f(x_j, y_i) = 0$

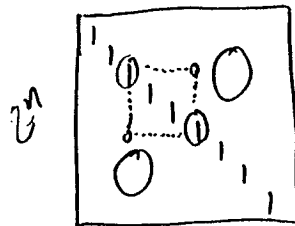
LEMMA: IF  $f$  HAS A FOOLING SET OF SIZE  $t$  THEN  $C^1(f) \geq t \Rightarrow D(f) \geq \lceil \lg_2 t \rceil$

EXAMPLE:

$X = Y = \{0, 1\}^n$   
 $EQ(x, y) = \begin{cases} 1 & x=y \\ 0 & x \neq y \end{cases}$

LEMMA:  $D(EQ) = n$

PROOF:  $\{ \langle x, x \rangle \mid x \in \{0, 1\}^n \}$  IS A FOOLING SET  $2^n$



# 1 RECTANGLE SIZE

LEMMA:  $C^1(F) \geq \frac{\text{NUMBER OF 1'S OF } F}{\text{SIZE OF LARGEST 1-RECTANGLE}}$

EXAMPLE:  $X=Y = \text{SUBSETS OF } \{1..n\}$

$$\text{DISJ}(x,y) = \begin{cases} 1 & x \cap y = \emptyset \\ 0 & x \cap y \neq \emptyset \end{cases}$$

FACT 1: DISJ HAS  $3^n$  1'S

LEMMA: EVERY 1-RECTANGLE HAS SIZE  $\leq 2^n$

PROOF: IF  $A \times B$  IS A 1-RECTANGLE THEN THERE EXISTS A PARTITION:

$\{1..n\} = S \cup T$  SUCH THAT:

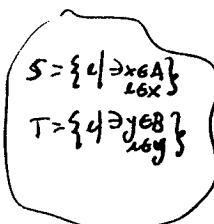
$$x \in A \Rightarrow x \subseteq S$$

$$y \in B \Rightarrow y \subseteq T$$

IF  $|S|=k$ ,  $|T| \leq n-k$

$$|A| \leq 2^k, |B| \leq 2^{n-k}$$

$$|A \times B| \leq 2^n$$

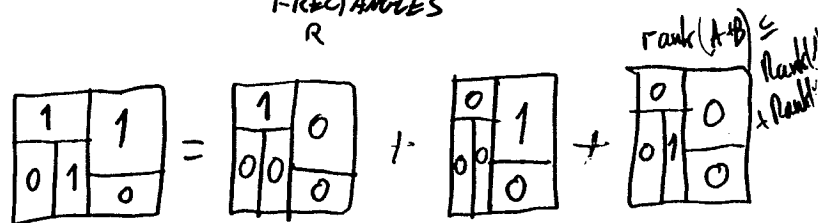


COROLLARY:  $D(\text{DISJ}) = \Omega(n)$

# RANK LOWER BOUND

LEMMA: IF THE MATRIX OF  $F$  HAS RANK  $r$  (OVER SOME FIELD  $\mathbb{F}$ ) THEN  $C(F) \geq r$ .

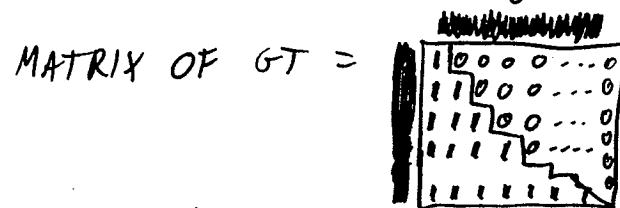
PROOF: MATRIX OF  $F = \sum_{R} 1_R$  (IN DISJOINT COVER)  
 $r = \text{RANK}(\text{MATRIX OF } F) \leq \sum_{R} \text{RANK}(1_R) = C(F) \cdot 1$



NOTE: BEST BOUNDS ACHIEVED WITH  $\mathbb{F} = \mathbb{Q}$

EXAMPLE: ~~GT~~  $X=Y = \{0..2^n-1\}$

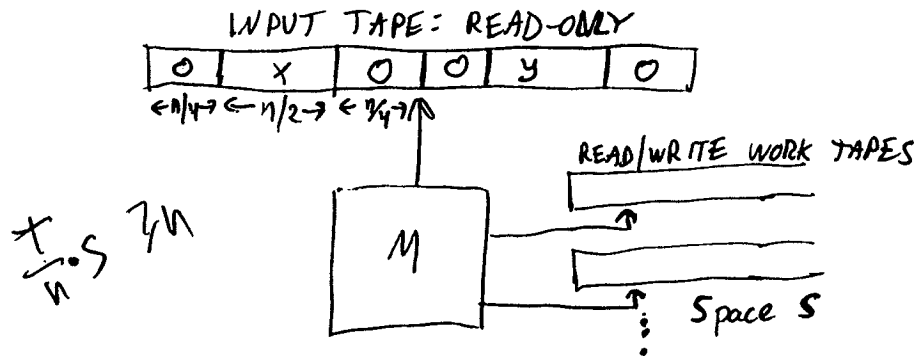
$$\text{GT}(x,y) = \begin{cases} 1 & x \geq y \\ 0 & x < y \end{cases}$$



RANK =  $2^n$

COROLLARY:  $D(\text{GT}) = n$

# TIME-SPACE TRADEOFFS FOR TURING MACHINES (Cobham.....)

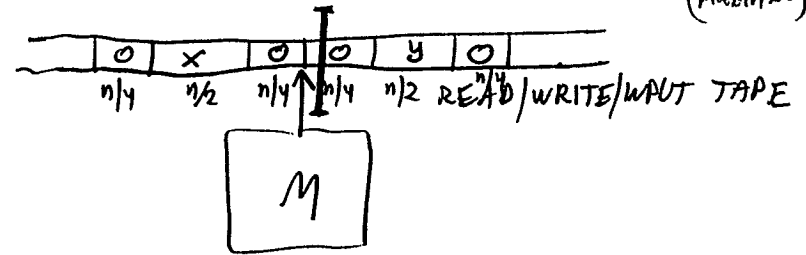


LEMMA: RECOGNISING THE LANGUAGE  $\{ww \mid w \in \{0,1\}^*\}$  REQUIRES A TIME-SPACE TRADEOFF OF  $TS \geq \Omega(n^2)$  ON ANY TM.

PROOF: TWO PLAYERS WISHING TO DECIDE  $EQ(x,y)$  CAN SIMULATE  $M$ : THE X-PLAYER CAN SIMULATE  $M$  AS LONG AS THE INPUT HEAD IS IN THE X-SECTION OR IN THE MIDDLE SECTION. WHEN THE HEAD REACHES THE Y-SECTION, THE TOTAL STATE OF  $M$  ( $O(S)$ -BITS) IS SENT TO  $y$ .

(9)

# TIME LOWER BOUNDS FOR 1-TAPE TMs (Aabm...)



LEMMA: ANY 1-TAPE TM RECOGNISING  $\{ww \mid w \in \{0,1\}^*\}$  REQUIRES TIME  $\geq \Omega(n^2)$

ALMOST PROOF: FOR EVERY CELL IN MIDDLE SECTION, THE HEAD MUST CROSS IT AT LEAST  $\Omega(n)$  TIMES: TWO PLAYERS MAY SIMULATE  $M$ , THE X-PLAYER WHEN HEAD IS TO THE LEFT OF  $|$ , WHEN THE HEAD CROSSES  $|$  THE STATE OF THE FINITE CONTROL ( $O(1)$  BITS) MUST BE SENT. TOTAL =  $\frac{n}{2} - \Omega(n) = \Omega(n)$

PROBLEM: THE WORST CASE COMM. COMPLEXITY OF  $\Omega(n)$  MAY BE ACHIEVED ON A DIFFERENT  $x$  FOR EACH MIDDLE CELL.

(10)

# FIXING THE PROOF

NEW LEMMA: IF  $f$  HAS A FOOLING SET OF SIZE  $t$  THEN  $D_{AVG}(f) \geq \lceil \lg_2 t \rceil$ ,  
 WHERE  $D_{AVG}$  IS THE AVERAGE COMPLEXITY OVER ALL POSSIBLE WPTS IN THE FOOLING SET.

PROOF EACH  $\langle x, y \rangle$  IN THE FOOLING SET MUST BE IN A DIFFERENT RECTANGLE.  $R_\alpha$   
~~REQUIRES~~ THE AVERAGE LENGTH OF  $\alpha$  MUST BE  $\geq \lg_2(\# \text{ of } \alpha \text{'s})$ .

NEW PROOF OF TM LOWER BOUND:

SAME AS BEFORE ONLY USING AVERAGE COMPLEXITY OVER ALL INPUTS OF THE FORM: 

0	x	0	0	x	0
---	---	---	---	---	---

.

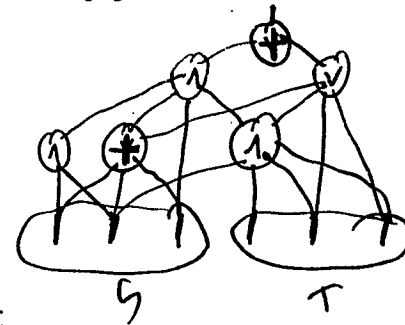
# WORST PARTITION COMPLEXITY

DEFINITION: LET  $f = \{0,1\}^n \rightarrow \{0,1\}$

$$D_{\text{WORST}}(f) = \max_{\substack{\text{OVER ALL PARTITIONS} \\ \{1 \dots n\} = S \cup T}} D_{S \leftrightarrow T}(f)$$

COMMUNICATION COMPLEXITY  
 WHERE  $X = \{0,1\}^S$   
 $Y = \{0,1\}^T$

LEMMA: ANY CIRCUIT OF UNBOUNDED-FANIN AND/OR XOR GATES COMPUTING  $f$  REQUIRES SIZE  $\geq D_{\text{WORST}}(f)$



COROLLARY:

$$f(x_1 \dots x_{2n}) = (x_1 = x_2) \wedge (x_3 = x_4) \wedge (x_5 = x_6) \wedge \dots \wedge (x_{2n-1} = x_{2n})$$

$$\text{SIZE}(f) \geq D_{\text{WORST}}(f) \geq D(\text{EQ}) = n$$

# BEST PARTITION COMPLEXITY

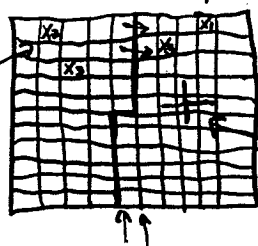
DEFINITION LET  $f = \{0,1\}^n \rightarrow \{0,1\}$

$D_{BEST}(f) = \min_{\substack{\uparrow \\ \text{OVER ALL PARTITIONS} \\ \{S, T\} = \text{SUT} \\ |S| = |T| = n/2}} D_{S \leftrightarrow T}(f)$

EXAMPLE:

VLSI (Thompson)

EACH INPUT BIT APPEARS IN EXACTLY ONE CELL



EACH CELL MAY NOW COMMUNICATE 1 BIT WITH EACH KBR EACH TIME STEP

LEMMA ANY VLSI CHIP WITH AREA  $A$  COMPUTING  $f$  REQUIRES TIME  ~~$T \geq \frac{D_{BEST}(f)}{\sqrt{A}}$~~

$T \geq \frac{D_{BEST}(f)}{\sqrt{A}}$

PROOF THERE EXISTS A ~~PARTITION~~ CUT OF LENGTH  $\sqrt{A} + 1$  PARTITIONING THE INPUT BITS INTO TWO SETS OF SIZE  $n/2$

# LOWER BOUNDS FOR $D_{BEST}$

DEFINITION: CYCLIC-SHIFT  $(x_1 \dots x_n, y_1 \dots y_n, s_1 \dots s_k) =$

$\begin{cases} 1 & \text{if } \forall i: x_{i+s} = y_i \text{ where } s = [s_1 \dots s_k] \\ 0 & \text{OTHERWISE} \end{cases}$

(where  $k = \lg n$ )

THE INTEGERS

LEMMA:  $D_{BEST}(\text{CYCLIC-SHIFT}) = \Omega(n)$

PROOF

LET  $A, B$  BE A PARTITION OF THE  $2n + o(\lg n)$  INPUT BITS, WHERE

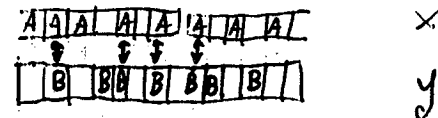
$A$  HAS AT LEAST  $\frac{n}{2} - \lg n$   $x$ -INPUTS

$B$  HAS AT LEAST  $\frac{n}{2} - \lg n$   $y$ -INPUTS

CLAIM THERE EXISTS A VALUE OF  $s$  SUCH THAT FOR AT LEAST  $\frac{n}{4} - o(\lg n)$  VALUES OF  $i: i+s \in A, i \in B$ .

PROOF: RANDOM CHOICE OF  $s$

COROLLARY:  $D_{BEST}(\text{CYCLIC-SHIFT}) \geq D(ER_{\frac{n}{4} - o(\lg n)}) = \Omega(n)$



D vs. L

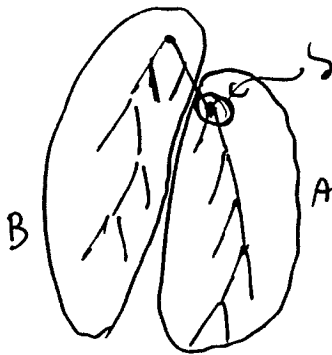
DEFINITION:  $L(F) =$  MINIMUM NUMBER OF DIFFERENT TRANSCRIPTS IN A PROTOCOL THAT COMPUTES  $F$ .

FACT:  $D(F) \geq \lg_2 L(F) \geq \lg_2 C(F)$

DISJOINT COVER

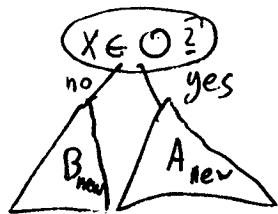
LEMMA:  $D(F) = O(\lg L(F))$

PROOF:



CLAIM: THERE EXISTS A NODE IN THE PROTOCOL WITH  $\frac{L(F)}{3} \dots \frac{2L(F)}{3}$  LEAFS UNDER IT.

NEW PROTOCOL:



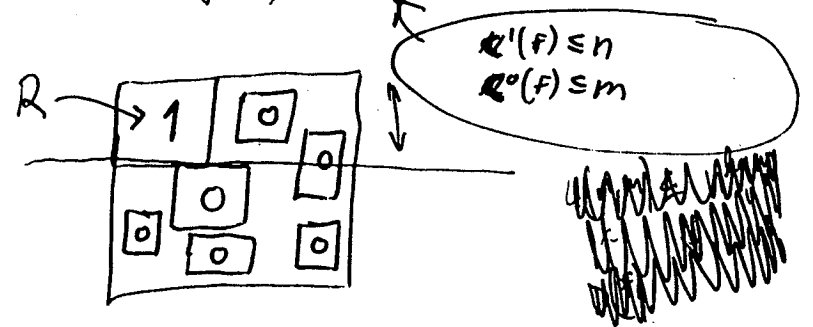
$$D(L) \leq 2 + D\left(\frac{2}{3}L\right)$$

D vs. ~~A~~ (Aho Ullman Panagiotis)

DEFINITION:  $N^1(F) = \lceil \lg_2 C^1(F) \rceil$  A COVER OF THE 1'S OF  $F$   
 $N^0(F) = \lceil \lg_2 C^0(F) \rceil$   
 $N(F) = \max(N^0(F), N^1(F))$

LEMMA:  $D(F) \leq O(N^1(F) \cdot N^0(F)) = O(\lg C(F))^2$

PROOF: LET  $L(n, m) = \max L(F)$



CLAIM: EVERY 0-RECTANGLE DOES NOT INTERSECT  $R$  EITHER ROW-WISE OR COLUMN-WISE.

COROLLARY:  $\begin{cases} L(n, m) \leq L(n-1, m) + L(n, m/2) \\ L(n, 0) = 1 \end{cases} \Rightarrow L(n, m) \leq n \lg m$

$$L(n, m) \leq n \lg m$$

$n^c \geq (n-1)^c + n^{c-1}$   
 SINCE  $n^c - n^{c-1} = (n-1)n^{c-1}$

NOTE: SAME PROOF SHOWS:  $D(F) \leq O(N^1(F) \cdot \lg \text{Rank}(F))$

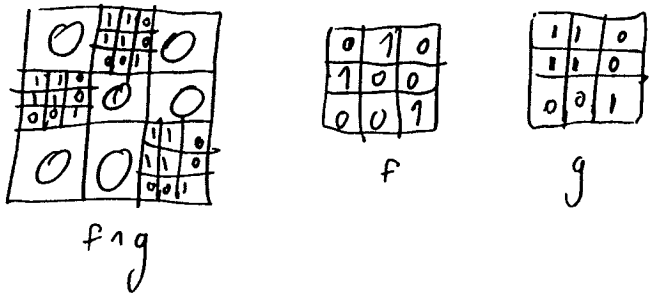


# DIRECT SUM FOR RANK (Mehlhorn Schmidt)

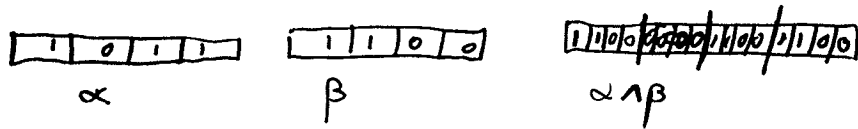
NOTATION LET  $f(x_1; y_1), g(x_2; y_2)$  BE FUNCTIONS.

$$f \wedge g(x_1, x_2; y_1, y_2) = f(x_1; y_1) \wedge g(x_2; y_2)$$

LEMMA:  $\text{RANK}(f \wedge g) = \text{RANK}(f) + \text{RANK}(g)$



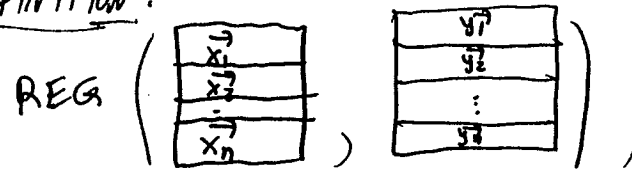
PROOF IF  $\{\alpha_1 \dots \alpha_n\}$  IS A BASIS FOR THE ROWS OF  $f$  AND  $\{\beta_1 \dots \beta_m\}$  IS A BASIS FOR THE ROWS OF  $g$  THEN  $\{\alpha_i \wedge \beta_j\}_{\substack{i=1 \dots n \\ j=1 \dots m}}$  IS A BASIS FOR THE ROWS OF  $f \wedge g$ .



- 1) SPAN EVERYTHING.
- 2) INDEPENDANT.

# D vs. N - SEPARATION (Mehlhorn Schmidt)

DEFINITION:



$$= \begin{cases} 1 & \exists i \ x_i = y_i \\ 0 & \text{OTHERWISE} \end{cases}$$

Each  $\vec{x}_i$  and  $\vec{y}_i$  are  $n$ -bit vectors

LEMMA: 1)  $N^1(REG) = \Theta(n)$

2)  $N^0(REG) = \Theta(n)$

3)  $D(REG) = \Theta(n^2)$

PROOF: 1) THE PROOF IS  $e, x_i$  S.T.  $x_i = y_i$

2)  $\Theta(n \lg n)$ . THE PROOF IS  $j_1 \dots j_n$  S.T.  $x_i[j_i] \neq y_i[j_i] \ \forall i$



$$3) \overline{REG} = \overline{EQ}(x_1, y_1) \wedge \overline{EQ}(x_2, y_2) \dots \wedge \overline{EQ}(x_n, y_n)$$

$$\text{RANK}(\overline{EQ}) \geq \text{RANK}(EQ) - 1 = 2^n - 1$$

↓

$$\text{RANK}(\overline{REG}) \geq (2^n - 1)^n = 2^{\Theta(n^2)}$$

# RANDOMIZED COMPLEXITY

PLAYERS MAY FLIP COINS.

0-ERROR: ANSWER ALWAYS CORRECT;

$R^0(f)$  MAY NOT GIVE ANY ANSWER WITH PROBABILITY  $\leq \frac{1}{2}$

1-SIDED ERROR:  $P(x,y) = 0 \Rightarrow$  ALWAYS ANSWER 0

$R^1(f)$   $P(x,y) = 1 \Rightarrow \Pr[\text{ANSWER } 0] \leq \frac{1}{2}$

2-SIDED ERROR:  $\Pr[\text{CORRECT}] \geq \frac{2}{3}$   
 $R^2(f)$

OBVIOUS RELATIONS:

~~CONFUSING RELATIONS~~

$$1) \quad D(f) \geq R^0(f) \geq R^1(f) \geq O(R^2(f))$$

$$2) \quad R^1(f) \geq N^1(f)$$

# RANDOMIZED COMPLEXITY OF EQ.

THM:

$$1) \quad R^1(EQ) = n$$

$$2) \quad R^1(\bar{EQ}) = \Theta(\lg n)$$

PROOF

$$1) \quad R^1(EQ) \geq N^1(EQ) = \lg(C^1(EQ)) = \lg(2^n)$$

2) A) X chooses a random prime number  $p \in_R [2..n^2]$  AND SENDS TO Y:

$$X \rightarrow Y: \quad p, (x \bmod p)$$

Y ACCEPTS IF:  $(y \bmod p) = (x \bmod p)$

CLAIM 1:

IF  $x=y$  THEN PROTOCOL ALWAYS ACCEPTS.

CLAIM 2:

IF  $x \neq y$  THEN PROTOCOL REJECTS W.H.P.

PROOF: THERE CAN BE AT MOST  $n$  DIFFERENT PRIMES  $p$  SUCH THAT  $p | (x-y)$

2) B)

$$R^1(\bar{EQ}) \geq N^1(\bar{EQ}) \geq \lg_2 D(\bar{EQ}) = \lg_2 n$$

$$D(p) \leq C^1(f)$$

19



**Avi Wigderson:**

# **Wonders of the Digital Envelope**

**Slides**

Institute of Computer Science  
Hebrew University  
Jerusalem, Israel  
E-mail: [avi@CS.HUJI.AC.IL](mailto:avi@CS.HUJI.AC.IL)



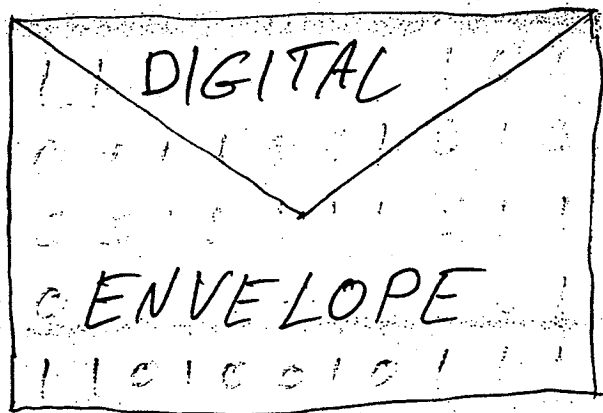
**Lecturer: Avi Wigderson**

**Title: The Wonders of the Digital Envelope  
- a crash course in modern cryptography**

**Material: Slides**

Institute of Computer Science  
Hebrew University  
Jerusalem, Israel  
E-mail: [avi@CS.HUJI.AC.IL](mailto:avi@CS.HUJI.AC.IL)

# WONDERS OF THE



AVI WIGDERSON

HEBREW UNIV.

JERUSALEM

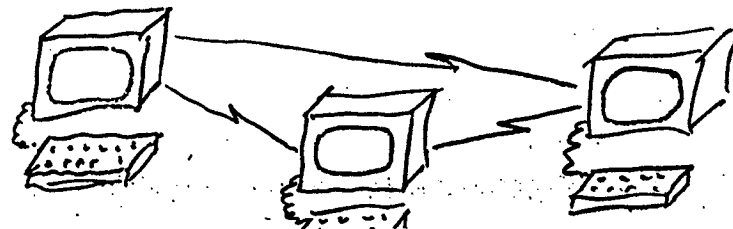
# MODERN CRYPTOGRAPHY

1. SECRECY (PRIVACY)
2. FAULT TOLERANCE

## TASKS

## IMPLEMENTS

- ENCRYPTION - CODE BOOKS
- IDENTIFICATION - DRIVER LICENSE
- MONEY TRANSFER - NOTES, CHECKS
- PUBLIC BIDS - SEALED ENVELOPES
- INFO. PROTECTION - LOCKS
- POKER GAME - PLAY CARDS
- PUBLIC LOTTERY - COINS, DICE
- SIGN CONTRACTS - LAWYERS



# THE MILLIONAIRES' PROBLEM

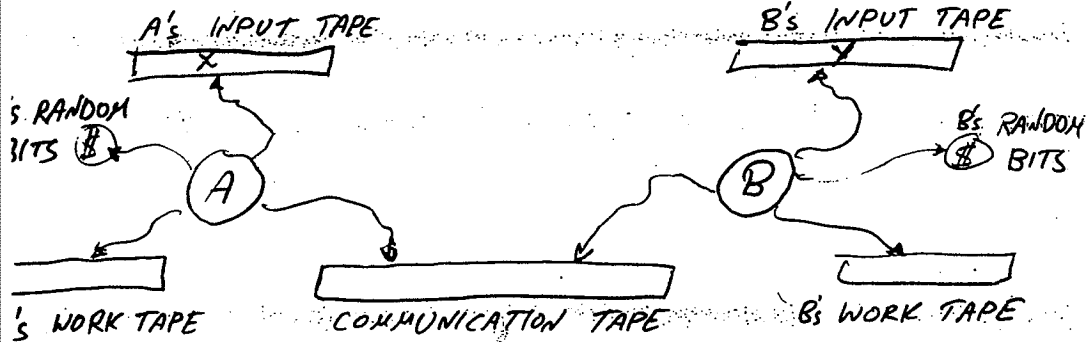
ALICE  
 $1 \leq x \leq 2^{100}$

BOB  
 $1 \leq y \leq 2^{100}$

$$RICH(x,y) = \begin{cases} 1 & x \geq y \\ 0 & x < y \end{cases}$$

- TASKS: (•) COMPUTE  $RICH(x,y)$   
 (••) NO PLAYER LEARNS ANYTHING ELSE

PROTOCOL: A PAIR A, B OF TURING MACHINES



CONVERSATION =  $[A(x), B(y)]$  - A RANDOM VARIABLE

# INFORMATION THEORY (PERFECT)

COMPLETENESS  $\forall x,y [A(x)B(y)]$  DETERMINES  $RICH(x,y)$

PRIVACY  $\forall x, x', y' \langle y, [A(x)B(y)] \rangle \langle y, [A(x')B(y')] \rangle$   
 $RICH(x,y) = RICH(x',y')$  ARE INDISTINGUISHABLE

I.T. (ALMOST PERFECT:  $\epsilon > 0$  TOLERANCE)

COMPLETENESS  $\forall x,y P_2 [ [A(x,\epsilon), B(y,\epsilon)] \text{ DET. } RICH(x,y) ] \geq 1 - \epsilon$

PRIVACY  $\forall x, x', y \left| \langle y, [A(x,\epsilon), B(y,\epsilon)] \rangle, \langle y, [A(x',\epsilon), B(y,\epsilon)] \rangle \right| \leq \epsilon$

COMPUTATIONAL COMPLEXITY ( $\epsilon > 0$ )

ALL PROGRAMS ARE COMPUTATIONALLY BOUNDED

COMP.  $\exists M \forall x,y P_2 [ M([A(x,\epsilon), B(y,\epsilon)]) = RICH(x,y) ] \geq 1 - \epsilon$

PRIVACY  $\forall N, \forall x, x', y \left| P_2 [ N(\langle y, [A(x,\epsilon), B(y,\epsilon)] \rangle) = 1 ] \right|$

$$- P_2 [ N(\langle y, [A(x,\epsilon), B(y,\epsilon)] \rangle) = 1 ] \leq \epsilon$$

FAULT

TOLERANCE: PRIVACY HOLDS FOR ALL  $\tilde{B}$



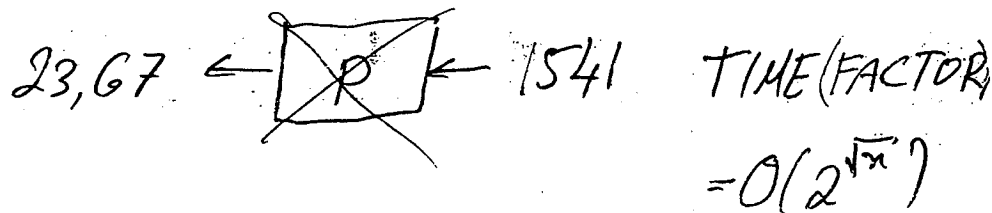
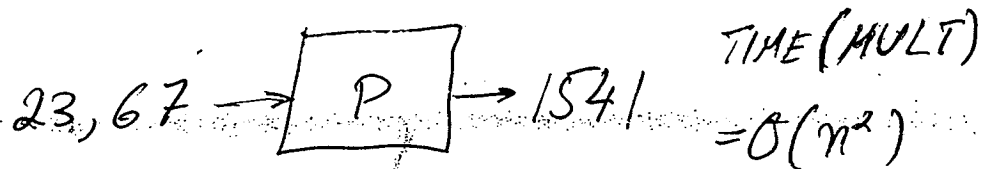
# COMPLEXITY BASED

## CRYPTOGRAPHY

\* PLAYERS ARE EFFICIENT  
COMMUNICATING TURING MACHINES

EFFICIENT = PROB. POLY. TIME

$n$  = BINARY INPUT LENGTH



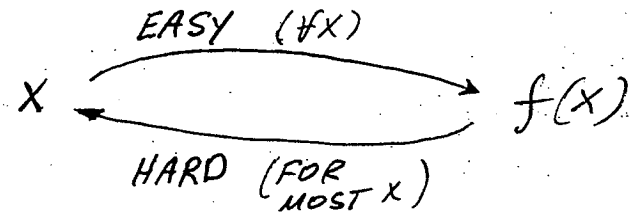
AXIOM FACTORING HAS NO  
EFFICIENT ALGORITHM

[ AXIOM  $\rightarrow$  P + NP ]

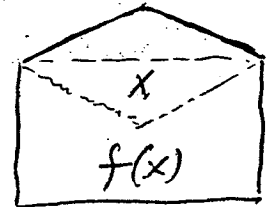
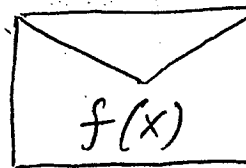
THM:  $\exists$  INCOMPUTABLE FUNCTIONS  $\infty$

AXIOM:  $\exists$  FUNCTIONS e.g.  $f: \{0,1\}^* \rightarrow \{0,1\}^*$

SUCH THAT




$f$  ONE-WAY FUNCTION  $\Rightarrow$  DIGITAL ENVELOPE

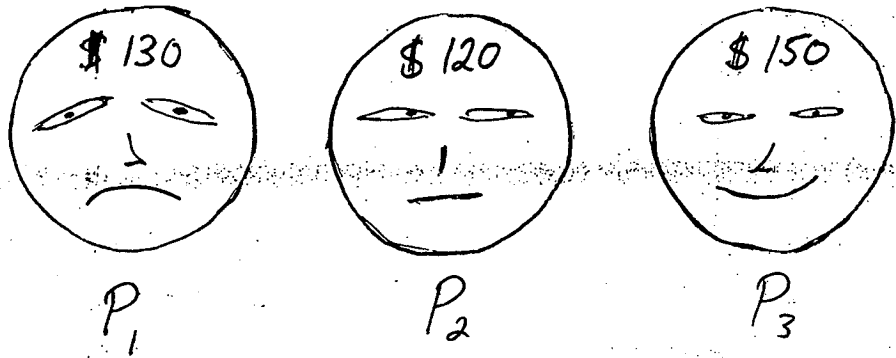


PROPERTIES OF THE ENVELOPE

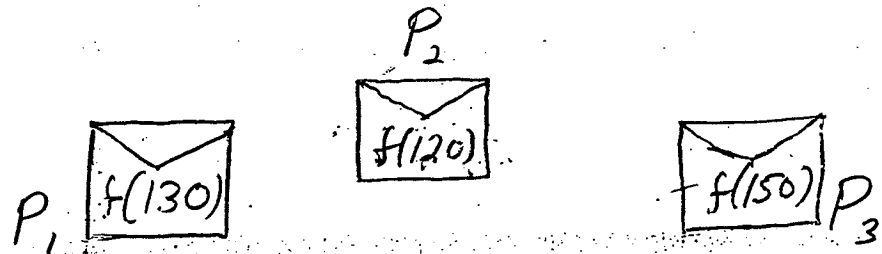
1. EASY TO INSERT  $x$  (ANY VALUE, EVEN 1 BIT)
2. HARD TO COMPUTE CONTENT (EVEN PARTIAL INFO.)
3. IMPOSSIBLE TO CHANGE CONTENT ( $f(x)$  DEFINES  $x$ )
4. EASY TO VERIFY  $x$  = CONTENT

THM:   $\rightarrow$  CRYPTOGRAPHY

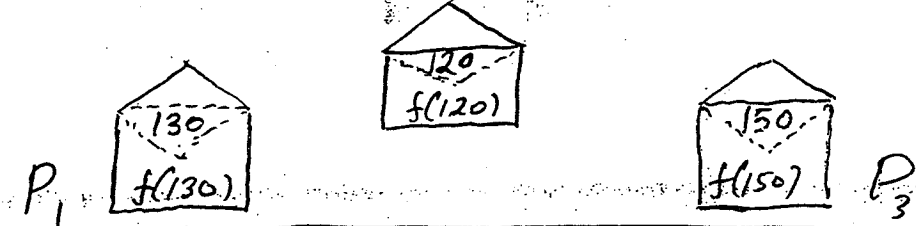
# PUBLIC BID (PLAYERS IN ONE ROOM)




## PHASE 1: COMMIT



## PHASE 2: EXPOSE

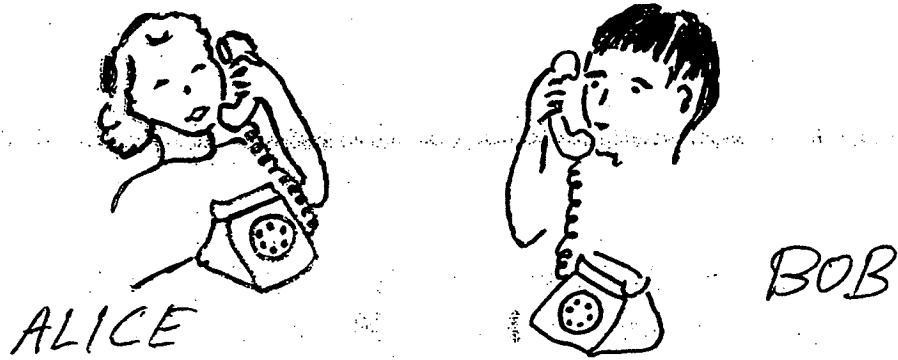



THM:  → SIMULTANEITY




~~WHAT DID YOU PICK?~~

# PUBLIC LOTTERY (ON THE PHONE)



ALICE: IF  I GET THE CAR  
(OTHERWISE YOU DO)

BOB: CAME OUT , YOU LOST!

THM:  → SYMMETRY BREAKING  
(EXTENDABLE TO MANY PLAYERS)


# IDENTIFICATION - PASSWORD

PUBLIC PSWD  
FILE

NAME	f(PSWD)
ALICE	P <sub>ALICE</sub>
AVI	P <sub>AVI</sub> = f(EINAT)
BOB	P <sub>BOB</sub>



COMPUTER CHECKS  $f(x) = P_{AVI}?$ , ERASES X.

THM:  → IDENTIFICATION

PROBLEM: REPEATED USE!

COMPUTER SHOULD CHECK IF I KNOW X  
SUCH THAT  $f(x) = P_{AVI}$  WITHOUT GETTING X

- ZERO-KNOWLEDGE PROOF 1. CONVINCING  
2. REVEALS NO INFORMATION

COPYRIGHTS

~~FERMAT'S LAST THEM~~

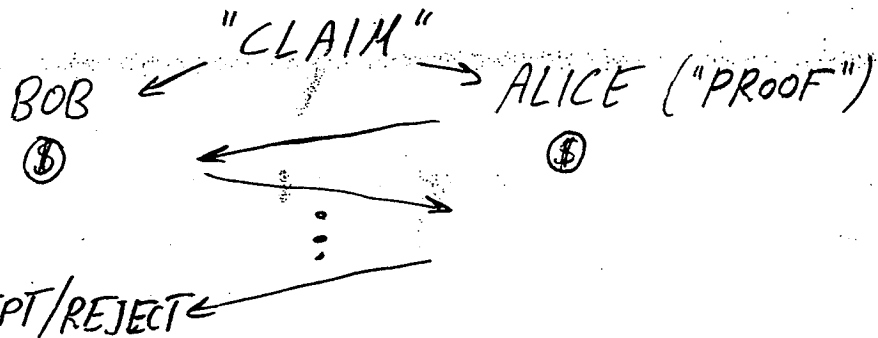
ALICE: I CAN PROVE THE RIEMANN HYPOTHESIS

BOB: IMPOSSIBLE! WHAT IS THE PROOF?

ALICE: LEMMA... PROOF... LEMMA... PROOF... QED.

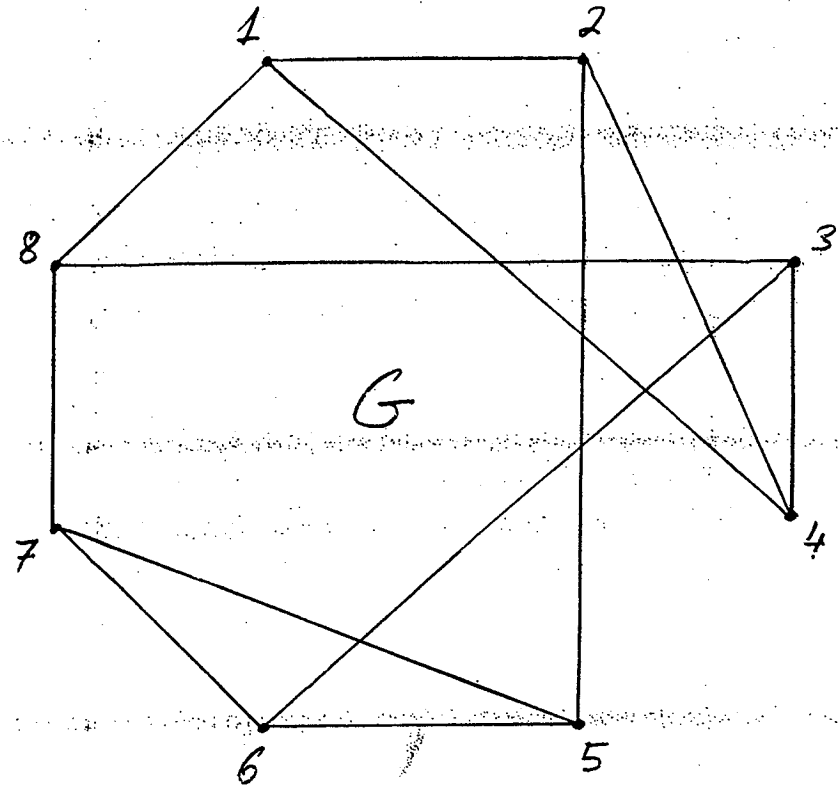
BOB: AMAZING!! I'LL RECOMMEND TENURE

ZERO-KNOWLEDGE PROOF



"CLAIM" FALSE → BOB REJECTS  
 "CLAIM" TRUE → BOB ACCEPTS  
 • BOB LEARNS NOTHING } WITH HIGH PROBABILITY

# THE HAMILTON CYCLE PROBLEM

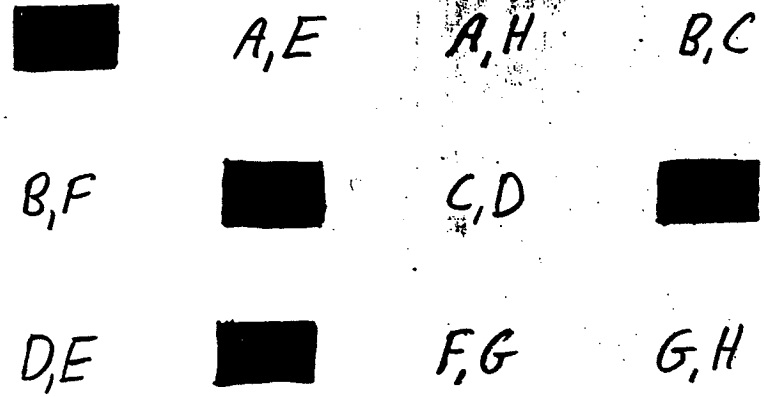
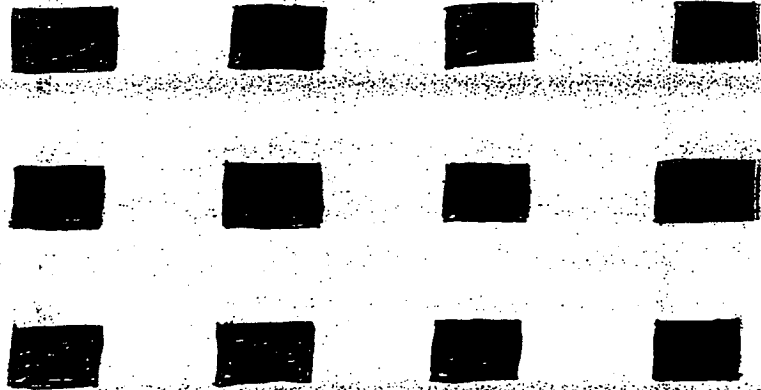


HAMILTON CYCLE: A CLOSED WALK VISITING EVERY VERTEX EXACTLY ONCE

CLAIM: G CONTAINS A HAMILTON CYCLE

# IMPLEMENTATION WITH ENVELOPES

22



A B C D E F G H



# IMPLEMENTATION WITH ENVELOPES

A,D    A,E    A,H    B,C

B,F    B,H    C,D    C,F

D,E    E,G    F,G    G,H

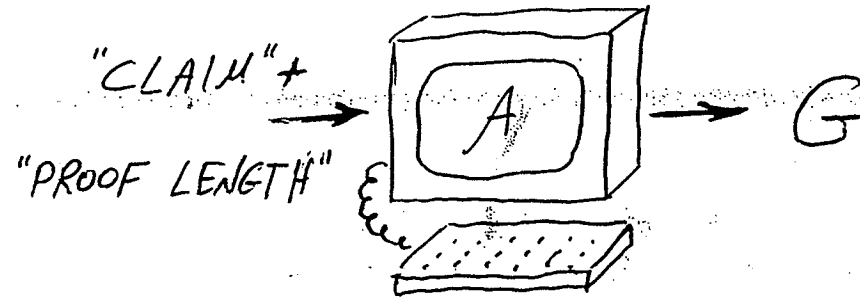
A	B	C	D	E	F	G	H
1	7	5	2	4	6	3	8

# WHY IS IT A ZERO-KNOWLEDGE PROOF?

- EXPOSED INFORMATION IS USELESS
- $G$  HAMILTONIAN  $\rightarrow$  ALWAYS CONVINCING
- $G$  NON-HAMILTONIAN  $\rightarrow \text{Prob}[\text{FAILURE}] \geq \frac{1}{2}$   
 $\text{Prob}[20 \text{ EXPERIMENTS CONVINCED}] \leq 2^{-20} < 10^{-6}$

# WHAT DOES IT HAVE TO DO WITH RIEMANN HYPOTHESIS

THM:  $\exists$  AN EFFICIENT ALGORITHM  $A$



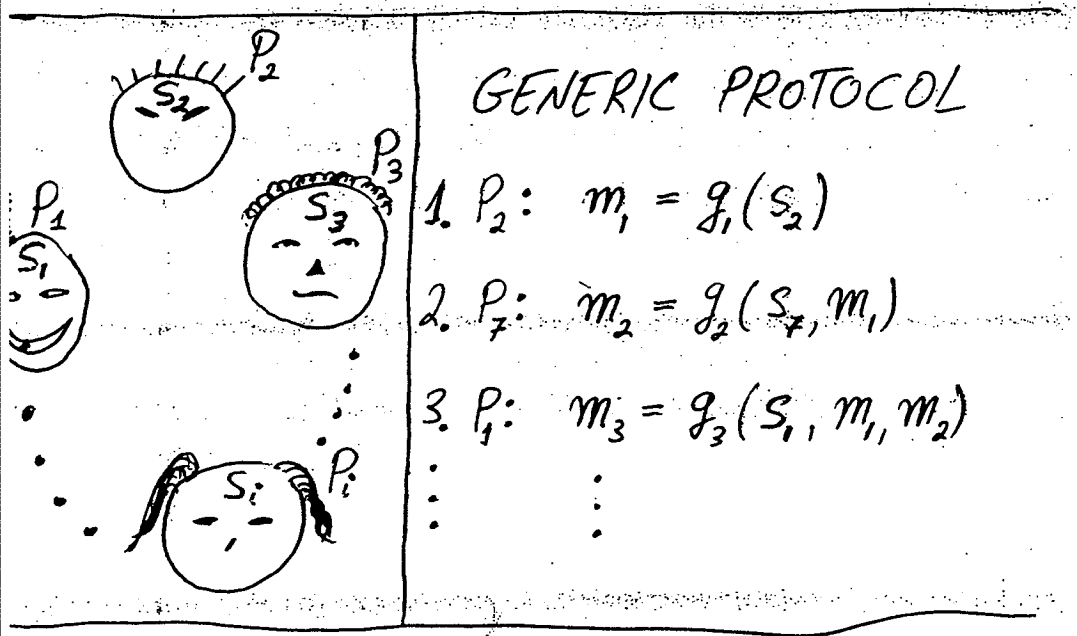
LENGTH =  $l$   $\longleftrightarrow$   $l^2$  NODES

"CLAIM" TRUE  $\longleftrightarrow$   $G$  HAMILTONIAN

"PROOF"  $\longleftrightarrow$  HAM. CYCLE IN  $G$

THM: SHORT PROOF  $\Rightarrow$  EFFICIENT ZK-PROOF

THM:   $\rightarrow$  FAULT TOLERANT PROTOCOLS



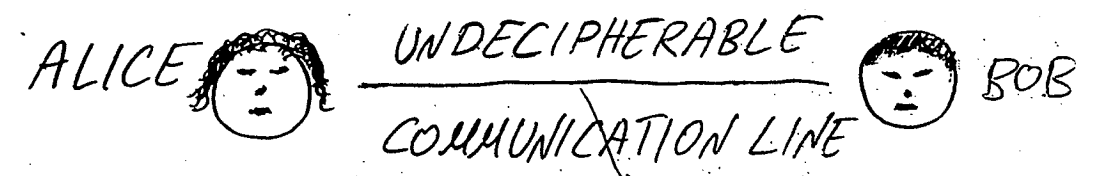
$\exists$ : EFFICIENT,  $m_i$  PUBLIC KNOWLEDGE


PROBLEM: DID e.g.  $P_i$  CHEAT IN STEP 3?

DOES  $m_3 = g_3(S_i, m_1, m_2)$ ?

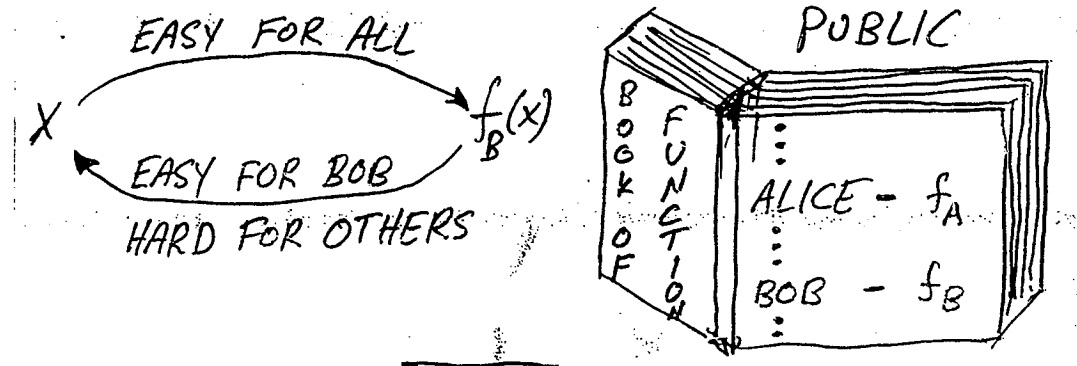
SOLUTION: THE CLAIM " $m_3 = g_3(S_i, m_1, m_2)$ " HAS A SHORT PROOF!  $P_i$  WILL PROVE IT IN ZK.


# PUBLIC KEY ENCRYPTION



LISTENS, DOESN'T UNDERSTAND  EAVES-DROPPER  
EVEN IF ALICE & BOB NEVER MET BEFORE


TRAP-DOOR FUNCTION (PERSONAL ENVELOPE,



NEW AXIOM:  $\exists$   PERSONAL

DIGITAL SIGNATURE BOB SIGNS DOCUMENT  $m$

- $(m, y) : f_B(y) = m$
1. EASY TO CHECK
  2. HARD TO FORGE

THM:  → OBLIVIOUS TRANSFER

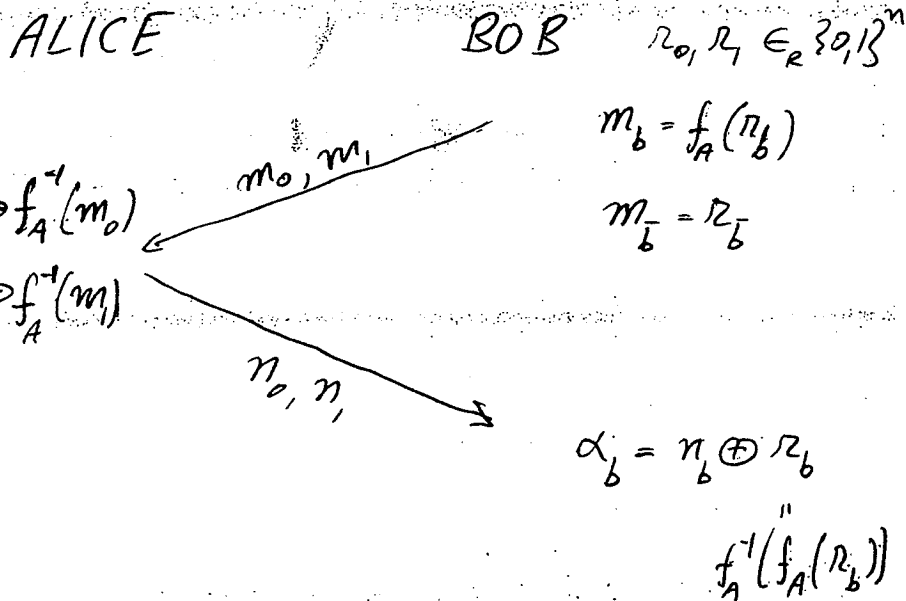
ALICE                      BOB                      (HONEST)

$\alpha_0, \alpha_1 \in \{0,1\}^m$                        $b \in \{0,1\}$

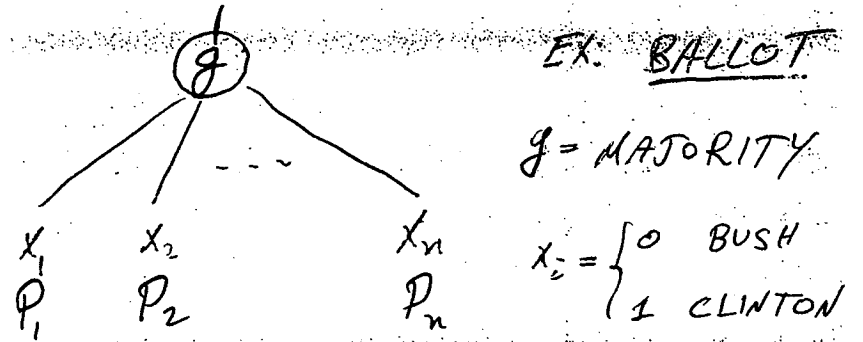
AT THE END OF THE PROTOCOL

- 1) BOB KNOWS  $\alpha_b$ , BUT NOTHING ON  $\alpha_{\bar{b}}$
- 2) ALICE HAS NO IDEA WHAT  $b$  IS

PROTOCOL ( $f_A$  1-1)

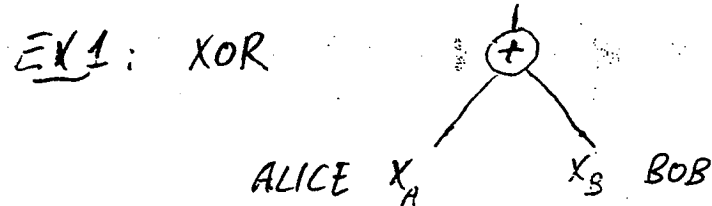


## COMPUTING FUNCTIONS ON SECRET INPUTS



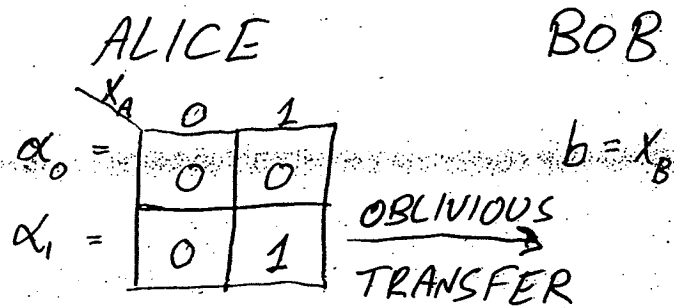
\* PLAYERS ARE HONEST

- (1) ALL PLAYERS LEARN  $g(x_1, x_2, \dots, x_n)$
- (2) NO SUBSET LEARNS ANYTHING MORE

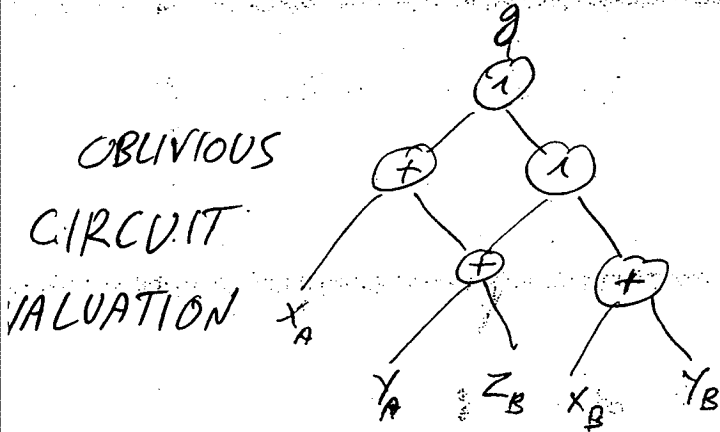




# "AND" PROTOCOL



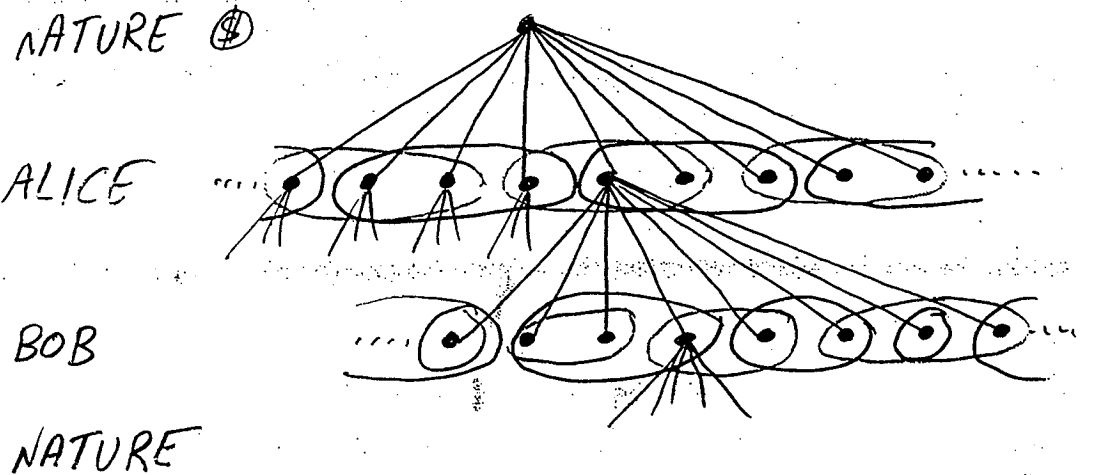
ANY (EFFICIENT) FUNCTION  $g$



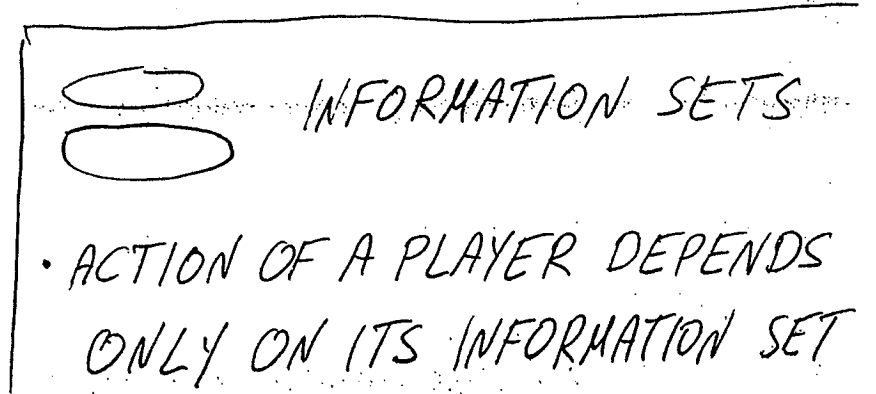
- MANY PLAYERS
- SECRET SHARING
- COMPUTING WITH SHARES

THM: PERSONAL  $\rightarrow$  EVERY "GAME", WITH ANY SECRECY REQUIREMENTS, CAN BE "IMPLEMENTED"

GAME THEORY: DESCRIPTION OF PARTIAL INFORMATION GAMES IN EXTENSIVE FORM



ALICE  
⋮  
BOB



# COMPLETENESS THEOREM

EVERY GAME WITH  $n$  PLAYERS

$s$  LISTENERS

$t$  FAULTS

CAN BE IMPLEMENTED IF

- ★ 1. PLAYERS COMPUTATIONALLY LIMITED
  - 2. TRAP-DOOR FUNCTIONS EXIST
  - 3.  $s \leq n$  ,  $t < \frac{n}{2}$
- 

★ 1.  $P_i \xrightarrow[\text{LINE}]{\text{SECURE}} P_j \quad \forall i, j$

2.  $s < \frac{n}{2}$  ,  $t < \frac{n}{3}$



**Lecturer: Allan Borodin**

**Title: Trade Offs between Time and Space**

**Material: Slides**

Department of Computer Science  
University of Toronto  
Toronto, Ontario M5S 1A4  
Canada

# TIME-SPACE TRADEOFFS

ALLAN BORODIN

8

## COMPLEXITY THEORY -

PAST, PRESENT  
PRESENT  
FUTURE

WHAT IS ULTIMATE  
GOAL OF COMPLEXITY THEORY?

UNDERSTAND "INTRINSIC COMPLEXITY  
OF FUNDAMENTAL (CLASSES OF)  
COMPUTATIONAL PROBLEMS

TO DO SO:

- A) UNDERSTAND MODELS AND  
COMPLEXITY RELATIONSHIPS,
- B) PROVIDE "WELL STRUCTURED"  
ALGORITHMS
- C) PROVIDE MATCHING LOWER  
BOUNDS FOR "COMPLETELY  
GENERAL MODELS"

# SOME COMPLEXITY CLASSES

$$NC^1 (\text{LOG DEPTH}) \subseteq DLOG (\text{LOG SPACE})$$

$$\subseteq RLOG$$

$$\subseteq NLOG$$

$$\subseteq SAC^1$$

$$\subseteq AC^1$$

$$\subseteq NC^2$$

$$\vdots$$

$$\subseteq NC$$

$$\vdots$$

$$\subseteq P \subseteq NP \subseteq UZ^k$$

$$\subseteq PSPACE \subseteq EXPTIME$$

$$\subseteq EXSPACE \subseteq \dots$$

"LOW LEVEL" COMPLEXITY

DET  $\in NC^2$

# THE LOWEST? COMPLEXITY

BARRIER:

PROVE THAT F NOT COMPUTABLE IN SIMULTANEOUS

LOG SPACE, LINEAR TIME,

FOR F A DECISION PROBLEM

IN NP.

# COMPLEXITY BARRIERS

FOR AN "EXPLICITLY DEFINED" (BOOLEAN DECISION) PROBLEM, PROVE THAT F IS HARD RELATIVE TO A "GENERAL MODEL" AND BASIC COMPLEXITY MEASURE.

## ROBUST COMPLEXITY ISSUES

$P \in NP$ ,  $P \in DLOG?$

$P \in \text{Polylogspace?}$

$P \in \text{Polylogspace-Polytime (SIMULTANEOUS?)}$   
 $= SC$

## "FINE TUNED" COMPLEXITY ISSUES

### CHOICE OF MODEL IS ESSENTIAL

WHAT FUNCTIONS CAN BE  
COMPUTED IN "LINEAR TIME"?  
(MULTIPLICATION?)  
OF INTEGERS

WHAT FUNCTIONS COMPUTABLE IN  
(SIMULTANEOUS) LOG 'PARALLEL  
TIME', LINEAR OPERATIONS?

WHAT IS COMPUTABLE IN  
(SIMULTANEOUS) LOG SPACE,  
LINEAR TIME?

COHRA (64) - {MURPHY} 1-TAP TO

BOOLEAN

LOW LEVEL:  $f \in NP$

32

BOOLEAN CIRCUIT (ST LINE PROGRAM)  
OVER  $\wedge, \vee, \neg$

- (i) SIZE  $\approx$  SEQUENTIAL TIME
- (ii) DEPTH  $\approx$  PARALLEL TIME

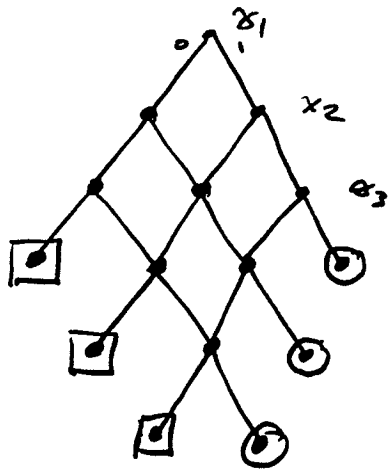
BOOLEAN BRANCHING PROGRAM  
(DEF: BY EXAMPLE)

- (ii)  $\log_2(\text{SIZE}) \approx$  SPACE

### STRUCTURED MODELS

1. ALGEBRAIC ST. LINE PROGRAMS  
NETWORKS  
ALG. COMPUTATION TREES  
BSS MODEL
2. COMPARISON BASED BRANCHING PROGRAMS
3. GRAPH TRAVERSAL MODELS

BOOLEAN B.P. FOR MAJORITY  $(x_1, \dots, x_3)$



○ ACCEPT  
 □ REJECT

ROBUST TIME-SPACE QUESTION

IS STCON IN SC?

$O(n + \epsilon)$  TIME,  $O(n \log n)$  SPACE

DEPTH FIRST SEARCH

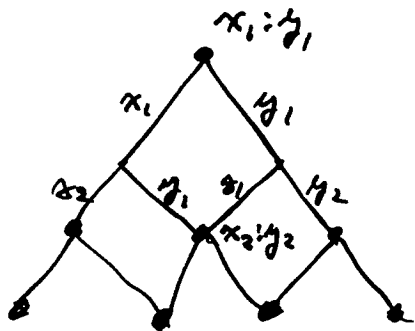
$O(\log^2 n)$  SPACE,  $n^{O(\log n)}$  TIME

SAVITCH (70)

$O(n / 2^{\sqrt{\log n}})$  SPACE,  $n^{O(1)}$  TIME

BARNES, BUSS, RUZZO, SCHUBER (72)

COMPARISON B.P. FOR MERGING



$\log_2(\text{size}) = \text{"SPACE"}$

DEPTH = "SEQ. TIME"

NON UNIFORM MODELS!

FINE TUNED QUESTION

IS ED (ELEMENT DISTINCTNESS)

IN LOG SPACE, 'LINEAR TIME' QUASI LINEAR?

$x = (x_1, \dots, x_n)$   $N = n / 2 \log n$

$0 \leq x_i \leq n^2$   $|x_i| = 2 \log n$

$O(\log n)$  SPACE,  $n^2 / \log n$  TIME

$O(n)$  SPACE,  $n \log n$  TIME (BY SORTING) <sup>83</sup>



# STATE OF THE (BOOLEAN) ART

1. FIND  $f \in NP$  :  $f \notin$  POLY LOG SPACE  
 = POLY LOG PARALLEL TIME

2. FIND  $f \in NP$  :  
 $f \notin$  LINEAR CIRCUIT SIZE  
 $f \notin$  LOG CIRCUIT DEPTH  
 $f \notin$  LOG SPACE (= POLY B.P. SIZE)

3. FIND  $f \in NP$  :  
 $f \notin$  LINEAR CIRCUIT SIZE,  
 LOG DEPTH SIMULTANEOUSLY  
 •  $f \notin$  LINEAR B.P. TIME, LOG SPACE  
 SIMULTANEOUSLY

## ALGEBRAIC MODEL (ST LINE PROGRAM)

SPACE = # REGISTERS (NON INPUT)

TIME = # STEPS

REMARKS? : SPACE & # OPERATIONS. (LOGS?)  
 + LOG (# STEPS)

WHY SHOULD BRICS, DENMARK  
 OR ANYONE CARE?

34

STCON DIRECTLY RELATED  
 TO  $DET(A) \in_{NE} A^n$

$$A^*(STCON) \leq A^n$$

$$A^* \leq_{\text{NON UNIFORM LOG SPACE}} A^n \quad \text{WIGDER 94}$$

GRAPHS, MATRICES CAN BE LARGE  
 (ALMOST SPACE) AND ONLY  
 IMPLICITLY REPRESENTED. THAT  
 IS, SMALL SPACE ALGORITHMS  
 IN COMPUTER ALGEBRA, AI, ETC  
 ARE IMPORTANT.

WHY CARE ABOUT E.D.?

# A FEW MOTIVATING RESULTS

I NOTHING IS OBVIOUS

FOR EXAMPLE, "OBVIOUSLY" (<sup>BMP</sup>83)  
CAN'T COMPUTE MAJORITY  
IN CONSTANT WIDTH (= CONSTANT  
AUX. <sup>BOOLEAN</sup> REGISTERS) AND POLY TIME

BARRINGTON (86)

$NC^1 = \text{WIDTH } S, \text{ POLY SIZE B.P.}$

BEN-OR, CLEVE (92)

ARITH  $NC^1 = 3 \text{ REGISTERS, POLY LENGTH S.L.P.}$

THUS DET  $\in NC^1$  (POLY FORMULA

SIZE) IFF DOABLE WIDTH

3 REGISTERS, POLY LENGTH S.L.P.

MANY IMPORTANT RESULTS ON THE  
ST LINE = CIRCUIT PEBBLING MODEL

HOPCROFT, PAUL, VALIANT (77)

EVERY  $N$  NODE DAG CAN BE PEBBLED  
WITH  $O(N/\lg N)$  PEBBLES (BUT METHOD  
USES  $2^S$  TIME -  $S = \#$  PEBBLES

PAUL, TARJAN, CELONI (77)

$\exists N$  NODE DAG, REQUIRING  $\Omega(N/\lg N)$  PEBBLES

PAUL, TARJAN (78), LEUNG, TARJAN (82)

$\forall N$  NODE DAG,  $T = 2^{O(N/S)}$

$\exists N$  NODE DAG,  $T = 2^{\Omega(N/S)}$

CON:  $S = O(N/\lg N)$  IS A "THRESHOLD"  
FOR POLY TIME

TOMPA (82)

FOR THE  $n \times n$  TRANSITIVE CLOSURE  $A^*$   
ANY CIRCUIT THAT WORKS BY REPEATED  
SQUARING (EG. SAVITEN'S METHOD) REQUIRES  
NON POLYNOMIAL TIME WHEN USING  
 $O(n)$  SPACE.

BUT NOTE BARNES, BULL, RUZZO, SCHIEBEL

II BOOLEAN / ARITHMETIC ST LINE PROG  
(CIRCUIT PEBBLING MODEL)

GRIGORIEV (76)

MATRIX MULT  $TS = \Omega(n^2)$

POLY MULT  $TS = \Omega(n^2)$

TOHRA (80)

CONVOLUTION  $TS = \Omega(n^2)$

DFT  $TS = \Omega(n^2)$

SA'GA'

MATRIX INVERSION  $TS = \Omega(n^4)$

III COMPARISON B.P. MODEL

(NOTE: TOHRA (80) SHOWS THAT ANY ORDERING NETWORK (SELECT, COMPARE, BOOLEAN OPS) HAS  $T.S = \Omega(n^2)$  FOR MERGING

BOROHO, FISCHER, KIRKPATRICK, LYNCH, TOMPA (79)  
SORTING  $T.S = \Omega(n^2)$  35

BOROHO, FICH, MEYER AUF DER HEIDE,  
VITAL, WIGGUSON (87)

ELEMENT DISTANCE  $T^2S = \Omega(n^2)$

YAO (88) ED  $TS = \Omega(n^2 - \epsilon)$

IV BOOLEAN / R-WAY B.P. MODEL

BOROHO, COOK (82); BEAM (91)

SORTING  $TS = \Omega(n^2)$

YESHA (84)

MATRIX MULT  $TS = \Omega(n^2)$

FOR FIELD F WITH  $|F| = O(n)$

DFT (n)  $n = p-1$ , p PRIME

$TS = \Omega(n^2)$

ABRAHAMSON (91)

MATRIX MULT  $D \in F$  (ARF FIELD)

$T^2S = \Omega(n^2 \log n)$

DFT  $T^2S = \Omega(n^2)$  AM F

ABC  $T^2S = \Omega(n^2 \log n)$

$$A'' \quad TS = \Omega(n^4)$$

(AGAIN, CONTRAST WITH

$$TS = \Theta(n^2) \text{ FOR } A \cdot B)$$

RAMSOU, PISAN, TIWARI (90)

ANY UNIVERSAL HASH FUNCTION

$$x + y \cdot z \text{ IN } O(2^n) \quad TS = \Omega(n^2)$$

## V USTCON, STCON

### JAG (AND JAG) RESULTS

EDMONDS (93) USTCON

$$P = O(\log n / \log \log n)$$

$$T = \Omega(n \cdot 2^{\Omega(\log n / \log \log n)})$$

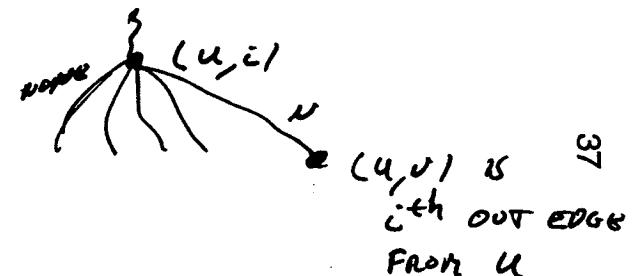
BARNES + RUBBO STCON

$$TS = \Omega(n^2 / \log n)$$

$$\text{POOR } TS = \Omega(n^2)$$

WHAT TO DO NEXT?

1. DECLARE VICTORY (= GIVE UP) AND START WORKING ON "HARD" COMPUTATIONAL PROBLEMS
2. OBTAIN  $TS = \Omega(n^2)$  FOR USTCON JAG/KYDAG MODEL
3. EXTEND THE JAG MODEL (DESIGN YOUR OWN EXTENSION)!!  
 ALLOW RANDOM JUMPING  
 TWO WAY EDGE LABELLING  
 "SLIDE" PEBBLE FROM  $i$  TO  $i+1$   
 "STRONG" JUMP TO NODE  $i = B.P.$   
 (ADJACENCY LIST INPUT)



OPEN: COMPUTE MAX IN-DEGREE (MODE)

4. WORK ON RESTRICTED B.P. MODELS.

OBLIVIOUS

BOUNDED WIDTH

READ ONCE, OBDD<sub>k</sub>

READ K-TIMES

WHAT CAN BE SAID

ABOUT THESE PROBLEMS

WHEN  $S = \# \text{PEBBLES} = O(1)$ ?

5. PRODUCE SURPRISING UPPER BOUNDS

SHOWING NEGATIVE RESULTS

NOT POSSIBLE

BOUNDED WIDTH BOOLEAN B.P.

OBLIVIOUS BOOLEAN B.P.

ALON, MAASS

BABAI, NISAN, SZEDRY

6. RETURN TO ALGEBRAIC CIRCUIT MODEL

ARE THERE A TS & P(n)

RESULT FOR A SINGLE

POLY (n VARIABLES, DEGREE = n)

$$\prod_{i,j} (x_i - y_j)$$

STRASSEN

BETTER

$\Omega(n \log n)$  LOWER BOUNDS

ANY SPACE

CONSIDER  $\prod_{i,j} (x_i - y_j)$

DET(A)

MEMORY EFFICIENT PAIR-STRASSEN?  
"INV & DET"

GAJEVANSKI: "MEMORY EFFICIENT"

EACH STRASSEN

SPACE S, TIME T

$$\left\{ \frac{O(1)}{O(n)}, \dots, \frac{O(1)}{O(n)} \right\}$$

SPACE  $(S+n) \log T$

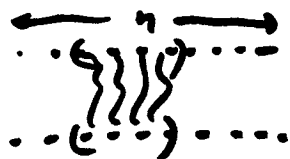
TIME  $T \log T$

(ARITH) CIRCUIT MODEL

CONVOLUTION  $T.S = \Omega(n^2)$

TOHFA'S PROOF FOLLOWING  
ORIGONOV; VALIANT (76)

VALIANT: ANY BILINEAR (WLG)  
CIRCUIT FOR CONVOLUTING TWO  
DEGREE  $n$  POLYS OVER  $F$  (CHAR  $F \neq 0$ )  
CONTAINS AN  $n$  SUPERCONCENTRATED



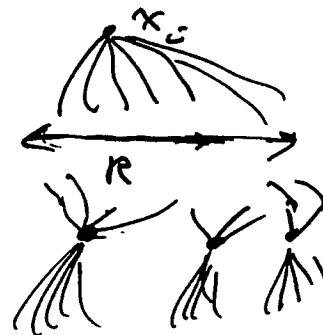
TOHFA: PEBBLING AN  $n$ -SUPERCONC.

WITH  $S$  PEBBLES REQUIRES

$$T_{in} \geq n(n-s)/(s+1) + s$$

"FOR EVERY  $s+1$  OUTPUTS, (PROGRESS)  
 $n-s$  INPUTS MUST BE  
(RE)PEBBLED"

R-WAY B.P. MODEL



BORODIN, COOK (82); BEAME (91)

ANY  $R (= n)$  WAY B.P. FOR  
OUTPUTTING THE UNIQUE ELEMENTS  
IN  $\{x_1, \dots, x_n\}$  WHERE  $x_i \in [1, n]$   
REQUIRES  $T.S = \Omega(n^2)$ . SORTING  
HAS SAME BOUND BY A REDUCTION.

SKETCH OF PROOF:

PROGRESS ARGUMENT

HERE PROGRESS = # OUTPUTS

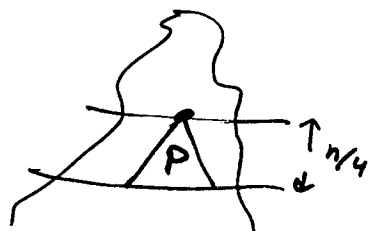
CHOOSE  $n$  ELEMENTS UNIFORMLY  
AT RANDOM FROM  $\{1, 2, \dots, n\}$ ;  $\vec{x} \in U_n^n$

L1: PROB [ $\vec{x}$  contains  $\geq n/2e$   
UNIQUE ELEMENTS]  $\geq \frac{1}{(2e-1)}$

L2: LET  $h \leq n/4$

PROB [A SUBPROGRAM  $P$  OF LENGTH  $h$   
OUTPUTS  $\geq m$  UNIQUE ELEMENTS  
CORRECTLY ON  $\vec{x}$ ]  
 $\leq e^{-m/2}$

L3: TO OUTPUT  $\geq n/2e$  UNIQUE ELEMENTS  
ON  $\vec{x}$ , MUST OUTPUT  $\geq n/2e / \frac{4T}{h}$



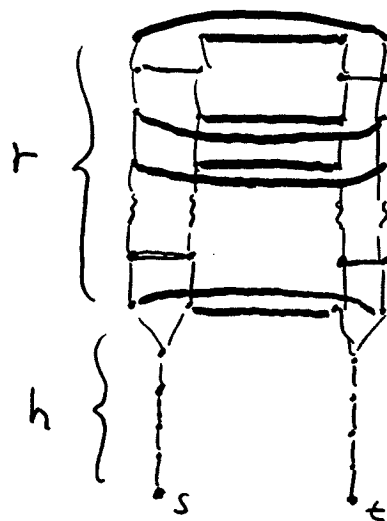
$$= \frac{n^2}{8eT} \text{ ON SOME } P$$

MUST HAVE  $\frac{2^S}{\#P} \cdot \underbrace{e^{-n^2/16eT}}_{\text{PROB ON ONE } P} > \frac{1}{2e-1}$

$$\Rightarrow S = \Omega(n^2/T)$$

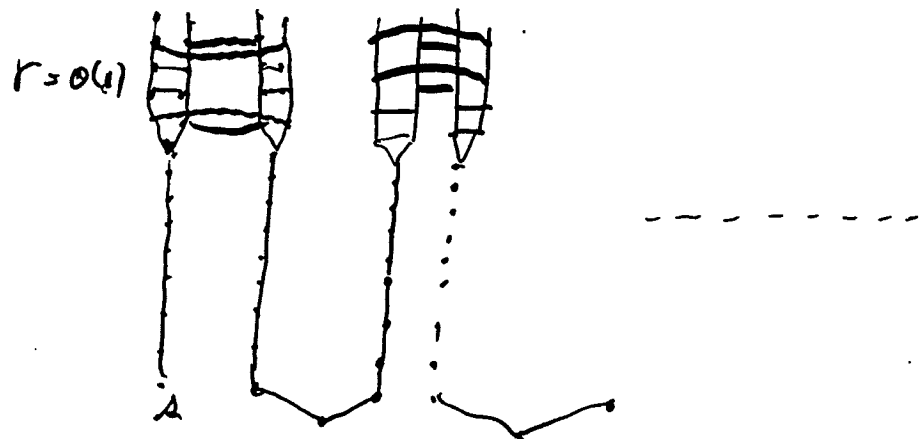
SKETCH OF LINEAR ALGEBRA  
BASED USTCON PROOFS

FLY SWATTER PAIR GRAPH (BRT, BBRT)



$T = h$   
 $\alpha \in \{0, 1\}^T$   
DESCRIBES GRAPH  
 $G_\alpha$ .  $G_\alpha$  HAS  $s-t$   
PATH IFF  
 $\alpha \neq 0^T$

RECURSIVE LINE OF FLY SWATTERS (EDMONDS)



STCON LOWER BOUNDS

BARNES, EDMONDS (93)

$$JAG \quad T.S = \Omega(n^2 / \log n)$$

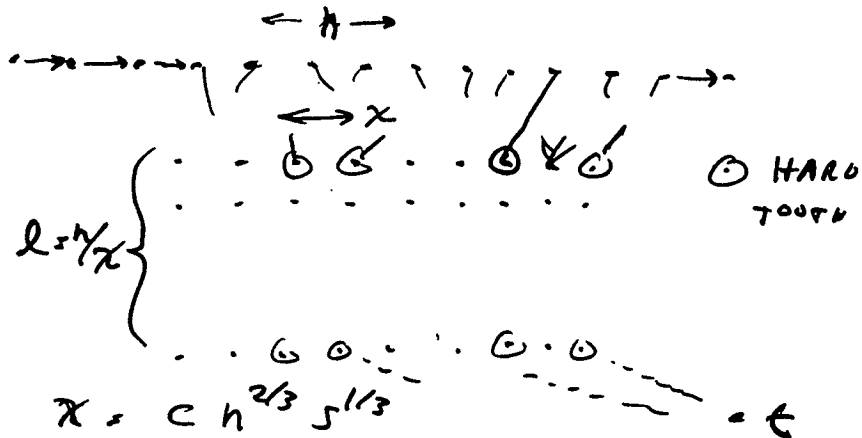
$$\text{OUTDEGREE} = 2$$

EDMONDS

NJAG (PROBABILISTIC)  
FOR NON CONNECTIVITY

$$TS^{1/3} = \Omega(n^{4/3})$$

SKETCH (COMB GRAPH)



PROGRESS = # OF HARD TEETH  
TRAVERSED





**Lecturer: Adi Shamir**

**Title: Open Problems in Cryptocomplexity**

No material, since blackboard was used.

Weizmann Institute of Science  
Rehovot, Israel  
E-mail: shamir@wisdom.weizmann.ac.il



**Lecturer: Dexter Kozen**

**Title: Efficient Average-Case Algorithms  
for the Modular Group**

**Material:** Jin-Yi Cai, Wolfgang H. Fuchs, Dexter Kozen, and  
Zicheng Liu: *Efficient Average-Case Algorithms for  
the Modular Group*

Department of Computer Science  
Cornell University  
Ithaca, New York 14850  
USA  
E-mail: kozen@cs.cornell.edu

# Efficient Average-Case Algorithms for the Modular Group\*

Jin-Yi Cai  
SUNY Buffalo  
cai@cs.buffalo.edu

Wolfgang H. Fuchs  
Cornell University  
fuchs@math.cornell.edu

Dexter Kozen  
Cornell University  
kozen@cs.cornell.edu

Zicheng Liu  
Princeton University  
zl@cs.princeton.edu

## Abstract

The modular group occupies a central position in many branches of mathematical sciences. In this paper we give average polynomial-time algorithms for the unbounded and bounded membership problems for finitely generated subgroups of the modular group. The latter result affirms a conjecture of Gurevich [5].

## 1 Introduction

### 1.1 The Modular Group

The modular group  $\Gamma$  is a remarkable mathematical object. It has several equivalent characterizations:

- (i)  $SL_2(\mathbb{Z})/\pm I$ , the quotient of the group  $SL_2(\mathbb{Z})$  of  $2 \times 2$  integer matrices with determinant 1 modulo its central subgroup  $\{\pm I\}$ ;
- (ii) the group of complex fractional linear transformations

$$z \mapsto \frac{az + b}{cz + d}$$

with integer coefficients satisfying  $ad - bc = 1$ ;

- (iii) the free product of cyclic groups of order 2 and 3; i.e., the group presented by generators  $R, S$  and relations  $R^2 \equiv S^3 \equiv 1$ ;
- (iv) the group of automorphisms of a certain regular tessellation of the hyperbolic plane (Figure 1);

\*Proc. 35th IEEE Symp. Foundations of Computer Science, Nov. 1994, to appear.

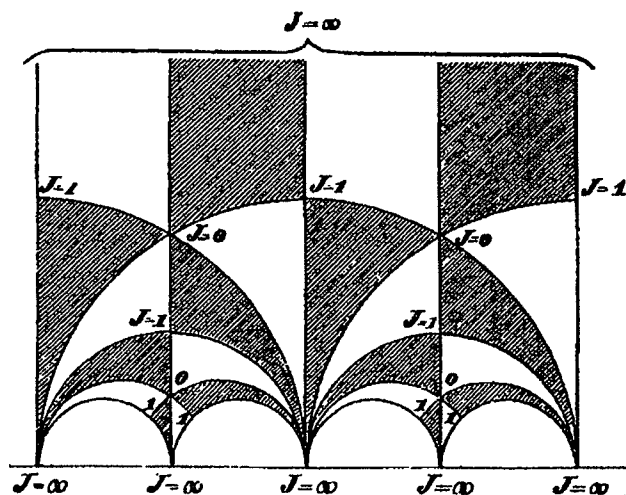


Figure 1: A tessellation of the hyperbolic plane<sup>1</sup>

- (v) the group of sense-preserving automorphisms of the undirected cubic plane tree (Figure 2).

The modular group is intimately connected with the theory of elliptic curves, modular functions and modular forms, hyperbolic geometry, and number theory [1].

For instance, it is known that elliptic curves can be uniformly parametrized by the Weierstrass  $\wp$  function. This function is invariant under the action of a group of transformations of the plane isomorphic to  $\mathbb{Z} \times \mathbb{Z}$ . This action gives rise to a discrete Euclidean tessellation of the plane. In contrast, a *hyperbolic uniformization* is a uniform parametrization of the elliptic curve by functions that are invariant under the

<sup>1</sup>Reproduced from Klein (1879) [9].

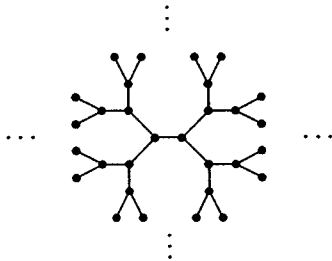


Figure 2: The undirected cubic plane tree

modular group  $\Gamma$  or some subgroup of it. Here the so-called *congruence subgroups* of  $\Gamma$  play a dominant role. The *Taniyama-Weil conjecture* states that all elliptic curves with rational coefficients admit such a uniformization by functions invariant under some congruence subgroup of  $\Gamma$ . It is known that a counterexample to Fermat's Last Theorem would invalidate this conjecture. While some difficulties remain, it appears that Andrew Wiles has made a significant advance towards resolving this conjecture.

The modular group is also deeply connected with many algorithmic issues. For instance, the ordinary Euclidean integer gcd algorithm can be understood in terms of a basis reduction algorithm on  $2 \times 2$  integer matrices, where the reducing operations are elements of the modular group in the form (i) above. This connection allows us to apply a result of Yao and Knuth [17] concerning the integer gcd algorithm in our analysis.

Some algorithms of Schönhage [14, 15] can be best understood in light of the modular group.

A recent paper by Yap [18] is concerned with the modular group and its connection with lattice basis reduction algorithms. The basis reduction algorithms of Lenstra, Lenstra and Lovász [10] have had considerable impact on algorithm design and analysis, ranging from integer programming to polynomial factorization.

Finally, we note that the modular group has found applications in computational learning theory [3].

## 1.2 Subgroup Membership

In this paper we consider four natural decision problems for the modular group  $\Gamma$ :

**The Unbounded Subgroup Membership Problem** Given a finite subset  $\mathcal{S} \subseteq \Gamma$  and an element  $x \in \Gamma$ , is  $x$  contained in the subgroup of  $\Gamma$  generated by  $\mathcal{S}$ ?

**The Bounded Subgroup Membership Problem** Given a finite subset  $\mathcal{S} \subseteq \Gamma$ , an element  $x \in \Gamma$ , and  $n \geq 0$  in unary, can  $x$  be expressed as a product of at most  $n$  elements of  $\mathcal{S}$  and their inverses (repetitions allowed)?

**The Unbounded Submonoid Membership Problem** Given a finite subset  $\mathcal{S} \subseteq \Gamma$  and an element  $x \in \Gamma$ , is  $x$  contained in the submonoid of  $\Gamma$  generated by  $\mathcal{S}$ ?

**The Bounded Submonoid Membership Problem** Given a finite subset  $\mathcal{S} \subseteq \Gamma$ , an element  $x \in \Gamma$ , and  $n \geq 0$  in unary, can  $x$  be expressed as a product of at most  $n$  elements of  $\mathcal{S}$  (repetitions allowed)?

The only difference between the subgroup and submonoid membership problems is that in the subgroup membership problems, inverses are allowed. The subgroup membership problems reduce to the submonoid membership problems by simply including the inverses in the set  $\mathcal{S}$ .

We assume that these problems are presented in the form (i) of §1.1; that is, as  $2 \times 2$  integer matrices with entries written in binary.

## 1.3 Average-Case Complexity

The study of *NP*-hard problems that are hard on average was initiated by Levin [11] and generated considerable subsequent interest [2, 6, 5, 8, 16].

Suppose the inputs to an algorithm occur randomly according to a distribution with the property that the probability that the input size is  $n$  is either zero or at least  $n^{-k}$  for some fixed  $k$ . Such a distribution is called *regular*. (For definiteness, Gurevich [5] takes the probability of the event  $|x| = n$  to be proportional to  $n^{-1}(\log n)^{-2}$ , but any regular distribution will do.)

A deterministic algorithm runs in *polynomial time on average* if there exists an  $\epsilon > 0$  such that

$$\sum_x \frac{T(x)^\epsilon}{|x|} \Pr(x) < \infty,$$

where  $T(x)$  is the running time of the algorithm on input  $x$ . For regular distributions, it suffices to show that there exists an  $\epsilon > 0$  such that for all  $n$ ,

$$\sum_{|x|=n} T(x)^\epsilon \cdot \Pr_n(x) \leq n^{O(1)},$$

where  $\Pr_n(x)$  denotes the conditional probability that  $x$  occurs given that the size of the input instance is  $n$  [6, 5].

Gurevich [5] applied this notion to several algebraic problems. In particular, he showed that certain matrix decomposition problems involving the modular group are hard on average.

Gurevich defined the bounded subgroup membership problem stated in §1.2 and conjectured that it was polynomial time on average.

## 1.4 Main Results

In this paper we show:

**Theorem 1.1** *The bounded and unbounded membership problems for finitely generated subgroups and submonoids of the modular group can be solved in polynomial time on average.*

This affirms Gurevich's conjecture.

We do not know whether the subgroup membership problems are *NP*-hard. However, the semigroup membership problems are quite easily shown to be *NP*-hard by a straightforward encoding of the subset sum problem.

## 1.5 Overview

Our approach is to convert  $x$  and every element in  $S$  to the representation (iii) of §1.1 (*i.e.*, words in  $\{R, S\}^*$  reduced modulo the identities  $R^2 \equiv S^3 \equiv 1$ ), and work in that representation.

This will be of little use if the representation (iii) is too long or if it is hard to compute from the representation (i). It turns out that it is easy to compute, but may be exponentially long in the worst case. However, it is short on average.

Our analysis makes use of an intermediate representation (2.3), which is similar to (iii), but for which a polynomial bound on the average length is known. The lengths of minimal representations in (iii) and (2.3) are mutually proportional.

Our analysis proceeds in two steps:

- (i) In §4, we give deterministic polynomial-time algorithms in representation (iii) for the bounded and unbounded membership problems. These algorithms reduce the problems to a certain automata-theoretic reachability problem.
- (ii) In §5 we show that the process of converting an input instance from representation (i) to representation (iii) and then executing the algorithm of §4 on the resulting data gives an average-case polynomial-time algorithm. This part of the argument relies on an estimate of Yao and Knuth [17].

The same techniques also handle other related groups such as  $SL_2(\mathbb{Z})$  or the congruence subgroups of  $\Gamma$ . We do not treat these cases in this paper.

## 2 Representations of $\Gamma$

To understand this work, one must first understand the relationships among the different representations (i)–(v) of  $\Gamma$  described in §1.1. See [1, 13, 12, 4] for details.

In the representation (i), elements of  $\Gamma$  are represented as  $2 \times 2$  matrices with integer entries. The group  $\Gamma$  is generated by the matrices

$$\begin{aligned} T &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} & R &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \\ S &= TR = \begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix} \end{aligned} \quad (2.1)$$

Any two of these three matrices generate  $\Gamma$ .

These matrices correspond to the fractional linear transformations

$$T : z \mapsto z + 1 \quad R : z \mapsto -\frac{1}{z} \quad S : z \mapsto 1 - \frac{1}{z} \quad (2.2)$$

on  $\mathbb{C}$ , respectively. The matrices (2.1) represent the transformations (2.2) in homogeneous coordinates, viewing them as linear transformations on the projective complex line. This gives the relationship between the representations (i) and (ii).

Note that  $R$  is of order 2 and  $S$  is of order 3 (recall we are working modulo  $\pm I$ ). In fact  $\Gamma$  is the free product of the cyclic groups generated by  $R$  and  $S$ . This gives the relationship with representation (iii).

To see the relationship with (iv), observe that the transformations (2.2) preserve the upper half plane  $\mathbf{H}$ .  $\mathbf{H}$  can be regarded as a model of hyperbolic geometry, where geodesic lines are semicircles or lines perpendicular to the real axis. Under the appropriate metric,  $\Gamma$  is a group of isometries of  $\mathbf{H}$ . The region

$$\{z \in \mathbb{C} \mid -\frac{1}{2} < \Re z < \frac{1}{2}, |z| > 1\}$$

is a fundamental region for the action of  $\Gamma$ , and its orbit gives a tessellation of  $\mathbf{H}$ . This region corresponds to the union of the two uppermost central regions, one shaded and one not, shown in Figure 1. Several works by M. C. Escher are based on this universe.

To understand the connection to (v), we observe that the infinite undirected cubic plane tree shown in Figure 2 is embedded in Figure 1 by considering the

segment of the circle of radius 1 centered at 0 from  $e^{2\pi i/3}$  to  $e^{\pi i/3}$  as a directed edge  $E$ , then taking the orbit of this edge under the action of the group. Every element of  $\Gamma$  is uniquely identified with a directed edge produced in this way.

With this identification, observe that  $R$  reverses the direction of  $E$ ,  $T$  corresponds to a left turn out of  $E$ , and  $S = TR$  rotates about the vertex at the head of  $E$ . In any product  $X_1 \cdots X_n \in \{T, R, S\}^*$  applied in order from right to left, the destination of  $E$  can be calculated by reading the string  $X_1 \cdots X_n$  from left to right and interpreting  $T$  as “turn left”,  $R$  as “reverse direction”, and  $S$  as “rotate clockwise about the vertex before you”. We can also define  $U = ST$  (“turn right”).

The group  $\Gamma$  has the following presentation in terms of  $T$  (turn left),  $U$  (turn right), and  $R$  (reverse):

$$\begin{aligned} TRU &= URT = R \\ TRT &= U \quad URU = T \\ R^2 &= 1 \end{aligned} \quad (2.3)$$

The equations (2.3) can be applied as term rewriting rules to reduce any string in  $\{R, T, U\}^*$  to normal form  $(R + \epsilon)(T + U)^*(R + \epsilon)$ . Every element of  $\Gamma$  can be expressed uniquely as a product of this form, and the length of any expression of this form is within two of minimal among all expressions in  $\{R, T, U, R^{-1}, T^{-1}, U^{-1}\}^*$  denoting the same group element. This is a consequence of the fact that shortest paths in the graph of Figure 2 are unique. A similar statement holds for the presentation (iii); in this case, normal forms are strings in  $\{R, S\}^*$  with no occurrence of two consecutive  $R$ 's or three consecutive  $S$ 's.

The presentations (2.3) and (iii) are interderivable using the facts  $T = SR$ ,  $U = SSR$ ,  $S = TR$ . Moreover, these relations show that for any group element, the lengths of the minimal representations in  $\{R, S\}^*$  and  $\{R, T, U\}^*$  differ by at most a factor of three.

In terms of representation (i), the left and right turns are

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad U = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},$$

respectively. Note that elementary row and column operations on  $2 \times 2$  matrices (adding a row or column to the other) are effected by multiplying on the left or right by  $T$  or  $U$ . In this interpretation, the significance of the normal form  $(R + \epsilon)(T + U)^*(R + \epsilon)$  is that for any matrix, we can multiply by  $R$  on the left or right if necessary to make all entries nonnegative, and

then there is a unique sequence of column operations to bring the matrix to  $I$  while keeping entries non-negative. The same is true for row operations. This gives us an effective method for converting between the representations (i) and (2.3).

## 2.1 Integer GCD

The matrices  $T$  and  $U$  have the following significance regarding integer gcd. Let  $s(m, n)$  be the number of steps in the following *subtractive Euclidean algorithm* for finding the gcd of  $m$  and  $n$ : replace the larger number by the difference of the two numbers until both are equal. Note that  $s(m, n)$  is one less than the sum of all partial quotients in the continued fraction representation of  $m/n$ ,  $1 \leq m \leq n$ . For example,

$$\frac{7}{16} = \frac{1}{2 + \frac{1}{3 + \frac{1}{2}}}$$

and  $s(7, 16) = (2 + 3 + 2) - 1 = 6$ .

The matrices  $T$  and  $U$  correspond to the basic operations of the subtractive gcd algorithm in the sense that if  $m$  and  $n$  are relatively prime and appear in the top row of a matrix  $A \in \Gamma$ , then  $T^{-1}$  and  $U^{-1}$  applied on the right hand side effect the column operations corresponding to the steps of the subtractive gcd algorithm. It follows that the length of the unique expression in  $\{T, U\}^*$  equivalent to  $A$  is exactly  $s(m, n)$ .

## 3 Length of Representations

Gurevich showed that the size of any element  $A \in \Gamma$  in representation (2.3) is polynomial in the size of  $A$  in representation (i) on average [5, Lemma 4.2]. Our observation that minimal-length representations in (2.3) and (iii) are mutually proportional implies that the size of  $A$  in representation (iii) is also polynomial in the size of  $A$  in representation (i) on average. This result, together with the polynomial time algorithm of the next section, do not immediately imply an average polynomial-time complexity of the membership problems, since the number of input matrices is not fixed.

Gurevich's argument is based on the following estimate of Yao and Knuth:

**Lemma 3.1 (Yao and Knuth [17])**

$$\begin{aligned} \sum_{m \leq n} s(m, n) \\ = \frac{6}{\pi^2} n (\log n)^2 + O(n \log n (\log \log n)^2). \end{aligned}$$



It follows immediately that for fixed  $n$ , the average value of  $s(m, n)$ , where  $m$  is chosen uniformly at random among all positive integers less than and relatively prime to  $n$ , is at most

$$O\left(\frac{n(\log n)^2}{\varphi(n)}\right) \leq O((\log n)^2 \log \log n), \quad (3.4)$$

where  $\varphi(n)$  is the Euler totient function. The inequality (3.4) follows from the estimate  $\varphi(n) = \Omega(n/\log \log n)$  [7, Theorem 328].

Except for  $I, T$ , and  $U$ , if  $A \in \Gamma$  has nonnegative entries and maximum entry  $n$ , and if  $m$  is the other entry in the same row as  $n$ , then  $1 \leq m < n$ ,  $(m, n) = 1$ , and the rest of  $A$  is uniquely determined by the constraint on the determinant of  $A$ . Since there are four ways to choose the position of the maximal entry  $n$  in  $A$ , such matrices are in four-to-one correspondence with the pairs  $m, n$  such that  $1 \leq m < n$  and  $(m, n) = 1$ . It follows that the length of the unique expression in  $\{T, U\}^*$  corresponding to  $A \in \Gamma$  is also polynomial on average.

## 4 Deterministic Algorithms

In this section we give deterministic polynomial-time algorithms for the unbounded and bounded membership problems when the input is given in representation (iii) of §1.1, i.e. in terms of generators  $R, S$  and relations  $R^2 \equiv S^3 \equiv 1$ .

Consider the term rewriting system over strings in  $\{R, S\}^*$  consisting of reduction rules  $R^2 \rightarrow \epsilon$ ,  $S^3 \rightarrow \epsilon$ . We write  $x \rightarrow y$  if the string  $x$  reduces to the string  $y$  in zero or more steps. A string is said to be *reduced* or in *normal form* if no reduction rule applies. This system has nonoverlapping redexes (the *redexes* are  $R^2$  and  $S^3$ ), thus it follows from term rewriting theory that normal forms are unique, and  $x \equiv y$  iff  $x$  and  $y$  have a common normal form.

Suppose now we are given a set  $\mathcal{S}$  of reduced strings in  $\{R, S\}^*$ , a reduced string  $x \in \{R, S\}^*$ , and (for the bounded membership problem) an integer  $n$  in unary. Let  $\mathcal{S}^*$  denote the submonoid of  $\{R, S\}^*$  generated by  $\mathcal{S}$ . The *unbounded membership problem* is to determine whether there exists a string  $y \in \mathcal{S}^*$  such that  $y \rightarrow x$ . For the *bounded membership problem*, we require in addition that  $y \in \mathcal{S}^m$  for some  $m \leq n$ . We will give an algorithm that runs in time polynomial in  $n$  and the sum of the lengths of  $x$  and the elements of  $\mathcal{S}$ .

Note that this formulation of the problem asks for membership of  $x$  in a finitely generated submonoid of

$\Gamma$ . If we wish to determine membership in a finitely generated subgroup, we can simply include the inverses of elements of  $\mathcal{S}$ .

In a fixed reduction sequence  $x \rightarrow y$ , we say that an occurrence of a letter  $a$  in  $y$  *comes from* an occurrence of  $a$  in  $x$  if  $x = uav$  and  $y = zaw$ , where the mentioned occurrences of  $a$  in  $x$  and  $y$  are as shown, and the appropriately chosen subsequences of the reduction sequence give  $u \rightarrow z$  and  $v \rightarrow w$ . For a fixed reduction sequence  $x \rightarrow y$ , every letter of  $y$  comes from a unique letter of  $x$ . The remaining letters of  $x$  must eventually become part of a redex and disappear.

For any set  $H$  of strings, we denote by  $H/\equiv$  the set of strings  $\equiv$ -equivalent to some string in  $H$ . Thus  $\mathcal{S}^*/\equiv$  denotes the set of strings representing elements of the submonoid of  $\Gamma$  generated by  $\mathcal{S}$ . This notation is slightly nonstandard but convenient for our purposes. Our task is to find an efficient membership test for  $\mathcal{S}^*/\equiv$  for the unbounded membership problem and  $\bigcup_{m \leq n} \mathcal{S}^m/\equiv$  for the bounded membership problem.

### 4.1 An Automata-Theoretic Characterization

Let  $M$  be the finite automaton with states

$$Q = \{u \mid u \text{ is a suffix of some } x \in \mathcal{S}\},$$

start and final state  $\epsilon$  (the null string), and transitions

$$\begin{aligned} au &\xrightarrow{a} u, & a \in \{R, S\}, \\ u &\xrightarrow{\epsilon} v, & u^{-1}v \in \mathcal{S}^*/\equiv. \end{aligned}$$

We will show below that for any reduced  $x$ ,  $x \in \mathcal{S}^*/\equiv$  iff  $x$  is accepted by  $M$ . Note that  $M$  has linearly many states and the  $\epsilon$  edges are transitive. Once we construct the automaton for a given set of generators  $\mathcal{S}$ , we can test membership in  $\mathcal{S}^*/\equiv$  of any string efficiently by reducing to normal form and then testing whether the resulting string is accepted by  $M$ . This will give us an efficient algorithm for the unbounded membership problem.

For the bounded membership problem, we will need a slightly stronger formulation. Define

$$A(x) = \{n \mid x \in \mathcal{S}^n/\equiv\}$$

for any string  $x$ . Note  $x \in \mathcal{S}^n/\equiv$  iff  $A(x) \neq \emptyset$ . Label each  $\epsilon$ -transition  $u \xrightarrow{\epsilon} v$  in  $M$  with the nonempty set  $A(u^{-1}v)$ . Let  $+$  denote setwise addition:

$$X + Y = \{m + n \mid m \in X, n \in Y\}.$$

For any computation path  $\sigma : u \xrightarrow{x} v$  in the automaton  $M$ , let  $A(\sigma)$  denote the sum of the sets labeling the  $\epsilon$ -transitions along the path  $\sigma$ . More formally,

$$\begin{aligned} A(\sigma) &= \{0\}, \text{ if } \sigma \text{ is of length } 0 \\ A(\sigma \cdot (au \xrightarrow{a} u)) &= A(\sigma) \\ A(\sigma \cdot (u \xrightarrow{\epsilon} v)) &= A(\sigma) + A(u^{-1}v). \end{aligned}$$

**Theorem 4.1** *For any reduced  $x$  and  $n \geq 0$ ,  $x \in \mathcal{S}^n / \equiv$  if and only if there is an accepting computation path  $\sigma : \epsilon \xrightarrow{x} \epsilon$  with  $n \in A(\sigma)$ . In other words, for any reduced  $x$ ,*

$$A(x) = \bigcup_{\sigma: \epsilon \xrightarrow{x} \epsilon} A(\sigma).$$

*Proof.* ( $\leftarrow$ ) We show by induction on the length of  $\sigma$  that if  $\sigma : \epsilon \xrightarrow{x} u$  and  $n \in A(\sigma)$  then  $xu \in \mathcal{S}^n / \equiv$ . The result follows by taking  $u = \epsilon$ . If  $\sigma$  is of length zero, then  $A(\sigma) = \{0\}$  and  $x = \epsilon \in \mathcal{S}^0$ . If  $\sigma = \tau \cdot (au \xrightarrow{a} u)$ , then  $x = ya$ ,  $\tau : \epsilon \xrightarrow{y} au$ , and  $n \in A(\tau) = A(\sigma)$ . By the induction hypothesis,  $xu = (ya)u = y(au) \in \mathcal{S}^n / \equiv$ . Finally, if  $\sigma = \tau \cdot (v \xrightarrow{\epsilon} u)$  where  $\tau : \epsilon \xrightarrow{x} v$ , then  $n = k + m$  for some  $k \in A(\tau)$  and  $m \in A(v^{-1}u)$ . Then  $v^{-1}u \in \mathcal{S}^m / \equiv$ , and by the induction hypothesis,  $xv \in \mathcal{S}^k / \equiv$ . Thus  $xu \equiv xv v^{-1}u \in \mathcal{S}^n / \equiv$ .

( $\rightarrow$ ) If  $x \in \mathcal{S}^0 / \equiv$ , then  $x = \epsilon$  since  $x$  is reduced. In this case take  $\sigma$  to be the null path  $\epsilon \xrightarrow{x} \epsilon$  and we are done. Otherwise, we show by induction that if  $r \in \mathcal{S}^{n-1}$ ,  $st \in \mathcal{S}$ , and  $rs \rightarrow y$  where  $y$  is reduced, then there is a computation path  $\sigma : \epsilon \xrightarrow{y} t$  with  $n \in A(\sigma)$ . The result then follows by taking  $t = \epsilon$ .

For  $n = 1$ , we have  $r = \epsilon$ . Then  $y = s$  since  $s$  is reduced, and there is a computation path  $\tau : \epsilon \xrightarrow{\epsilon} st$  of length one with  $1 \in A(\tau) = A(st)$ . Combining this with  $|s|$  transitions of the form  $au \xrightarrow{a} u$ , we obtain a computation path  $\sigma : \epsilon \xrightarrow{s} t$  with  $1 \in A(\sigma)$ .

Now suppose  $n \geq 2$ . If  $s = \epsilon$ , we have  $y \equiv r \in \mathcal{S}^{n-2} \mathcal{S}$  and  $t \in \mathcal{S}$ . Then  $1 \in A(t)$  and by the induction hypothesis, we have a computation path  $\tau : \epsilon \xrightarrow{y} \epsilon$  with  $n-1 \in A(\tau)$ . Combining this with the transition  $\epsilon \xrightarrow{\epsilon} t$ , we obtain a path  $\sigma : \epsilon \xrightarrow{y} t$  with  $n \in A(\sigma)$ .

If  $s \neq \epsilon$  and the last symbol of  $y$  comes from the last symbol of  $s$  in the reduction  $rs \rightarrow y$ , then  $s = ua$ ,  $y = va$ , and  $ru \rightarrow v$  for some  $u, v$ . By the induction hypothesis, we have a computation path  $\tau : \epsilon \xrightarrow{v} at$  with  $n \in A(\tau)$ . Combining this with the transition  $at \xrightarrow{a} t$ , we obtain a computation path  $\sigma : \epsilon \xrightarrow{y} t$  with  $n \in A(\sigma)$ .

Finally, if the last symbol of  $y$  does not come from the last symbol of  $s$ , then the last symbol of  $y$  cannot come from any symbol of  $s$ , since  $s$  is reduced. Thus

we can write  $r = upqv$  where  $u \in \mathcal{S}^k$ ,  $v \in \mathcal{S}^m$ ,  $pq \in \mathcal{S}$ , the last symbol of  $y$  comes from the last symbol of  $p$ , and  $qvs \equiv \epsilon$ . Then  $up \equiv upqvs = rs \equiv y$ . Since  $y$  is reduced,  $up \rightarrow y$ . By the induction hypothesis, we have a computation path  $\tau : \epsilon \xrightarrow{y} q$  with  $k+1 \in A(\tau)$ . Moreover, since  $qvs \equiv \epsilon$ , we have  $q^{-1}t \equiv vst \in \mathcal{S}^{m+1}$ , thus  $m+1 \in A(q^{-1}t)$ . Combining  $\tau$  with the transition  $q \xrightarrow{\epsilon} t$ , we obtain a computation path  $\sigma : \epsilon \xrightarrow{y} t$  with  $n = k + m + 2 \in A(\sigma)$ .  $\square$

**Corollary 4.2** *For any reduced  $x$ ,  $x \in \mathcal{S}^* / \equiv$  if and only if  $M$  accepts  $x$ .*

## 4.2 Construction of $M$

We have reduced the problem of determining membership in  $\mathcal{S}^* / \equiv$  of arbitrary strings  $x$  to the problem of determining membership in  $\mathcal{S}^* / \equiv$  of  $u^{-1}v$  for  $u, v \in Q$ . We now give an efficient algorithm for this problem.

Let  $N$  be the set of normal forms of strings  $u^{-1}v$  for  $u, v \in Q$ . Note  $\mathcal{S} \subseteq N$  and  $N$  is finite. Let  $B(x)$ ,  $x \in N$ , be the smallest family of sets closed under the following rules:

- (i)  $0 \in B(\epsilon)$
- (ii)  $1 \in B(x)$ ,  $x \in \mathcal{S}$
- (iii)  $B(x) + B(y) \subseteq B(z)$ , where  $z \equiv xy$ .

If  $x$  is not reduced but  $x \rightarrow y \in N$ , we define  $B(x) = B(y)$ .

We show below that  $A(x) = B(x)$  for  $x \in N$ . This gives a simple inductive method for determining the  $\epsilon$ -transitions of  $M$ : mark  $\epsilon$  and all  $x \in \mathcal{S}$  as required by rules (i) and (ii), then mark  $z \in N$  whenever  $x, y \in N$  are marked and  $xy \rightarrow z$ . Then  $u \xrightarrow{\epsilon} v$  iff the normal form of  $u^{-1}v$  is marked.

**Lemma 4.3** *If  $u \in Q$ ,  $pq \in \mathcal{S}$ ,  $r \in \mathcal{S}^n$ , and  $urp \equiv \epsilon$ , then  $n+1 \in B(u^{-1}q)$ .*

*Proof.* If  $n = 0$ , then  $u^{-1}q \equiv pq$ , and the conclusion follows from rule (ii).

If  $n \geq 1$  and  $u = \epsilon$ , then we can write  $r = vs$  with  $v \in \mathcal{S}$ ,  $s \in \mathcal{S}^{n-1}$ , and  $vsp \rightarrow \epsilon$ . Then  $1 \in B(u^{-1}v)$ , and by the induction hypothesis,  $n \in B(v^{-1}q)$ , therefore  $n+1 \in B(u^{-1}q)$  by rule (iii).

Similarly, if  $p = \epsilon$ , then we can write  $r = sv$  with  $s \in \mathcal{S}^{n-1}$ ,  $v \in \mathcal{S}$ , and  $usv \rightarrow \epsilon$ . Then  $1 \in B(\epsilon^{-1}v)$ , and by the induction hypothesis,  $n \in B(u^{-1}\epsilon)$ , therefore  $n+1 \in B(u^{-1}q)$  by rule (iii).

Assume now that  $n \geq 1$  and both  $u$  and  $p$  are nonnull. The proof proceeds by induction on the length of the reduction sequence  $urp \rightarrow \epsilon$ .

If  $urp$  can be expressed as the concatenation of two nonnull strings, each of which reduces to  $\epsilon$ , then the first of these cannot be a substring of  $u$  and the second cannot be a substring of  $p$ , since  $u$  and  $p$  are reduced. Thus we can write  $r = stxy$  where  $tx \in \mathcal{S}$ ,  $s \in \mathcal{S}^k$ ,  $y \in \mathcal{S}^m$ ,  $m + k + 1 = n$ ,  $ust \equiv xyq \equiv \epsilon$ . By the induction hypothesis,  $k + 1 \in B(u^{-1}x)$  and  $m + 1 \in B(x^{-1}q)$ . By rule (iii),  $n + 1 = m + k + 2 \in B(u^{-1}q)$ .

If  $urp$  has no such decomposition, then in the reduction  $urp \rightarrow \epsilon$ , if the last reduction rule applied is  $RR \rightarrow \epsilon$ , the first  $R$  must come from the leftmost symbol of  $u$  and the second must come from the rightmost symbol of  $p$ , otherwise we would have a decomposition as in the previous case. Thus  $u = Rx$ ,  $p = yR$ , and  $xry \rightarrow \epsilon$ . By the induction hypothesis, we have  $n + 1 \in B(x^{-1}Rq) = B(u^{-1}q)$ .

If the last reduction rule applied is  $SSS \rightarrow \epsilon$ , then again the first  $S$  must come from the leftmost symbol of  $u$  and the third must come from the rightmost symbol of  $p$ .

If the second  $S$  comes from  $u$ , then we have  $u = SSx$  and  $p = yS$ , where  $xry \rightarrow \epsilon$ . By the induction hypothesis we have  $n + 1 \in B(x^{-1}Sq) = B(u^{-1}q)$ .

If the second  $S$  comes from  $p$ , then we have  $u = Sx$  and  $p = ySS$ , where  $xry \rightarrow \epsilon$ . By the induction hypothesis we have  $n + 1 \in B(x^{-1}SSq) = B(u^{-1}q)$ .

Finally, if the second  $S$  comes from  $r$ , then we have  $u = Sx$ ,  $r = yzSws$ , and  $p = tS$ , where  $zSw \in \mathcal{S}$ ,  $y \in \mathcal{S}^k$ ,  $s \in \mathcal{S}^m$ , and  $xyz \equiv wst \equiv \epsilon$ . By the induction hypothesis we have  $k + 1 \in B(x^{-1}Sw)$  and  $m + 1 \in B(w^{-1}Sq)$ , therefore by rule (iii) we have  $n + 1 = m + k + 2 \in B(x^{-1}SSq) = B(u^{-1}q)$ .  $\square$

**Theorem 4.4**  $A(x) = B(x)$  for  $x \in N$ .

*Proof.* We argue first that the sets  $A(x)$  satisfy all the rules (i)–(iii) for  $x \in N$ , thus  $B(x) \subseteq A(x)$ . The rule (i) just says  $\epsilon \in \mathcal{S}^0$ , (ii) just says that  $x \in \mathcal{S}^1$  for  $x \in \mathcal{S}$ , and (iii) says that if  $x$  in  $\mathcal{S}^m$  and  $y \in \mathcal{S}^n$ , then  $xy \in \mathcal{S}^{m+n}$ .

For the reverse inclusion, we show by induction on  $n$  that for all  $u, v \in Q$ , if  $n \in A(u^{-1}v)$  then  $n \in B(u^{-1}v)$ . If  $n = 0$ , then  $u^{-1}v \equiv \epsilon$ , and  $0 \in B(u^{-1}v)$  by rule (i). If  $n = 1$ , then  $u^{-1}v \equiv x \in \mathcal{S}$ , and  $1 \in B(u^{-1}v)$  by rule (ii).

Assume now that  $n \geq 2$ . Let  $u^{-1}v \equiv r \in \mathcal{S}^n$ . Then  $ur \equiv v$ , and since  $v$  is reduced, we have  $ur \rightarrow v$ .

We proceed by induction on the length of  $v$ . If  $v = \epsilon$ , then writing  $r = st$  with  $s \in \mathcal{S}^{n-1}$  and  $t \in \mathcal{S}$ , we have  $ust \rightarrow \epsilon$ , so  $n \in B(u^{-1}v)$  by Lemma 4.3.

Suppose now that  $v$  is nonnull. If the first letter of  $v$  comes from  $u$  in the reduction  $ur \rightarrow v$ , then it must come from the first letter of  $u$ , since  $u$  is reduced. Thus  $u = ay$ ,  $v = aw$ , and  $yr \rightarrow w$ . By the induction hypothesis,  $n \in B(y^{-1}w) = B(u^{-1}v)$ .

If the first letter of  $v$  comes from  $r$ , then we can write  $r = styz$  where  $s \in \mathcal{S}^k$ ,  $z \in \mathcal{S}^m$ ,  $ty \in \mathcal{S}$ , and the first letter of  $v$  comes from the first letter of  $y$ . Then  $ust \rightarrow \epsilon$  and  $yz \rightarrow v$ . By Lemma 4.3,  $k + 1 \in B(u^{-1}y)$ , and by the induction hypothesis,  $m \in B(y^{-1}v)$ . By rule (iii),  $n = m + k + 1 \in B(u^{-1}v)$ .  $\square$

### 4.3 Unbounded Membership

Once we have constructed the automaton  $M$  for a given set of generators  $\mathcal{S}$ , we can solve the unbounded membership problem for a given string efficiently by reducing to normal form and then testing whether the resulting string is accepted by  $M$ . Corollary 4.2 asserts the correctness of this procedure.

### 4.4 Bounded Membership

One approach to solving the bounded membership problem is to observe that the closure rules (i)–(iii) are essentially equivalent to the following context-free grammar over a single-letter alphabet  $\{a\}$  and nonterminals  $A_x$ ,  $x \in N$ :

$$\begin{aligned} A_\epsilon &\rightarrow \epsilon \\ A_x &\rightarrow a, \quad x \in \mathcal{S} \\ A_z &\rightarrow A_x A_y, \quad xy \equiv z. \end{aligned}$$

Then for  $x \in N$ ,  $A(x)$  is the set of lengths of strings in  $\{a\}^*$  generated from the nonterminal  $A_x$ . By Parikh's Theorem, this is a regular set, and we can determine membership in  $A(x)$  efficiently using known algorithms for context-free language recognition.

However, for the purpose of deciding whether there exists an accepting computation path  $\sigma : \epsilon \xrightarrow{x} \epsilon$  with  $m \in A(\sigma)$  and  $m \leq n$ , we do not need to know the entire set  $A(u^{-1}v)$  but only its smallest element. Indeed, if  $A(u^{-1}v)$  is nonempty but its smallest element is greater than  $n$ , then we might as well delete the edge  $u \xrightarrow{\epsilon} v$ , since it cannot contribute to such an accepting computation path.

Let  $r$  be the number of relations  $x \equiv yz$  that hold among elements of  $N$ . Here is an  $O(nr)$  algorithm for determining all the minimum elements of  $A(x)$  for  $x \in N$ . For each  $x \in N$  we have an integer variable  $m_x$  that holds a current estimate of  $\min A(x)$ . We initialize  $m_x$  to  $n + 1$ , which we regard as  $\infty$ . We assume that for each  $x \in N$  we have a list  $L_x$  of all relations

$z \equiv xy$  or  $z \equiv yx$  that hold among the elements of  $N$  with  $x$  on the right hand side. The combined length of all the lists  $L_x$  is at most  $2r$ .

Now  $\min A(\epsilon) = 0$  and  $\min A(x) = 1$  for  $x \in \mathcal{S}$ , so we set  $m_\epsilon := 0$  and  $m_x := 1$  for  $x \in \mathcal{S}$  and put  $\epsilon$  and all  $x \in \mathcal{S}$  in a bag for further processing. We then repeat the following procedure until the bag becomes empty. Take the next  $x$  out of the bag and scan through the list  $L_x$ . For each relation  $z \equiv xy$  or  $z \equiv yx$  on the list, check whether  $m_z > m_x + m_y$ . If so, set  $m_z := m_x + m_y$  and put  $z$  in the bag.

Each  $x$  taken out of the bag takes  $O(|L_x|)$  time to process, and a particular  $x$  can enter the bag at most  $n$  times, since  $m_x$  is decremented each time. This gives  $O(nr)$  in all.

Once we have computed the minimum element of  $A(u^{-1}v)$  for each pair  $u, v \in Q$ , we can weight the  $\epsilon$ -transition  $u \xrightarrow{\epsilon} v$  with this quantity and weight the other transitions  $au \xrightarrow{a} u$  zero. Then to compute the minimum element of  $A(x)$  for a given reduced  $x$ , we can use a variant of Dijkstra's shortest path algorithm to find a minimum-weight computation path  $\epsilon \xrightarrow{x} \epsilon$  and check that its weight is at most  $n$ . The correctness of this method is given by Theorem 4.1. This solves the bounded membership problem.

## 5 Average Case Algorithms

In this section we prove Theorem 1.1, which states that the bounded and unbounded subgroup and submonoid membership problems are polynomial-time on average.

For a positive integer  $m$ , we take the *size* of  $m$  to be  $\log m$ , the base 2 logarithm of  $m$ . For a sequence  $\bar{m}$  of positive integers, we take the *size* of  $\bar{m}$ , denoted  $\|\bar{m}\|$ , to be the sum of the sizes of its components.

An instance of the unbounded subgroup or submonoid membership problem of §1.2 is a sequence  $\mathcal{S}$  of  $2 \times 2$  integer matrices with determinant one and entries written in binary. An instance of the bounded subgroup or submonoid membership problem is a pair  $(\mathcal{S}, n)$  where  $\mathcal{S}$  is as above and  $n$  is a positive integer. For our analysis, we will measure the size of such instances as follows. For a matrix with entries  $a, b, c, d$ , we take  $\mu(A) = \max\{|a|, |b|, |c|, |d|\}$ , where  $|a|$  denotes the absolute value of  $a$ . Let  $\mu(\mathcal{S})$  be the sequence  $(\mu(A) \mid A \in \mathcal{S})$ . We define the *size* of an instance  $\mathcal{S}$  of the unbounded membership problem to be  $\|\mathcal{S}\| = \|\mu(\mathcal{S})\|$ , and the size of an instance  $(\mathcal{S}, n)$  of the bounded membership problem to be  $\|(\mathcal{S}, n)\| = \|\mathcal{S}\| + n$ .

Let  $\sigma(\mathcal{S})$  denote the sum of the lengths of the  $R, S$  representations of the matrices in  $\mathcal{S}$ , as described in §2.

**Lemma 5.1** *Let  $\bar{m} = (m_1, \dots, m_k)$ . For  $d \geq 1$ , the quantity  $\sum_{i=1}^k (\log m_i)^d$  is maximized subject to the constraints  $1 \leq m_i$ ,  $1 \leq i \leq k$ , and  $\prod_{i=1}^k m_i = n$  at the extremes  $m_i = n$  and  $m_j = 1$ ,  $j \neq i$ .*

*Proof.* Taking  $a_i = \log m_i / \log n$ , the problem is equivalent to maximizing  $\sum_{i=1}^k a_i^d$  subject to the constraints  $0 \leq a_i$ ,  $1 \leq i \leq k$ , and  $\sum_{i=1}^k a_i = 1$ . This occurs at the extremes, since the function is convex and symmetric.  $\square$

*Proof of Theorem 1.1.* We treat the unbounded membership problems first. As remarked in §1.3, we need only show that there exists an  $\epsilon > 0$  such that

$$\sum_{\|\mathcal{S}\|=n} T(\mathcal{S})^\epsilon \cdot \Pr_n(\mathcal{S}) = n^{O(1)}, \quad (5.5)$$

where  $T(\mathcal{S})$  is the running time of the algorithm on input  $\mathcal{S}$  and  $\Pr_n(\mathcal{S})$  denotes the conditional probability that  $\mathcal{S}$  occurs given that the size of the input instance is  $n$ .

By results of §4, we have  $T(\mathcal{S}) = \sigma(\mathcal{S})^c$  for some constant  $c$ . Since all instances of size  $n$  are equally likely,  $\Pr_n \mathcal{S} = |\{\mathcal{S} \mid \|\mathcal{S}\| = n\}|^{-1}$  for  $\mathcal{S}$  of size  $n$ , where  $|X|$  denotes the cardinality of the set  $X$ . Taking  $\epsilon = 1/c$ , (5.5) becomes

$$\frac{\sum_{\|\mathcal{S}\|=n} \sigma(\mathcal{S})}{|\{\mathcal{S} \mid \|\mathcal{S}\| = n\}|} = n^{O(1)}. \quad (5.6)$$

We now establish (5.6). For  $\bar{\ell} = (\ell_1, \dots, \ell_k)$  and  $\bar{m} = (m_1, \dots, m_k)$ ,  $\bar{\ell} \leq \bar{m}$  means that  $\ell_i \leq m_i$ ,  $1 \leq i \leq k$ , and  $(\bar{\ell}, \bar{m}) = 1$  means that  $\ell_i$  and  $m_i$  are relatively prime,  $1 \leq i \leq k$ . The numerator of (5.6) is

$$\sum_{\|\mathcal{S}\|=n} \sigma(\mathcal{S}) = \sum_{k=1}^n \sum_{\substack{\bar{m} \in \mathbb{N}^k \\ \|\bar{m}\|=n}} \sum_{\mu(\mathcal{S})=\bar{m}} \sigma(\mathcal{S}) \quad (5.7)$$

and for  $\bar{m} \in \mathbb{N}^k$ ,

$$\begin{aligned} & \sum_{\mu(\mathcal{S})=\bar{m}} \sigma(\mathcal{S}) \\ & \leq 12k \sum_{\substack{\bar{\ell} \leq \bar{m} \\ (\bar{\ell}, \bar{m}) = 1}} \sum_{i=1}^k s(\ell_i, m_i) \\ & = O(n \sum_{\substack{\bar{\ell} \leq \bar{m} \\ (\bar{\ell}, \bar{m}) = 1}} \sum_{i=1}^k s(\ell_i, m_i)). \end{aligned} \quad (5.8)$$

The coefficient  $12k$  reflects the number of ways of choosing the positions of the largest elements of the matrices in  $\mathcal{S}$  and the factor bounding the lengths of the  $R, S$  and  $R, T, U$  representations as discussed in §2. The vectors  $\bar{\ell}$  represent the possible entries in the same row as the largest entry of each matrix in  $\mathcal{S}$ . As discussed in §2.1, once that row is given, the rest of the matrix is uniquely determined, and the length of the  $R, T, U$  representation of the  $i^{\text{th}}$  matrix in  $\mathcal{S}$  is  $s(\ell_i, m_i)$ .

Changing the order of summation in (5.8), we have

$$\begin{aligned}
& \sum_{i=1}^k \sum_{\substack{\bar{\ell} \leq \bar{m} \\ (\bar{\ell}, \bar{m}) = 1}} s(\ell_i, m_i) \\
&= \sum_{i=1}^k \left( \prod_{\substack{j=1 \\ j \neq i}}^k \varphi(m_j) \right) \left( \sum_{\substack{\ell_i \leq m_i \\ (\ell_i, m_i) = 1}} s(\ell_i, m_i) \right) \\
&= O\left( \sum_{i=1}^k m_i (\log m_i)^2 \prod_{\substack{j=1 \\ j \neq i}}^k \varphi(m_j) \right) \quad (5.9) \\
&= O\left( \left( \prod_{j=1}^k \varphi(m_j) \right) \sum_{i=1}^k \frac{m_i}{\varphi(m_i)} (\log m_i)^2 \right) \\
&= O\left( \left( \prod_{j=1}^k \varphi(m_j) \right) \sum_{i=1}^k (\log m_i)^3 \right). \quad (5.10)
\end{aligned}$$

Step (5.9) uses Lemma 3.1 and step (5.10) uses the estimate  $\varphi(m) \geq \Omega(m/\log \log m)$  [7, Theorem 328]. Thus (5.7) is bounded by

$$\begin{aligned}
& O\left( n \sum_{k=1}^n \sum_{\substack{\bar{m} \in \mathbb{N}^k \\ \|\bar{m}\| = n}} \left( \prod_{j=1}^k \varphi(m_j) \right) \sum_{i=1}^k (\log m_i)^3 \right) \\
&\leq O\left( n^4 \sum_{k=1}^n \sum_{\substack{\bar{m} \in \mathbb{N}^k \\ \|\bar{m}\| = n}} \prod_{j=1}^k \varphi(m_j) \right). \quad (5.11)
\end{aligned}$$

The inequality (5.11) follows from Lemma 5.1.

The denominator of (5.6) is

$$\begin{aligned}
& |\{\mathcal{S} \mid \|\mathcal{S}\| = n\}| \\
&= \sum_{k=1}^n \sum_{\substack{\bar{m} \in \mathbb{N}^k \\ \|\bar{m}\| = n}} \sum_{\mu(\mathcal{S}) = \bar{m}} 1 \\
&= \sum_{k=1}^n \sum_{\substack{\bar{m} \in \mathbb{N}^k \\ \|\bar{m}\| = n}} \sum_{\bar{\ell} \leq \bar{m} \\ (\bar{\ell}, \bar{m}) = 1} 4k
\end{aligned}$$

$$\geq \Omega\left( \sum_{k=1}^n \sum_{\substack{\bar{m} \in \mathbb{N}^k \\ \|\bar{m}\| = n}} \prod_{j=1}^k \varphi(m_j) \right). \quad (5.12)$$

Dividing the upper bound (5.11) for the numerator of (5.6) by the lower bound (5.12) for the denominator of (5.6), we obtain the polynomial bound  $O(n^4)$  for the quotient.

Thus the condition (5.5) is fulfilled, and the algorithm is polynomial time on average.

For the bounded membership problems, as above we need to show for each  $n$  that

$$\frac{\sum_{\|\mathcal{S}\| + m = n} \sigma(\mathcal{S}) + m}{|\{(\mathcal{S}, m) \mid \|\mathcal{S}\| + m = n\}|} = n^{O(1)}.$$

But the left hand side is bounded by

$$\begin{aligned}
& \frac{\sum_{\|\mathcal{S}\| \leq n} \sigma(\mathcal{S}) + \sum_{\|\mathcal{S}\| \leq n} n}{|\{\mathcal{S} \mid \|\mathcal{S}\| \leq n\}|} \\
&\leq \sum_{m=1}^n \frac{\sum_{\|\mathcal{S}\| = m} \sigma(\mathcal{S})}{|\{\mathcal{S} \mid \|\mathcal{S}\| = m\}|} + n,
\end{aligned}$$

which by (5.6) is polynomial in  $n$ .  $\square$

## Acknowledgements

We thank Michael Ben-Or, Marshall Cohen, Joachim von zur Gathen, David Henderson, Russell Impagliazzo, Donald Knuth, Richard Lipton, Gabriele Meyer, Paul Pedersen, Adi Shamir, Avi Wigderson, Andrew Yao, and Richard Zippel for their help. We gratefully acknowledge the support of the National Science Foundation under grants CCR-9057486 and CCR-9317320, BRICS (Basic Research in Computer Science), a Centre of the Danish National Research Foundation, the U.S. Army Research Office through ACSyAM, a branch of the Mathematical Sciences Institute of Cornell University under contract DAAL03-91-C-0027, and the Alfred P. Sloan Foundation.

## References

- [1] T. M. APOSTOL, *Modular Functions and Dirichlet Series in Number Theory*, Springer-Verlag, 1976.
- [2] S. BEN-DAVID, B. CHOR, O. GOLDBREICH, AND M. LUBY, *On the theory of average case complexity*, in 21st Symp. Theory of Computing, ACM, 1989, pp. 204–216.
- [3] N. BSHOUTY, T. HANCOCK, AND L. HELLERSTEIN, *Learning arithmetic read-once formulas*, in 24th Symp. Theory of Computing, 1992, pp. 370–381.

- [4] H. S. M. COXETER AND W. O. J. MOSER, *Generators and Relations for Discrete Groups*, Springer-Verlag, 4th ed., 1984.
- [5] Y. GUREVICH, *Matrix decomposition problem is complete for the average case*, in 31st Symp. Foundations of Computer Science, IEEE, 1990, pp. 802–811. SIAM J. Comput., to appear.
- [6] ———, *Average case complexity*, J. Comput. Syst. Sci., 42 (1991), pp. 346–398.
- [7] G. H. HARDY AND E. M. WRIGHT, *An Introduction to the Theory of Numbers*, Oxford University, 5th ed., 1979.
- [8] R. IMPAGLIAZZO AND L. LEVIN, *No better ways to generate hard NP instances than picking uniformly at random*, in 31st Symp. Foundations of Comput. Sci., IEEE, 1990, pp. 812–821.
- [9] F. KLEIN, *Über die Transformation der elliptischen Functionen und die Auflösung der Gleichungen fünften Grades*, Math. Ann., 14 (1879), pp. 111–172.
- [10] A. K. LENSTRA, H. W. LENSTRA, AND L. LOVÁSZ, *Factoring polynomials with rational coefficients*, Math. Ann., 261 (1982), pp. 515–534.
- [11] L. LEVIN, *Average case complete problems*, SIAM J. Comput., 15 (1986), pp. 285–286.
- [12] W. MAGNUS, A. KARRASS, AND D. SOLITAR, *Combinatorial Group Theory*, Interscience, 1966.
- [13] M. NEWMAN, *Integral Matrices*, Academic Press, 1972.
- [14] A. SCHÖNHAGE, *Schnelle berchnung von kettenbruchentwicklungen*, Acta Informatica, 1 (1971), pp. 139–144.
- [15] ———, *Fast reduction and composition of binary quadratic forms*. Preprint, 1992.
- [16] R. VENKATESAN AND L. LEVIN, *Random instances of a graph coloring problem are hard*, in 20th Symp. Theory of Computing, ACM, 1988, pp. 217–222.
- [17] A. C. YAO AND D. E. KNUTH, *Analysis of the subtractive algorithm for greatest common divisors*, Proc. Nat. Acad. Sci. USA, 72 (1975), pp. 4720–4722.
- [18] C. YAP, *Fast unimodular reductions: planar integer lattices*, in 33rd Symp. Foundations of Comput. Sci., IEEE, 1992, pp. 437–446.



**Lecturer: Noam Nisan**

**Title: Direct sums, products, and help bits in circuits  
and decision trees**

**Material: Noam Nisan, Steven Rudich, Michael Saks:**  
*Products and Help Bits in Decision Trees*

Institute of Computer Science  
Hebrew University  
Jerusalem, Israel  
E-mail: noam@CS.HUJI.AC.IL



# Products and Help Bits in Decision Trees

Noam Nisan \*

Steven Rudich †

Michael Saks ‡

## Abstract

We investigate two problems concerning the complexity of evaluating a function  $f$  at a  $k$ -tuple of unrelated inputs by  $k$  parallel decision tree algorithms.

In the *product problem*, for some fixed depth bound  $d$ , we seek to maximize the fraction of input  $k$ -tuples for which all  $k$  decision trees are correct. Assume that for a single input to  $f$ , the best decision tree algorithm of depth  $d$  is correct on a fraction  $p$  of inputs. We prove that the maximum fraction of  $k$ -tuples on which  $k$  depth  $d$  algorithms are all correct is at most  $p^k$ , which is the trivial lower bound. We show that if we replace the depth  $d$  restriction by “expected depth  $d$ ”, then this result fails.

In the *help-bit problem*, we are permitted to ask  $k - 1$  arbitrary binary questions about the  $k$ -tuple of inputs. For each possible  $k - 1$ -tuple of answers to these queries we will have a  $k$ -tuple of decision trees which are supposed to correctly compute all functions on  $k$ -tuples that are consistent with the particular answers. The complexity here is the maximum depth of any of the trees in the algorithm. We show that for all  $k$  sufficiently large, this complexity is equal to  $\deg^s(f)$  which is the minimum degree of a multivariate polynomial whose sign is equal to  $f$ .

Finally, we give a brief discussion of these problems in the context of other complexity models.

## 1 Introduction

Pick your favorite computation model and complexity measure, e.g. boolean circuit size, communication complexity, decision tree depth, interactive proof length, tensor rank, etc. Any attempt to understand such a model and complexity measure requires understanding the ways that an “unreasonable” computation can be more efficient than a “reasonable” one. Of course, what is “reasonable” changes as our understanding of the model improves.

Suppose we are given several unrelated instances of a problem to solve. The “reasonable” approach is to solve each instance separately; intuitively, any computation that is useful for solving one instance is irrelevant to any of the others. To what extent is this intuition valid in a given model? The following question is the most common way of formalizing this.

**The Direct-sum problem:** Suppose that the complexity of computing some function  $f$  is  $c$ . Is it true that computing  $f$  twice, on two unrelated inputs requires complexity  $2c$ ? How about computing  $f$  on  $k$  unrelated inputs?

This question was first studied in the context of Boolean circuits [Ulig, Paul, GF]. Subsequent work has concerned bilinear circuits [J, Bsh], Boolean circuits [FKN], and communication complexity [KRW]. In this paper we consider two related problems of a similar flavor:

**The Product Problem:** Let  $f$  be a function and suppose that for any complexity  $c$  computation, the fraction of inputs on which it correctly computes  $f$  is at most  $p$ . Suppose that we have two independent computations, each taking as input an ordered pair  $a, b$  of inputs to  $f$ , where the first computation is trying to compute  $f(a)$  and the second is trying to compute  $f(b)$ . If each of the two computations has complexity at most  $c$ , can the fraction of input pairs  $a, b$  on which both are correct exceed  $p^2$ ? What about the analogous question for  $k$  independent computations and  $k$  inputs?

If the first computation only uses the input  $a$  and

---

\*Computer Science Department, Hebrew University, Jerusalem, Israel. Supported by BSF grant 92-00043 and by a Wolfson award administered by the Israeli Academy of Sciences.

†Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa. Partially supported by NSF grant CCR-9119319.

‡Department of Mathematics, Rutgers University, New Brunswick, NJ 08903. Supported in part by NSF contracts CCR-9215293 and STC-91-19999 and by DIMACS

the second only uses the input  $b$ , then the  $p^2$  upper bound is trivial. Intuition suggests that there is no advantage in having each computation access the others input. A variant of this problem, in which we seek to compute  $f$  on the two inputs by a single computation was studied recently in [IRW].

**The Help-bit Problem:** Suppose that the complexity of computing the boolean function  $f$  is  $c$ . Suppose that one wishes to compute  $f$  on two inputs  $a$  and  $b$ , and is allowed for free one “help-bit”, i.e. an arbitrary function of the two inputs. Is it possible to choose this help-bit function so that, given the help-bit,  $f(a)$  and  $f(b)$  can each be computed by a computation of complexity less than  $c$ , and if so, by how much? How about computing  $f$  on  $k$  inputs with  $k - 1$  help bits?

The help-bit problem was introduced (to our knowledge) in the context of constant depth circuits in [Cai], and was also studied in the context of boolean circuits in [ABG]. The point here is that if we have  $k$  inputs, then with  $k$  help bits we can use them to obtain the value of  $f$  on each of the inputs, and no further computation is necessary. With only  $k - 1$  help bits, we can for instance obtain the value of  $f$  at  $k - 1$  inputs, but then we still need complexity  $c$  to compute  $f$  on the last input. Is there a more effective use of the help bits?

In this paper we consider these problems in the context of the boolean decision tree complexity – perhaps the simplest computational model. The cost of a computation (decision tree) is simply the number of input variables that are read (the depth of the decision tree); a more precise definition is given in Section 2. While it is an easy exercise to see that “direct-sum” holds for decision tree depth, the other two problems are more difficult. Our answer for the product problem is a qualified “Yes”:

**Theorem 1** *Let  $f$  be an  $n$ -variable boolean function and suppose that any depth  $d$  decision tree computes  $f$  correctly on a fraction at most  $p$  of the inputs. Let  $T_1, T_2, \dots, T_k$  be decision trees that each access a set of  $nk$  variables corresponding to a  $k$ -tuple  $a_1, a_2, \dots, a_k$  of inputs to  $f$ . If each of the  $T_i$  have depth at most  $d$ , then the fraction of  $k$ -tuples  $a_1, a_2, \dots, a_k$  on which each  $T_i$  correctly outputs  $f(a_i)$  is at most  $p^k$ .*

The theorem seems completely obvious; however, the reader might test her intuition on the following variation. Suppose that, in the above Theorem we change the complexity measure from “depth” to “average depth”, i.e. the average over all inputs of the

depth of the leaf reached by the input. This modified statement of the Theorem seems similarly obvious, but, as we will see, it is false.

The recent work of [IRW], which was done independently of ours, includes a (substantially different) proof of a weaker variant of this theorem, namely that a *single* depth  $d$  tree that tries to compute all  $k$  functions can be correct on at most a  $p^k$  fraction of the inputs. Our result shows that even if we use  $k$  parallel decision trees then we can’t do better than this.

For the help bit problem, the answer is more complicated. Nathan Linial [Lin] has shown that the complexity of computing  $f$  on two inputs with one help bit is at least  $\deg(f)$ , the degree of the (unique) multilinear real polynomial that is equal to  $f$ . Since almost all boolean functions on  $n$ -variables have  $\deg(f) = n$ , this says that help bits don’t help for most functions. This result does not seem to extend to  $k \geq 3$ . In fact, for sufficiently large  $k$  our results imply that it is false. We manage to prove a lower bound that holds for all  $k$ , and is always tight when  $k$ , the number of instances to be solved, is sufficiently large. We need the following definitions. If  $f$  is an  $n$ -variate boolean function, we say that the  $n$ -variate real polynomial  $p$  *sign-represents*  $f$  if for all inputs  $a$ ,  $f(a) = \text{sgn}(p(a))$  where  $\text{sgn}(z) = 1$  if  $z > 0$  and  $\text{sgn}(z) = -1$  otherwise (here we are taking our Boolean set to be  $\{-1, 1\}$ ). The *sign-degree* of  $f$ ,  $\deg^s(f)$ , is the minimum degree of a polynomial that sign represents  $f$ .

**Theorem 2** *Let  $f$  be an  $n$ -variate boolean function, and suppose that the optimal decision tree that computes  $f$  has depth  $d$ . Then for all  $k \geq 1$ , any solution to the help bit problem for  $f$  for  $k$  inputs and  $k - 1$  help bits requires depth at least  $\deg^s(f)$ . Furthermore, for all sufficiently large  $k$ , there is a decision tree algorithm with  $k - 1$  help bits whose depth is  $\deg^s(f)$ .*

In the case that  $f$  is equal to the product of  $n$  variables (which corresponds to the parity function for  $\{0, 1\}$ -valued variables),  $\deg^s(f) = n$  and so, the lower bound implies that help-bits don’t help in this case. Actually, this function and its negative are the only functions with  $\deg^s(f) = n$ . Since the ordinary decision tree complexity of most boolean functions is  $n$ , this means that for large enough  $k$ , the complexity of  $k$  instances given  $k - 1$  help bits is less than the ordinary decision tree complexity for most functions. In particular, if  $f$  is the majority function, then  $\deg^s(f) = 1$ , and the lower bound is vacuous, while the upper bound says that for  $k$  sufficiently large, it is possible to ask  $k - 1$  binary questions so that, given the answers, the value of the function on any one of the

$k$  inputs can be computed by probing just one variable. This remarkable savings is not typical, it was recently shown [R.R] that almost all functions satisfy  $\text{deg}^s(f) > n/20$ .

In the next section, we review the decision tree model. In Section 3 we give a general formulation for the product problem in decision trees, and prove a generalization (Theorem 3.1) of Theorem 1. In Section 4, we discuss the help bits problem and prove Theorem 2. Most proofs are in the appendices.

While some of the techniques we develop apply only to the decision tree model, some of them may be applied to other models as well, and in fact suffice for obtaining many of the known results in the boolean circuit model. We sketch these applications in the last section.

## 2 Preliminaries

In this section we present some basic definitions and notation. Most of the notions discussed here are very familiar, but in some cases our notation is non-standard.

### 2.1 Boolean functions

For purposes of this paper it will be convenient to use  $B = \{-1, 1\}$  as our Boolean domain, instead of  $\{0, 1\}$ . If  $X$  is a set, a *boolean assignment* to  $X$  is a map  $\alpha$  from  $X$  to  $B$ . The set of boolean assignments to  $X$  is denoted  $B^X$ . We refer to the elements of  $X$  as *variables*. We will consider probability distributions over the set of assignments. For a specified distribution  $D$ , a random assignment chosen according to  $D$  is denoted by placing a  $\sim$  above the identifier, e.g.,  $\tilde{\alpha}$ . A *boolean function* over the variable set  $X$  and range  $R$ , or  $(X, R)$ -*function* is a function from  $B^X$  to  $R$ . In this paper, the range  $R$  is always equal to  $B^k$  for some integer  $k$ .

### 2.2 Decision Trees

All trees in this paper are rooted, ordered, binary trees. For such a tree  $T$  every internal node  $v$  has exactly two children, and the two children are distinguished as the  $(-1)$ -child and  $(+1)$ -child of  $v$ . The depth  $d_T(v)$  of a node  $v$  is, as usual, the number of edges along the path from  $v$  to the root and the depth  $d_T$  of  $T$  is the maximum depth of any node in  $T$ .

Formally, a *decision tree over the variable set  $X$  with range  $R$*  or  $(X, R)$ -*decision tree* is a triple  $(T, p, a)$

where  $T$  is a rooted binary tree,  $p$  is a map that associates to each internal node  $v$  a variable  $x = p_v$  in the set  $X$ , and  $a$  is a map that associates each leaf  $v$  to an element  $a_v$  of  $R$ . The label  $p_v$  is called the *query* associated to  $v$ , and node  $v$  is said to *probe* variable  $p_v$ . We will generally say that  $T$  is an  $(X, R)$ -*decision tree*, keeping the maps  $p$  and  $a$  implicit. The set of  $(X, R)$ -*decision trees* over  $X$  is denoted  $\mathcal{T}(X, R)$ , or simply  $\mathcal{T}$ .

Let  $T$  be an  $(X, R)$ -*decision tree*. If  $\alpha$  is any assignment in  $B^X$ , the *computation* of  $T$  on  $\alpha$ , is the unique path  $v^0, v^1, v^2, \dots, v^s$  from the root of  $T$  to some leaf  $v^s = l_T(\alpha)$  as follows: start from the root  $v^0$  and inductively define  $v^{i+1}$  for  $i \geq 0$  as the  $\alpha(p_{v^i})$ -child of  $v^i$ . The output of the computation is the label  $a_{l_T(\alpha)}$ . Thus  $T$  can be viewed as a boolean function over  $X$  with range  $R$ . Trivially, every  $(X, R)$ -function  $f$  is computed by some  $(X, R)$ -*decision tree*.

The usual cost function for the computation performed by  $T$  on  $\alpha$  is the length (number of internal nodes) of the computation path, denoted  $C(T, \alpha)$ . The worst case complexity  $C(T)$  is the maximum over  $\alpha$  of  $C(T, \alpha)$ .  $C(f)$ , the decision tree depth of  $f$ , is the minimum of  $C(T)$  over all decision trees that compute  $f$ . For a distribution  $D$  on assignments, the distributional complexity  $C_D(T)$  is the average of  $C(T, \tilde{\alpha})$  with respect to the distribution  $D$ .

For a given  $(X, R)$ -function  $f$ , and a complexity bound  $b$  (with respect to some complexity measure), we are interested in how well  $f$  can be approximated by a tree of complexity at most  $b$ . The closeness of approximation is defined with respect to a probability distribution  $D$  on boolean assignments to  $X$ . Thus for each  $(X, R)$ -*decision tree*  $T$ , the *agreement probability*  $q_D(f; T)$  of  $T$  with  $f$  relative to  $D$ , is the probability that  $T(\tilde{\alpha}) = f(\tilde{\alpha})$ , with respect to the random assignment  $\tilde{\alpha}$  chosen according to  $D$ . The *decision tree approximation problem* for  $(f, D, \mathcal{U})$  where  $f$  is an  $(X, R)$ -function,  $D$  is a distribution over boolean assignments to  $X$ , and  $\mathcal{U}$  is a set of decision trees is to determine  $q_D(f; \mathcal{U})$ , which is defined to be the maximum agreement probability  $q_D(f; T)$  over all  $T \in \mathcal{U}$ . Of particular interest is the case that  $\mathcal{U}$  is the set  $\mathcal{T}_d(X, R)$  of decision trees of depth at most  $d$ .

Finally, a *decision forest*  $F$  over  $X$  and ranges  $R_1, R_2, \dots, R_k$  is an ordered sequence  $T_1, T_2, \dots, T_k$ , where  $T_i$  is an  $(X, R_i)$ -*decision tree*.  $F$  computes a boolean function from  $B^X$  to  $R = R_1 \times R_2 \times \dots \times R_k$ .

### 3 The Product Problem

Let  $X_1, X_2, \dots, X_k$  be pairwise disjoint sets of variables, and let  $D_1, D_2, \dots, D_k$  be, respectively, distributions over assignments to  $X_1, X_2, \dots, X_k$ . Let  $X = X_1 \cup X_2 \cup \dots \cup X_k$ . A boolean assignment  $\beta$  for  $X$  will be viewed as a  $k$ -tuple  $(\beta_1, \beta_2, \dots, \beta_k)$  where  $\beta_i$  is an assignment for  $X_i$ . Let  $D$  denote the distribution over assignments to  $X$  given by  $\text{Prob}_D(\tilde{\alpha} = \beta) = \prod_{i=1}^k \text{Prob}_{D_i}(\tilde{\alpha}_i = \beta_i)$ , i.e., the product distribution  $D_1 \times D_2 \times \dots \times D_k$ .

Now suppose that we have  $k$  decision tree approximation problems  $(f_1, D_1, \mathcal{U}_1), (f_2, D_2, \mathcal{U}_2), \dots, (f_k, D_k, \mathcal{U}_k)$ , where for each  $i$ ,  $f_i$  is a  $(X_i, R_i)$ -function, and let  $q_i = q_{D_i}(f_i; \mathcal{U}_i)$  be the optimal agreement probability for  $\mathcal{U}_i$  with  $f_i$  relative to  $D_i$ . It will be convenient sometimes to view  $f_i$  as a function of the entire variable set  $X$  that ignores all variables except those in  $X_i$ . We consider the problem of *simultaneously approximating*  $f_1, f_2, \dots, f_k$  by a decision forest  $F = (T_1, T_2, \dots, T_k)$  where  $T_i \in \mathcal{U}_i$ . The *simultaneous agreement probability*  $q_D(f_1, f_2, \dots, f_k; T_1, T_2, \dots, T_k)$  for  $T_1, T_2, \dots, T_k$  with  $f_1, f_2, \dots, f_k$  denotes the probability, for  $\tilde{\alpha}$  chosen according to  $D$ , that  $(T_1(\tilde{\alpha}) = f_1(\tilde{\alpha})) \wedge (T_2(\tilde{\alpha}) = f_2(\tilde{\alpha})) \wedge \dots \wedge (T_k(\tilde{\alpha}) = f_k(\tilde{\alpha}))$ . For  $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_k$  where  $\mathcal{U}_i$  is a family of  $(X, R_i)$ -trees,  $q_D(f_1, f_2, \dots, f_k; \mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_k)$  denotes the maximum of  $q_D(f_1, f_2, \dots, f_k; T_1, T_2, \dots, T_k)$  over all choices of trees with  $T_i \in \mathcal{U}_i$ .

Now, since  $f_i$  only depends on  $X_i$  and  $D$  chooses the assignments  $\tilde{\alpha}_1, \tilde{\alpha}_2, \dots, \tilde{\alpha}_k$  to  $X_1, X_2, \dots, X_k$  independently, it would seem that  $q_D(f_1, f_2, \dots, f_k; T_1, T_2, \dots, T_k)$  should just be the product of the probabilities  $q_{D_i}(f_i; T_i)$ . This is clearly the case if each tree  $T_i$  only queries variables in  $X_i$ . However (as shown by the examples in below), if  $T_i$  is allowed to query variables outside of  $X_i$ , then this need not be the case. Intuitively, it would seem that variables outside of  $X_i$  could not help to approximate  $f_i$  and indeed this is trivially true, if we are only trying to approximate  $f_i$ . But when we seek to approximate all of the functions simultaneously, it is no longer obvious that such "cross-queries" are irrelevant.

Nevertheless, one might expect that for "reasonable" classes  $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_k$  of decision trees, the optimal simultaneous agreement probability is attained by a sequence of trees  $T_1, T_2, \dots, T_k$  with  $T_i$  querying variables only in  $X_i$ , and is thus equal to the product of the individual optimal agreement probabilities. The main result of this section is to prove this in the case that for each  $i$ ,  $\mathcal{U}_i$  is the set of trees of some fixed

depth  $d_i$ .

**Theorem 3.1** Let  $f_1, f_2, \dots, f_k$  and  $D_1, D_2, \dots, D_k, D$  be as above. Let  $d_1, d_2, \dots, d_k$  be nonnegative integers. Then

$$q_D(f_1, f_2, \dots, f_k; \mathcal{T}_{d_1}, \mathcal{T}_{d_2}, \dots, \mathcal{T}_{d_k}) = \prod_{i=1}^k q_{D_i}(f_i, \mathcal{T}_{d_i})$$

Note that Theorem 1 is a special case of the above. Before giving the proof we present two examples to show that multiplicativity fails for some natural alternative choices of the classes  $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_k$ .

**Example 3.1** Theorem 3.1 fails if we replace the class  $\mathcal{T}_{d_i}$  by the class  $\mathcal{S}_{d_i}^i$  of trees that are restricted to query at most  $d_i$  variables from  $X_i$  along any path, but can query variables outside  $X_i$  for free. Consider the following trivial example. Let  $k = 2$  and let  $X_1 = \{x_1\}, X_2 = \{x_2\}$ . The distribution  $D_1$  assigns  $x_1$  to 1 with probability  $1/2$ , and  $D_2$  assigns  $x_2$  to 1 with probability  $1/2$ . The functions  $f_1$  and  $f_2$  are given by  $f_1(x_1) = x_1, f_2(x_2) = x_2$ . Now let  $d_1 = d_2 = 0$ . This means that we do not allow  $T_1$  to look at any variables in  $X_1$  and we do not allow  $T_2$  to look at any variable in  $X_2$ . Clearly  $q_{D_1}(f_1, \mathcal{S}_0^1) = q_{D_2}(f_2, \mathcal{S}_0^2) = 1/2$ . However, we can achieve simultaneous agreement probability better than  $1/4$ . Let  $T_1$  be the tree that queries  $x_2$  and outputs  $x_2$  and  $T_2$  be the tree that queries  $x_1$  and outputs  $x_1$ . Then, the probability that both  $T_1$  and  $f_1$  agree and  $T_2$  and  $f_2$  agree is just the probability that  $x_1$  and  $x_2$  are assigned the same value, which is  $1/2$ .

A somewhat more subtle example is given by:

**Example 3.2** For a distribution  $D$  over  $B^X$ , let  $\mathcal{T}_d^D$  be the class of trees whose expected depth with respect to  $D$  is  $d$ , i.e.,  $T \in \mathcal{T}_d^D$  if the average number of variables queried, with respect to  $\tilde{\alpha}$  chosen from  $D$  is at most  $d$ . Then the above theorem is false if we replace  $\mathcal{T}_{d_i}$  by  $\mathcal{T}_{d_i}^D$ . To see this, let  $X$  be a set of four variables, and  $f$  be the parity function on  $X$ . Let  $U$  be the uniform distribution over assignments to  $X$  and let  $d = 3$ . First we show that the maximum agreement probability with  $f$  attained by a decision tree  $S$  of expected depth at most 3, is equal to  $3/4$ . Agreement probability  $3/4$  is attained by the tree  $S$  that queries a particular variable  $x$ , and if it is 0, then it returns 0, and otherwise it queries the remaining three variables and returns the parity of them. To see that this is best possible, note that if  $T$  is any decision tree algorithm, then for each leaf  $l$  in  $T$  of depth less than 4,  $T$  will agree with  $f$  on exactly half of the inputs

that reach  $l$ . Thus, if  $p_i$  is the probability that a random input  $\tilde{\alpha}$  ends up at a leaf of depth  $i$ , then the agreement probability  $q_D(f; T)$  can be bounded above by  $p_4 + 1/2(1 - p_4)$ ; it suffices to show that  $p_4 \leq 1/2$ . Now  $p_1$  either equals 0,  $1/2$  or 1. If  $p_1 > 0$  then  $p_4 \leq 1/2$ . If  $p_1 = 0$ , then the expected depth of the tree is at least  $4p_4 + 2(1 - p_4) = 2 + 2p_4$ , which means that  $p_4 \leq 1/2$ .

Now let  $X_1, f_1, D_1$  and  $X_2, f_2, D_2$  be copies of  $X, f, U$  on disjoint variable sets. We show that it is possible to choose decision trees  $T_1, T_2$  each of expected depth at most 3, whose agreement probability exceeds  $9/16 = (3/4)^2$ . Let  $T_1$  be the  $S$  described above and let  $x_1$  denote the variable in  $X_1$  probed first by  $T_1$ . Let  $T_2$  be the following tree: first probe  $x_1$  (in  $X_1$ ). If it is 0, output 0. If it is one, then read all four variables in  $X_2$  and output their parity. The expected depth of this tree is 3, since half the paths have depth one and half the paths have depth five. Now, let us consider the probability of the event  $A$  that both  $T_1(\tilde{\alpha}) = f_1(\tilde{\alpha})$  and  $T_2(\tilde{\alpha}) = f_2(\tilde{\alpha})$ . Then  $\text{Prob}_D(A) = 1/2(\text{Prob}_D(A|x_1 = 0) + \text{Prob}(A|x_1 = 1))$ . The conditional probability of  $A$  given  $x_1 = 0$  is  $1/4$ . If  $x_1 = 1$  then  $T_1$  must agree with  $f_1$ , and  $T_2$  must agree with  $f_2$ . Thus the probability of simultaneous agreement is  $5/8 = 10/16$ .

What happens in the above example is that the variable  $x_1$  acts as a shared random coin that partially coordinates the two computations so that they are more likely to be simultaneously correct.

**Proof of Theorem 3.1** Fix a sequence  $T_1, T_2, \dots, T_k$  of decision trees with  $T_i$  of depth at most  $d_i$ . For  $I \subseteq [k] = \{1, 2, \dots, k\}$ , let  $C(I)$  denote the event  $\bigwedge_{i \in I} (T_i = f_i(x^i))$ , i.e., the event that all of the trees indexed by  $I$  evaluate their respective functions correctly. We seek to prove that  $\text{Prob}[C([k])]$  is bounded above by  $\prod_{i=1}^k q_{D_i}(f_i, \mathcal{T}_{d_i})$ .

The proof is by induction on  $k$ , and for fixed  $k$  by induction on the  $k$ -tuple  $d_1 + d_2 + \dots + d_k$ . The result is vacuous if  $k = 1$ .

So assume that  $k \geq 2$ . Consider first the case that  $d_i = 0$  for some  $i$ . We may assume that  $d_k = 0$ . Thus, the  $k^{\text{th}}$  party must guess the value of  $f_k(\tilde{\alpha}_k)$  without looking at any variables, so  $T_k$  consists of a single leaf labeled -1 or 1. Now, by conditioning on the value of the vector  $\tilde{\alpha}_k$ , the probability,  $P^*$  that  $C([k])$  holds can be written:

$$P^* \leq \sum_{\beta \in B^{x_k}} \text{Prob}[\tilde{\alpha}_k = \beta] \times$$

$$\begin{aligned} & \text{Prob}[T_k(\beta) = f_k(\beta)] \times \\ & \text{Prob}[C([k-1])|\tilde{\alpha}_k = \beta] \\ \leq & \max_{\beta \in B^{x_k}} \text{Prob}[C([k-1])|\tilde{\alpha}_k = \beta] \times \\ & \sum_{\beta \in B^n} \text{Prob}[\tilde{\alpha}_k = \beta] \text{Prob}[T_k(\beta) = f_k(\beta)] \\ = & \max_{\beta \in B^{x_k}} \text{Prob}[C([k-1])|\tilde{\alpha}_k = \beta] \times \\ & \text{Prob}[T_k(\tilde{\alpha}) = f_k(\tilde{\alpha})] \\ \leq & \max_{\beta \in B^{x_k}} \text{Prob}[C([k-1])|\tilde{\alpha}_k = \beta] \times \\ & q_{D_k}(f_k, 0). \end{aligned}$$

Now let  $\gamma$  be the assignment of  $\tilde{\alpha}_k$  that maximizes the probability in the last expression. For each  $i$  between 1 and  $k-1$ , define the tree  $U_i$  by contracting  $T_i$  using  $\tilde{\alpha}_k = \gamma$ . Then we may rewrite the last term as  $\text{Prob}[(U_1(\tilde{\alpha}) = f_1(\tilde{\alpha})) \wedge \dots \wedge (U_{k-1}(\tilde{\alpha}) = f_{k-1}(\tilde{\alpha}))] \times q_{D_k}(f_k, 0)$ .

Each tree  $U_i$  has depth at most  $d_i$ , and so we may bound the first factor by  $q_D(f_1, f_2, \dots, f_{k-1}; \mathcal{T}_{d_1}, \mathcal{T}_{d_2}, \dots, \mathcal{T}_{d_{k-1}})$  which by the induction hypothesis equals  $\prod_{i=1}^k q_{D_i}(f_i, \mathcal{T}_{d_i})$ . Thus the desired result follows.

Now we assume that  $d_i > 0$  for all  $i$ . Define a directed graph on  $\{1, 2, \dots, k\}$  with an edge from  $i$  to  $j$  if the first variable probed by  $T_i$  is an input to  $f_j$ . Since this directed graph has out-degree one, it has a directed cycle. Let  $j \geq 1$  be the length of the cycle. Let us rename the set of indices in the cycle by the set  $[j] = \{1, 2, \dots, j\}$  in such a way that for each  $i < j$ , the first probe of  $T_i$  is a variable, denoted  $x_{i+1}$ , in  $X_{i+1}$  and the first probe of  $T_j$  is a variable, denoted  $x_1$ , in  $X_1$ .

The intuition behind the rest of the proof is that for  $i \in [j]$ , it is possible to replace each tree  $T_i$  by trees of the same depth in which the first probe in  $T_i$  is  $x_i$ , without decreasing the probability of simultaneous agreement.

For  $b \in B$ , let  $f_i^b$  denote the function obtained from  $f_i$  by fixing  $x_1^i = b$ . Also, let  $D_i^b$  be the distribution on the set  $X_i - x_i$  obtained from  $D_i$  by conditioning on  $x_1^i = b$ .

Now, for  $\mathbf{b} = (b_1, b_2, \dots, b_j) \in B^j$  let  $A(\mathbf{b})$  denote the event that  $(\tilde{\alpha}_1(x_1) = b_1) \wedge \dots \wedge (\tilde{\alpha}_j(x_j) = b_j)$ . We can write the probability that all of the  $T_i$  compute correctly by conditioning on  $\mathbf{b}$  as follows:

$$\sum_{\mathbf{b} \in B^j} \text{Prob}[A(\mathbf{b})] \text{Prob}[C([k])|A(\mathbf{b})] \quad (1)$$

We seek to upper bound this expression by:

$$\prod_{i=1}^k q_{D_i}(f_i, \mathcal{T}_{d_i}). \quad (2)$$

To do this we show:

**Claim.** For each  $\mathbf{b} \in B^{[l]}$ , the conditional probability of  $C([k])$  given  $A(\mathbf{b})$  is at most:

$$\left( \prod_{i=1}^j q_{D_i^{b_i}}(f_i^{b_i}, \mathcal{T}_{d_i}) \right) \left( \prod_{i=j+1}^k q_{D_i}(f_i, \mathcal{T}_{d_i}) \right)$$

Assuming the claim for the moment, we can then substitute into the expression (1) to obtain the following bound on the probability that all of the trees are correct:

$$\left( \prod_{i=j+1}^k q_{D_i}(f_i, \mathcal{T}_{d_i}) \right) \sum_{\mathbf{b} \in B^j} \text{Prob}[A(\mathbf{b})] \left( \prod_{i=1}^j q_{D_i^{b_i}}(f_i^{b_i}, \mathcal{T}_{d_i}) \right) \quad (3)$$

The sum can be rewritten as:

$$\sum_{\mathbf{b} \in B^j} \prod_{i=1}^j \text{Prob}_{D_i}[x_i = b_i] q_{D_i^{b_i}}(f_i^{b_i}, \mathcal{T}_{d_i}),$$

which is equal to:

$$\prod_{i=1}^j \left( (\text{Prob}_{D_i}[\tilde{\alpha}(x_i) = -1] q_{D_i^{-1}}(f_i^{-1}, \mathcal{T}_{d_i}) + \text{Prob}_{D_i}[\tilde{\alpha}(x_i) = 1] q_{D_i^1}(f_i^1, \mathcal{T}_{d_i}) \right)$$

Now, the  $i^{\text{th}}$  term in this product corresponds to the probability of correctly computing  $f_i$  if we first probe  $x_i$  and then, depending on the outcome, use the optimal depth  $d_i - 1$  tree to evaluate the residual function. Thus, we can upper bound this term by  $p(f_i, D_i, d_i)$ . But then, the expression (3) is upper bounded by the expression (2) as required.

So it suffices to prove the claim. Define  $f_i^{A(\mathbf{b})}$  to be the function  $f_i^{b_i}$  for  $i \leq j$  and to be  $f_i$  otherwise. Similarly, the distribution  $D_i^{A(\mathbf{b})}$  is equal to  $D_i^{b_i}$  for  $i \leq j$  and to  $D_i$  otherwise. Observe that by the mutual independence of  $\tilde{\alpha}_1, \tilde{\alpha}_2, \dots, \tilde{\alpha}_k$ , their joint distribution given  $A(\mathbf{b})$  is the product distribution of  $D_i^{A(\mathbf{b})}$  for  $i$  between 1 and  $k$ .

Let  $T_i^{A(\mathbf{b})}$  be the tree obtained from contracting  $T_i$  under the assumption that  $A(\mathbf{b})$  holds. Then the conditional probability that  $T_i = f_i(x^i)$  for all  $i$ , given

$A(\mathbf{b})$  is equal to the probability (with respect to the product distribution on  $D_i^{A(\mathbf{b})}$ ) that for all  $i$ ,  $T_i^{A(\mathbf{b})} = f_i^{A(\mathbf{b})}$ . Now for each  $i$  the depth of  $T_i^{A(\mathbf{b})}$  has at most  $d_i - 1$  if  $i < j$ , and is at most  $d_i$  for  $i > j$ , so we may apply induction to say that the probability with respect to the product distribution on  $D_i^{A(\mathbf{b})}$  that for all  $i$ ,  $T_i^{A(\mathbf{b})} = f_i^{A(\mathbf{b})}$  is at most:

$$\left( \prod_{i=1}^j q_{D_i^{A(\mathbf{b})}}(f_i^{A(\mathbf{b})}, \mathcal{T}_{d_i}) \right) \left( \prod_{i=j+1}^k q_{D_i^{A(\mathbf{b})}}(f_i^{A(\mathbf{b})}, \mathcal{T}_{d_i}) \right)$$

which is equal to the expression in the claim. This proves the claim and the Theorem.

**Remark.** The proof of the Theorem can be extended to a more general model of decision tree computation. For this model, in the case of a single function we are given a function  $f$  from an arbitrary domain  $S$  to  $R$ , and want to compute  $f(s)$  for an unknown input  $s \in S$ . We are further given a set  $Q$  of admissible queries, where each query  $q \in Q$  is a partition of  $S$  into sets  $(S_1^q, S_2^q, \dots, S_r^q)$ . The response to query  $q$  is the index  $i$  such that  $s \in S_i^q$ . The nodes of a decision tree are labeled by queries, and the branches out of the node correspond to the answers to the query. For a collection of functions  $f_i$  on disjoint domains  $S_i$ , the formulation of the product problem generalizes to this model. The statement and proof of the Theorem now go through assuming: (1) That the any allowed query depends only on variables from one function and (2) The distributions  $D_i$  are independent.

## 4 Help Bits

In the help bits problem, We have  $k$  boolean functions  $f_1, f_2, \dots, f_k$  over disjoint variable sets  $X_1, X_2, \dots, X_k$ . Given an unknown assignment  $\alpha$  to the variables of the set  $X = X_1 \cup \dots \cup X_k$ , we want to evaluate  $f_i(\alpha_i)$  for all  $i$ , by a decision forest. We are allowed to ask, "for free", an arbitrary set of  $l$  binary questions about the assignment  $\alpha$ . The answer to these  $l$  questions is a vector  $\mathbf{a} \in B^l$ . For each such  $\mathbf{a}$  we will have a decision forest  $F^{\mathbf{a}} = (T_1^{\mathbf{a}}, T_2^{\mathbf{a}}, \dots, T_k^{\mathbf{a}})$ , where we require that  $F^{\mathbf{a}}(\alpha)$  agrees with  $(f_1(\alpha_1), \dots, f_k(\alpha_k))$  for every assignment  $\alpha$  that is consistent with  $\mathbf{a}$ .

Thus, such an algorithm is specified by  $l$  arbitrary boolean functions  $h_1, h_2, \dots, h_l$  (the help bits) on variable set  $X$ , together with  $2^l$  decision forests. The complexity of the algorithm is the maximum depth of any

of the  $2^l k$  decision trees in these forests. In general, the decision tree  $T_i^a$  that computes  $f_i(\alpha_i)$  for  $\alpha$  consistent with  $\mathbf{a}$  is allowed to probe variables outside of  $X_i$ . This is conceivably useful, because together with the help bits, such probes could imply information about the variables in  $X_i$ . For instance if one of the help bit functions is  $(f_i(\alpha_i) \times \alpha_j(x))$  where  $x$  is a variable in  $X_j$ , then by probing the variable  $x$ , we can deduce  $f_i(\alpha_i)$ . If  $T_i^a$  only probes variables in  $X_i$  we say that it is *pure*. If each of the  $2^l k$  decision trees are pure, the algorithm is pure.

In this paper, we will restrict attention to the case that, for some variable set  $X$  and boolean function  $f$  over  $X$ , each of the  $X_i$  are copies of  $X$  and the functions  $f_i$  are copies of  $f$ . The help bits problem  $H^{k,l}(f)$  is to evaluate  $k$  copies of  $f$  given  $l$  help bits. Define  $C^{k,l}(f)$  to be the complexity of the optimal algorithm that solves it. We also define the problem  $H_{\text{pure}}^{k,l}(f)$  to be the same as  $H^{k,l}(f)$  except that we require that the algorithm be pure. Define  $C_{\text{pure}}^{k,l}(f)$  to be the complexity of the optimal pure algorithm. Our goal is to obtain bounds on  $C^{k,l}(f)$  and  $C_{\text{pure}}^{k,l}(f)$ . The main result of this section (which is a slight refinement of Theorem 2), is:

**Theorem 4.1** *For any boolean function  $f$  on  $n$  variables and any positive integer  $k$ ,*

$$C_{\text{pure}}^{k,k-1}(f) \geq C^{k,k-1}(f) \geq \text{deg}^*(f).$$

*If  $k$  is sufficiently large, then*

$$C^{k,k-1}(f) = C_{\text{pure}}^{k,k-1}(f) = \text{deg}^*(f).$$

We first reformulate the problems  $H^{k,l}(f)$  and  $H_{\text{pure}}^{k,l}(f)$ . Given functions  $f_1, f_2, \dots, f_k$  as above, and a decision forest  $F$ , we say that  $F$  covers the assignment  $\alpha$  of  $X$ , with respect to  $f_1, f_2, \dots, f_k$ , if  $F(\alpha) = (f_1(\alpha), f_2(\alpha), \dots, f_k(\alpha))$ . Let  $\tau^k(f, d)$  be the minimum number of forests, each consisting of trees of depth at most  $d$ , needed to cover all inputs with respect to  $f$ . Let  $\tau_{\text{pure}}^k(f, d)$  be the corresponding minimum when we restrict to forests that are pure.

**Proposition 4.1** *Let  $f$  be a boolean function and  $k, l, d$  be nonnegative integers. Then:*

1.  $C^{k,l}(f) \leq d$  if and only if  $\tau^k(f, d) \leq 2^l$ ,
2.  $C_{\text{pure}}^{k,l}(f) \leq d$  if and only if  $\tau_{\text{pure}}^k(f, d) \leq 2^l$ .

*In other words,  $\lceil \log_2 \tau^k(f, d) \rceil$  is the minimum  $l$  such that  $H^{k,l}$  can be solved with trees of depth  $d$ , and  $\lceil \log_2 \tau_{\text{pure}}^k(f, d) \rceil$  is the minimum  $l$  such that  $H_{\text{pure}}^{k,l}$  can be solved with trees of depth  $d$ .*

**Proof.** We prove the first assertion; the proof of the second is completely analogous. If  $C^{k,l}(f) \leq d$ , then the  $2^l$  forests given by the algorithm are also a cover and  $\tau^k(f, d) \leq 2^l$ . Now suppose  $\tau^k(f, d) \leq 2^l$ . Then there is a collection of  $2^l$  forests that cover all assignments of  $X$ . Index these forest as  $F^z$  where  $z$  ranges over  $B^l$ . Order the forests lexicographically, and define  $A(z)$  to be the set of assignments that are covered by  $F^z$  but not covered by  $F^y$  for any  $y \leq z$ . Then the sets  $\{A(z) : z \in B^l\}$  partition the set of all assignments of  $X$ . Now define the help bit functions  $h_1, h_2, \dots, h_l$  so that for each  $\alpha$ ,  $(h_1(\alpha), h_2(\alpha), \dots, h_l(\alpha))$  is the unique index  $z$  such that  $\alpha \in A(z)$ . Then these functions together with  $\{F^z : z \in B^l\}$  solves  $H^{k,l}$ . ■

So we now concentrate on obtaining bounds on  $\tau^k(f, d)$  and  $\tau_{\text{pure}}^k(f, d)$ . For this we need yet another definition. A *randomized  $(X, R)$ -decision tree algorithm* is a probability distribution  $Q$  over  $(X, R)$ -decision trees. Such an algorithm is said to *approximate  $f$  with probability  $p$*  if for each assignment  $\alpha$ , if  $\tilde{T}$  is a random decision tree chosen according to  $Q$ , then the probability that  $\tilde{T}(\alpha) = f(\alpha)$  is at least  $p$ . We define  $p(f, d)$  to be the maximum  $p$  such that there is a distribution  $Q$  over the set of decision trees of depth at most  $d$  that approximates  $f$  with probability  $p$ . It is easy to see that  $p(f, d) \geq 1/2$ . and that if  $d = C(f)$ , the ordinary decision tree complexity of  $f$ , then  $p(f, d) = 1$ . The following result relates  $\tau(f, d)$  to  $p(f, d)$ .

**Lemma 4.1** *For any boolean function  $f$  on  $n$  variables and  $k, d \geq 0$ , we have:*

$$\frac{1}{p(f, d)^k} \leq \tau(f, d) \leq \tau_{\text{pure}}(f, d) \leq \lceil \frac{nk \ln 2}{p(f, d)^k} \rceil$$

**Proof.** The middle inequality is trivial. For the last inequality, we use a standard probabilistic argument to show that there is family of at most  $\lceil \frac{nk \ln 2}{p(f, d)^k} \rceil$  pure forests of depth at most  $d$  that cover all of the assignments. Let  $Q$  be the distribution over  $(Y, R)$ -decision trees of depth at most  $d$  that approximates  $f$  with probability  $p(f, d)$ . For  $i \leq k$ , let  $Q_i$  be the corresponding distribution over the set of  $(X_i, R)$  decision trees;  $Q_i$  approximates  $f_i$  with probability  $p(f, d)$ . Consider the distribution  $P = Q_1 \times \dots \times Q_k$  over forests. Suppose we select  $t$  forests  $\tilde{F}_1, \tilde{F}_2, \dots, \tilde{F}_t$  according to  $P$ . For a given assignment  $\alpha$  and  $j \leq t$ , the probability that  $\tilde{F}_j$  covers  $\alpha$  is at least  $p(f, d)^k$ . Thus the probability that none of the forests cover  $\alpha$  is at most  $(1 - p(f, d)^k)^t$ , and the probability that there

exists an assignment  $\alpha$  that is covered by none of the forests is at most  $2^{nk}(1-p(f,d)^k)^t < 2^{nk}e^{-p(f,d)^kt}$ . If  $t = \lceil (nk \ln 2)/p(f,d)^k \rceil$  then this expression is at most 1, so there is a positive probability that the forest covers all assignments, and so there must be a collection of  $t$  forests of depth  $d$  that cover all assignments.

Now we turn to the lower bound on  $\tau(f,d)$ . For this, we need the following relationship between  $p(f,d)$  and the agreement probability  $q_{\hat{D}}(f,d)$  with respect to a particular distribution  $\hat{D}$  on assignments.

**Lemma 4.2** *For any  $(Y,R)$ -boolean function  $f$  and integer  $d \geq 0$ , there exists a distribution  $\hat{D}$  on assignments to  $Y$  such that  $q_{\hat{D}}(f,d) = p(f,d)$ .*

This is a variant of a fundamental observation of Yao [Y1], and follows from the min-max theorem for two person zero sum games.

Let  $\hat{D}$  be the distribution of the lemma. Suppose that  $F_1, F_2, \dots, F_t$  is a family of forests that cover all assignments  $\alpha$  to  $X$ . Consider the distribution  $P$  over all assignments  $\alpha$  which is the product  $\hat{D}_1 \times \hat{D}_2 \times \dots \times \hat{D}_k$ , where  $\hat{D}_i$  is the copy of  $\hat{D}$  on  $X_i$ . Then, by Theorem 3.1, for any forest  $F_i$ , the probability that it covers  $\alpha$  is at most  $p(f,d)^k$ . Then the expected number of assignments covered by  $F_1, F_2, \dots, F_t$  is at most  $tp(f,d)^k$ . Since  $F_1, F_2, \dots, F_t$  covers all assignments, this expectation must be at least 1, so  $t \geq 1/p(f,d)^k$ . ■

As an immediate corollary of the above lemma and proposition 4.1 we get the following bounds on the complexity of the help bits problem:

**Corollary 4.1** *For any boolean function  $f$  on  $n$  variables and integers  $k, l, d \geq 0$ :*

1. If  $2^l \leq 1/p(f,d)^k$  then  $C^{k,l}(f) > d$ .
2. If  $2^l \geq nk/p(f,d)^k$  then  $C_{\text{pure}}^{k,l}(f) \leq d$ .

Next we need to connect the quantity  $p(f,d)$  to the sign-degree  $\text{deg}^s(f)$ .

**Proposition 4.2** *For any boolean function  $f$ ,  $p(f,d) > 1/2$  if and only if  $d > \text{deg}^s(f)$ .*

**Proof.** Let  $d > \text{deg}^s(f)$ . Then there is an  $n$ -variate polynomial  $p(x_1, x_2, \dots, x_n)$  of degree at most  $d$  such that  $g(\alpha) > 0$  if and only if  $f(\alpha) = 1$ . By shifting

the polynomial by a small constant we may assume that  $g(\alpha)$  is never 0. We may assume without loss of generality that the sum of the absolute values of the coefficients of  $g$  is 1. Consider the following randomized decision tree algorithm: choose a monomial of  $g$  at random, where the probability a given monomial is chosen is the absolute value of its coefficient. Probe the variables of the monomial and output the product of the values. It is easily seen that for any assignment  $\alpha$ , the probability of correctly evaluating  $f(\alpha)$  minus the probability of incorrectly evaluating  $f(\alpha)$  is equal to  $|g(\alpha)| > 0$  (here we use that our domain is  $\{-1, 1\}$ ). Thus for any  $\alpha$  this algorithm correctly evaluates  $f(\alpha)$  with probability exceeding  $1/2$ .

Now suppose  $p(f,d) > 1/2$ . There must exist a randomized decision tree algorithm  $Q$  on depth  $d$  trees that evaluates  $f(\alpha)$  correctly with probability exceeding  $1/2$ . Now, it is well known, and easy to see (by induction on  $d$ , looking at the two subtrees of the root) that if  $T$  is a decision tree of depth  $d$  on variables  $\{x_1, \dots, x_n\}$  then there is a polynomial  $g_T(x_1, \dots, x_n)$  of degree  $d$  such that  $g_T(\alpha) = T(\alpha)$  for all assignments  $\alpha$ . Define the polynomial  $g(x_1, \dots, x_n)$  to be the sum of  $Q(T)(g_T - 1/2)$  where the sum is over all trees of depth  $d$  and  $Q(T)$  is the probability that  $T$  is selected under the distribution  $Q$ . Then  $g(\alpha_1, \dots, \alpha_n) = \text{Prob}_Q[\tilde{T}(\alpha) = 1] - 1/2$ . By the choice of  $Q$ , this latter term is positive if and only if  $f(\alpha) = 1$ . ■

Theorem 4.1 now follows easily.

**Proof of Theorem 4.1.** By Corollary 4.1,  $C^{k,k-1}(f) \leq \text{deg}^s(f)$  would follow from  $2^{k-1} \geq nk/p(f, \text{deg}^s(f))^k$ . This holds for all sufficiently large  $k$  since  $p(f, \text{deg}^s(f)) > 1/2$ , by Proposition 4.2.

Also, by Corollary 4.1, to show  $C_{\text{pure}}^{k,k-1}(f) > \text{deg}^s(f) - 1$ , it suffices to show  $2^{k-1} < 1/p(f, \text{deg}^s(f) - 1)^k$  for all  $k$ , which follows immediately from the fact, by Proposition 4.2, that  $p(f, \text{deg}^s(f) - 1) = 1/2$ . ■

**Remark 1.** It is interesting to note that, for  $k$  large enough, it is possible to construct to obtain an optimal algorithm in which all of the decision trees have a particularly simple form. The randomized algorithm in the proof of Proposition 4.2 uses only decision trees that correspond to computing monomials of  $g$ . Using this randomized algorithm in the proof of the upper bound of lemma 4.1 the decision trees used in the help-bits algorithm are all of the same form.



**Remark 2.** As noted in the introduction, if  $f$  is the majority function the  $\deg^s(f) = 1$  and so the decision trees used in the optimal algorithm for  $H^{k,k-1}$  for large  $k$  all have depth 1. In the case that  $f$  is the majority function on three variables, Manuel Blum gave the following constructive protocol to solve  $H^{k,k-1}$ . Enumerate the subsets of  $[k]$  having size at least  $2k/3$ . The number of these sets is  $2^{ck}$  for some  $c < 1$ . Fix an encoding of these sets by  $ck$  bits. Now given  $k$  separate inputs to the majority-of-3 function, and imagine the inputs arranged in a  $k \times 3$  array. In each row, at least two of the three entries agree with the majority value, so there is a column in which at least  $2k/3$  of the entries agree with the function value on that column. For the help bits, we ask for the lowest index of such a column (requiring 2 bits) and then the set  $S$  of rows for which this column gives the function value (requiring  $ck$  bits.) Armed with this information, the value of the function on row  $r$  is equal to the entry in that row and the designated column if  $r \in S$  and is the negative of the entry otherwise.

**Remark 3.** In the proof of the lower bound in Lemma 4.1, we used Theorem 3.1 in order to deduce that for any forest  $F$  of depth at most  $d$ , the probability with respect to a particular distribution  $P$  on assignments  $F$  is correct for all  $k$  functions is at most  $p(f, d)^k$ . In the special case  $d = \deg^s(f) - 1$ , which is the relevant case for proving that  $C^{k,k-1} > \deg^s(f) - 1$  in Theorem 4.1, there is an alternative argument. We sketch this argument, which has the benefit that it extends to other models besides decision trees, as will be seen in the next section. As noted above, for  $d = \deg^s(f) - 1$ , we have  $p(f, d) = 1/2$ , and thus for  $\tilde{\alpha}$  selected from  $\hat{D}$  (the distribution of Lemma 4.2) any decision tree of depth  $d$  agrees with  $f$  with probability exactly  $1/2$ . In particular, this can be shown to imply that if we fix the values of any  $d$  variables then either that partial assignment occurs with probability 0 under  $\hat{D}$ , or that the value of  $f$  conditioned on this assignment is unbiased.

Now, define the random variable  $c_i$  to be 0 if  $T_i(\tilde{\alpha}) = f_i(\tilde{\alpha})$  and 1 otherwise. We want to show that the probability that  $c_i = 0$  for all  $i$  is at most  $1/2^k$ . In fact, the distribution on  $(c_1, c_2, \dots, c_k)$  is uniform on  $\{0, 1\}^k$ . By the XOR lemma of [Vaz] (see also [CGHFRS]) a distribution over  $\{0, 1\}^k$  is uniform if for any subset  $J$  of  $[k]$ , the random variable  $c_J$  defined to be the XOR of the  $c_i$  for  $i \in J$  is unbiased. Let  $s_J$  be the probability that  $c_J = 0$ . The event  $c_J = 0$  is the same as the event that  $T_J(\tilde{\alpha}) (= \prod_{i \in J} T_i(\tilde{\alpha}))$  is equal to  $f_J(\tilde{\alpha}) (= \prod_{i \in J} f_i(\tilde{\alpha}))$ . Now by combining

the decision trees  $\{T_i | i \in J\}$  we can get a single decision tree of depth at most  $|J|d$  that computes  $T_J$ . We claim that such a decision tree must agree with  $f_J$  with probability exactly  $1/2$ , which is enough to finish the argument. We prove the claim by showing that for each leaf of the tree  $T_J$  that is reached with nonzero probability,  $f_J(\tilde{\alpha})$  conditioned on  $\tilde{\alpha}$  reaching the leaf is unbiased. For each such leaf of the tree, there is an  $i \in J$  such that at most  $d$  variables of  $X_i$  appear on the path. Recall that the value of  $f_i$  is unbiased when conditioned on the values of these  $d$  variables. If we further condition the value of  $f_J$  by the values of all variables not in  $X_i$ , then  $f_i$  is still unbiased and therefore so is  $f_J$ .

**Remark 4.** One implication of Theorem 4.1 is that for large enough  $k$ , the best algorithm for  $H^{k,k-1}(f)$  uses pure trees. It is reasonable to speculate that this is the case for  $H^{k,l}(f)$  for all  $k$  and  $l$ , and this is open. For the case  $k = 2$ , it is interesting to note that for the case  $k = 2$  and  $l = 1$ , it is not hard to show that pure tree algorithm can not do better than  $C(f)$ , the ordinary decision tree complexity of  $f$ . To see this, note that the help bit partitions the set of assignments of  $X = X_1 \cup X_2$  into two groups  $A_1$  and  $A_2$ . It is not hard to see that either the set of assignments on  $X_1$  induced by  $A_1$  is all of  $B^{X_1}$ , or the set of assignments on  $X_2$  induced by  $A_2$  must be all of  $B^{X_2}$ . In the first case, then given  $A_1$ , a pure tree computation for  $f$  on  $X_1$  is as hard as the problem without the help bits, and in the second case, then given  $A_2$ , a pure tree computation for  $f$  on  $X_2$  is as hard as the problem without the help bits.

## 5 Other Models

Some of the ideas used so far are also relevant to other models of computation. We can get results for these models that are similar to but neither as precise or as strong as what we obtain for decision trees. It is convenient to describe our results in the following very general framework. We fix some computational model for computing a function  $f$  on input  $\alpha \in X$ , and some class,  $FEAS$ , of “feasible” algorithms.

Our results will only hold for classes having certain closure properties. A class  $FEAS$  is *closed under  $k$ -counting* if for any  $k$  algorithms in  $FEAS$ , any algorithm that runs all  $k$  of these algorithms on the input and accepts or rejects based on the number of computations out of  $k$  that accept, is also in  $FEAS$ . Examples of such classes are polynomial size circuits, which are closed under poly-counting, and polylog-bit

communication complexity protocols which are closed under polylog-counting.

From such a class we define when a multi-input algorithm is feasible. An algorithm for computing a function  $f$  on a pair of inputs  $\alpha_1, \alpha_2 \in D^2$  is said to be *rectangularly-feasible*, in  $FEAS_*$ , if for every fixed value of  $\alpha_1$  the induced algorithm for  $f$  is in  $FEAS$ , and for every fixed value of  $\alpha_2$  the induced algorithm for  $f$  is in  $FEAS$ . Notice that for the two examples mentioned above (and essentially any model one may think of),  $FEAS \subset FEAS_*$ . Thus, for example, for the case of poly-size circuits, the lower bounds given below for two-input algorithms apply to all poly-size circuits as well.

## 5.1 Products

A product theorem in such a setting may be proven using Yao's XOR-lemma [Y2], which we observe applies in this general setting. Let  $D_1, D_2$  distributions, and denote  $p_1 = q_{D_1}(f_1; FEAS)$ ,  $p_2 = q_{D_2}(f_2; FEAS)$ .

**Lemma 5.1** (Yao) *Assume that  $FEAS$  is closed under  $k$ -counting. Then*

$$q_{D_1 \times D_2}(f_1(\alpha_1) \oplus f_2(\alpha_2); FEAS_*) \leq p_1 p_2 + (1 - p_1)(1 - p_2) + 1/k^{\Omega(1)}$$

From this one can deduce an "approximate product theorem".

**Theorem 5.1** *Assume that  $FEAS$  is closed under  $k$ -counting. Then*

$$q_{D_1 \times D_2}(f_1, f_2; FEAS_*) \leq p_1 p_2 + 1/k^{\Omega(1)}.$$

**Proof.** Fix an algorithm  $A$  in  $FEAS_*$ , and denote by  $p_{YY}$  the probability that it is correct on both inputs, by  $p_{NN}$  the probability that it is incorrect on both, by  $p_{YN}$  the probability that it is correct only on the first input and by  $p_{NY}$  the probability that it is correct only on the second input. Since for every fixed value of  $\alpha_1$  the probability that  $A$  is correct on  $f_2$  is at most  $p_2$ , then by averaging over all  $\alpha_1$ , we have  $p_{YY} + p_{NY} \leq p_2$ . Similarly,  $p_{YY} + p_{YN} \leq p_1$ . Finally, Yao's xor-lemma implies  $p_{YY} + p_{NN} \leq p_1 p_2 + (1 - p_1)(1 - p_2) + 1/k^{\Omega(1)}$ . These inequalities, together with the fact that  $p_{YY} + p_{YN} + p_{NY} + p_{NN} = 1$ , directly imply  $p_{YY} \leq p_1 p_2 + 1/k^{\Omega(1)}$ , which proves the lemma. ■

## 5.2 Help Bits

We can use the approximate product theorem to get help-bit results for randomized algorithms. Given a

class of "feasible algorithms"  $FEAS$ , We say that a function is *randomly feasibly computable*, in  $RFC$ , if there exists a probability distribution on algorithms in  $FEAS$  such that for any input, an algorithm chosen from this distribution will be correct on  $f$  with probability of at least  $2/3$ . The constant  $2/3$  is not important as the usual "amplification" lemmas work in this general case.

**Lemma 5.2** *If  $FEAS$  is closed under  $k$ -counting then the constant  $2/3$  can be replaced by  $1/2 + 1/k$  (or by  $2^{-k}$ ) without changing the class  $RFC$ .*

For the case where  $FEAS$  is the class of polynomial size circuits, it is known that randomization does not increase power, and thus  $RFC$  is exactly equal to the functions computable by deterministic poly-size circuits. For the case where  $FEAS$  is polylog-bit communication protocols,  $RFC$  is the functions computable by randomized polylog-bit protocols with two-sided error.

Let us define what is feasible computation with a help-bit. Let  $FEAS$  be a given class of algorithms. A 1-help-bit-feasible algorithm, in  $FEAS^1$ , is a set of two algorithms  $A_0, A_1$  in  $FEAS$ , and a boolean function  $h$ , whose value on input  $\alpha$  is the output of  $A_{h(\alpha)}$ . A function is in  $RFC^1$  if there is a  $FEAS^1_*$  algorithm for computing two copies of  $f$ , which on every pair of inputs is correct on both with probability of at least  $2/3$ . We then can prove a randomized help-bit theorem.

**Theorem 5.2** *If  $FEAS$  is closed under  $O(1)$ -counting then  $RFC^1 = RFC$ .*

**Proof.** Assume that  $f \notin RFC$  then, amplifying and similarly to lemma 4.2, there exists a distribution  $D$  such that  $q_D(f; FEAS) \leq 0.51$ . Using the approximate product theorem, any  $FEAS_*$  algorithm for two copies of  $f$  can be correct on at most  $0.51^2 + o(1)$  fraction of inputs (under distribution  $D \times D$ ). It follows that any  $FEAS^1_*$  algorithm can be correct with probability at most twice that, a probability smaller than  $2/3$  (again probability taken over a pair of inputs chosen from  $D \times D$ .) This in turn implies that  $f \notin RFC^1$ . ■

For the case of boolean circuits, this was proven in [ABG].

## 5.3 The log $k$ Barrier

The "approximate product" theorem and the "randomized help-bit" theorem can be naturally general-

ized to up to  $\log k$  functions where the family  $FEAS$  is closed under  $k$ -counting. After that, these techniques break down. It is unknown for example whether a polynomial size circuit using  $n$  help-bits can compute  $n+1$  copies of function which doesn't have polynomial size circuits. One can show that in a black box model, alternatively, relative to a particular oracle, that the generalizations are *false* using  $\omega(\log k)$  functions.

Consider the model of polynomial-size circuits each with access to the same black-box.

**Theorem 5.3** *There is a black-box so that there exists a Boolean function  $f$  which can't be computed by a polynomial-sized circuit family, but  $l(n) = \omega(\log n)$  help-bits will allow a polynomial-sized circuit to always compute the answer to  $n$  disjoint copies of  $f$ , where  $n$  is the input size to  $f$ .*

**Proof.** It is well known that a random  $f$  can't be computed by a polynomial-sized circuit. Fix such an  $f$ . A successful circuit would take inputs  $X_1, X_2, \dots, X_k$  and output the vector  $V = \langle f(X_1), f(X_2), \dots, f(X_k) \rangle$ . We "hide"  $V$  in the black-box in such a way that a circuit without help-bits can't find it, but a circuit with help-bits goes directly to it. Let  $n$  be the size of each  $X_i$  and choose  $k = n$ . For each input tuple, and output  $V$  do the following: Let  $s$  be a random  $l(n)$ -bit string. Place  $V$  in the location indexed by  $X_1, X_2, \dots, X_n, s$ . For  $t \neq s$ , place a "SORRY" in location  $X_1, X_2, \dots, X_n, t$ . By a standard counting argument, one can show that no polynomial-sized circuit family (with access to the black box) can answer correctly on all  $n$ -tuples of inputs. However, given  $l$  help-bits, it is easy to query the oracle at the location revealing the answer tuple. ■

It is interesting to note that the Yao XOR lemma fails relative to this black-box in the sense that once we XOR more than  $l(n)$  variables the parity stops getting harder to compute. In other words, the XOR lemma has the same  $\log n$  barrier as above.

**Acknowledgement.** The authors have had many conversations with several people regarding this research. We would especially like to acknowledge the contributions of Richard Beigel, Nati Linial, Russell Impagliazzo, and Avi Wigderson.

## References

- [ABG] A. Amir, R. Beigel, W. Gasarch, *Some connections between bounded query classes and nonuniform complexity*, Proceedings 5th Conference on Structure in Complexity Theory, 1990.
- [Bsh] N.H. Bshouty, *On the extended direct sum conjecture*, Proc. 21st ACM Symp. on Theory of Computing, 1989, pp. 177-185.
- [Cai] J. Cai, *Lower bounds for constant depth circuits in the presence of help bits*, Proc. 30th IEEE Symp. on Foundations of Computer Science, 1989, pp. 532-537, 1989.
- [CGHFRS] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, R. Smolensky, *The bit extraction problem of  $t$ -resilient functions*, Proc. 26th IEEE Symp. on Foundations of Computer Science, 1985, 396-407.
- [FKN] T. Feder, E. Kushilevitz, M. Naor, *Amortized Communication Complexity*, Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, pp. 239-248.
- [GF] G. Galbati, M. J. Fischer, *On the complexity of 2-output Boolean networks*, TCS. 16, 1981, pp. 177-185.
- [HW] J. Hastad, A. Wigderson, *Composition of the Universal Relation*, in "Advances in Computational Complexity Theory", AMS-DIMACS book series, to appear.
- [J] J. Ja'Ja', J. Takche *On the Validity of the Direct Sum Conjecture*, SIAM J. Comput. 15 (4), 1986, pp. 1004-1020.
- [KKN] M. Karchmer, E. Kushilevitz, N. Nisan, *Fractional Covers and Communication Complexity*, in Proc. 7th Structures in Complexity Theory Conference, 1992, pp. 262-274.
- [KRW] M. Karchmer, R. Raz, A. Wigderson, *On Proving Super-Logarithmic Depth Lower Bounds via the Direct Sum in Communication Complexity*, Proc. 6th Conference on Structures in Complexity Theory, 1991, pp. 299-304.
- [IRW] R. Impagliazzo, R. Raz, A. Wigderson, *A Direct Product Theorem*, Proc. 9th IEEE Conference on Structure in Complexity Theory, 1994, to appear.
- [Lin] Nathan Linial, personal communication.
- [NW] N. Nisan, A. Wigderson, *Rounds in Communication Complexity Revisited*, SIAM Journal on Computing, 22, (1) 1993.

- [Paul] W. J. Paul, *Realizing Boolean functions on disjoint set of variables*, TCS. 2, 1978, pp. 383-396.
- [RR] A. Razborov, S. Rudich, *Natural Proofs*, Proceedings of the twenty-sixth annual ACM symposium on the theory of computing, 1994, pp. 204-213.
- [Ulig] D. Ulig, *On the synthesis of self-correcting schemes from functional elements with a small number of reliable components*, Math Notes Acad. Sci. USSR. 15, 1974, pp. 558-562.
- [Vaz] U. Vazirani, *Randomness, adversaries and computation*, Ph.D. Thesis, UC Berkeley, 1986.
- [Y1] A. Yao, *Theory and applications of trapdoor functions*, Proc. 23rd Annual IEEE Symp. on Foundations of Computer Science, 1982, 80-91.
- [Y2] A. Yao, *Probabilistic computations: towards a unified measure of complexity*, Proc. 18th Annual IEEE Symp. on Foundations of Computer Science, 1977, 222-227.



**Lecturer:** Alexander A. Razborov

**Title:** Independence Results in Bounded Arithmetic  
& Natural Proofs

**Material:** Slides &  
Alexander A. Razborov: *Unprovability of Lower Bounds on  
Circuit Size in Certain Fragments of Bounded Arithmetic*

Steklov Mathematical Institute  
Vavilova 42, 117966 GSP-1  
Moscow, Russia  
E-mail: razborov@mian.su

# Independence Results in Bounded Arithmetic

&  
Natural Proofs\*

Alexander A. Razborov

Steklov Mathematical Institute

Moscow, Russia

\* This part represents a joint work with Steven Rudich (Carnegie Mellon, Pittsburg)

## I. Motivations

72

Uniform models	Non-uniform models
Diagonalization techniques	More direct methods: Combinatorial, algebraic etc.
Succeeded in the classical recursion theory	Succeeded in proving lower bounds for restricted models
Was faltering for a while in extending this to the polynomial world	Was faltering for a while in extending this to stronger, "unrestricted" models
Inherent reasons: relativizing proofs (Baker, Gill, Solovay [75]).	Inherent reasons: natural proofs (Razborov, Rudich [94])
	??? provability in Bounded Arithmetic

The life goes on...

Uniform models

Non-uniform models

non-relativizing proofs:

Lund, Fortnow, Karloff, Nisan,  
Shamir [92]

Balai, Fortnow, Lund [91]

.....

transparent proofs

???

Decision problems: recognize whether  $x \in L$   
where  $L \subseteq \{0,1\}^*$  is a language

Uniform ( $\approx$  Turing) complexity:

a machine  $M$  should decide whether  $x \in L$   
uniformly for all possible  $x \in \{0,1\}^*$

Non-uniform ( $\approx$  Boolean) complexity:

a sequence  $C_1, C_2, \dots, C_n, \dots$  of circuits should be  
designed so that  $C_n$  is good only for  $x \in \{0,1\}^n$

Technicalities:

$$L_n = L \cap \{0,1\}^n$$

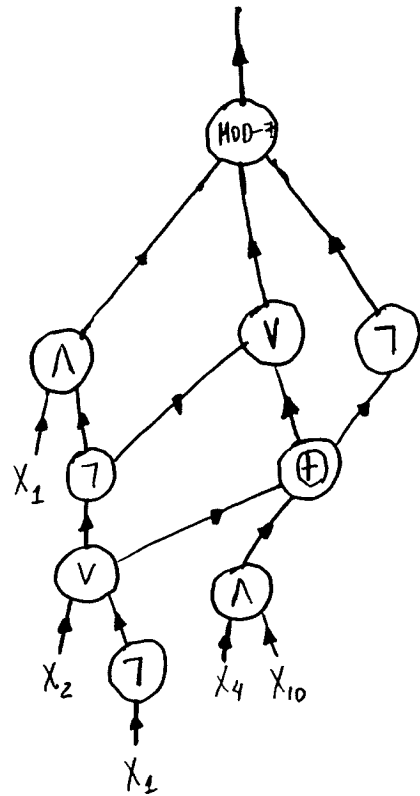
$f_n: \{0,1\}^n \rightarrow \{0,1\}$  is the characteristic

function of  $L_n$

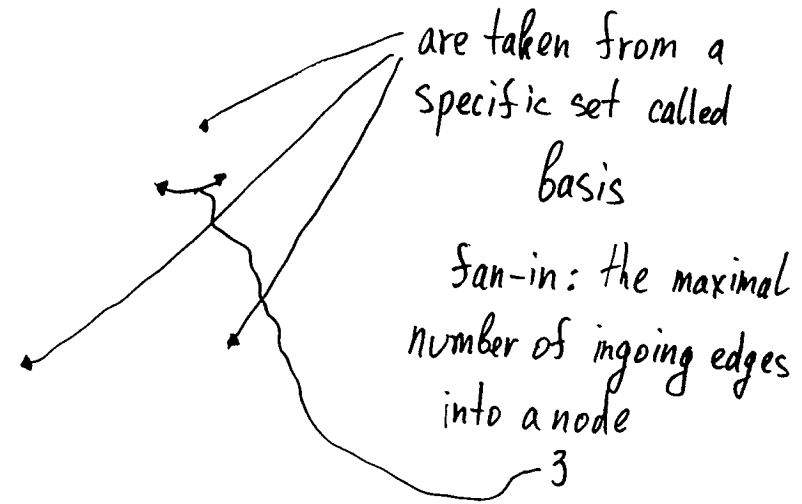
Mappings of this form are called  
Boolean functions



# II. A Bit of Boolean Complexity.



Local constraints



Global goals



Size: = the number  
of nodes  $g$

depth: = the length of  
the longest path  $5$

# 1.2.

Connections with <sup># 1.3, # 1.3a, # 1.3b.</sup> Turing complexity

# 1.4

$\mathcal{C}$  - a complexity class

$\mathcal{C}/\text{poly}$  -  $\mathcal{C}$ -machines + polynomial size advice  
depending only on the length of the input

$\mathcal{C}/\text{poly} \Rightarrow \{A \subseteq \{0,1\}^* \mid \exists L \in \mathcal{C} \exists (a_1, \dots, a_n, \dots)$   
 $|a_n| \leq \text{poly}(n) \ \& \ (x \in A \Leftrightarrow x \# a_{|x|} \in L)\}$

polynomial size circuits =  $P/\text{poly}$

logarithmic depth circuits =  $NC^1/\text{poly}$

polylogarithmic depth and  
polynomial size =  $NC/\text{poly}$

In this talk we are exclusively interested in non-uniform models.

Diagonalization does not (apparently) apply to non-uniform models (there are continuously many circuits  $C_i = (C_{i,1}, \dots, C_{i,n}, \dots)$ ...)

Combinatorial and algebraic machinery is comfortable with models described as simply as possible, and a Boolean circuit is the most primitive thing in the world.

76  
A brief survey of Bounded Arithmetic

$L_1$  - the language of Peano Arithmetic (plus  $|x|, \lfloor \frac{1}{2}x \rfloor$ )

$L_2 \stackrel{\text{def}}{=} L_1 + \{\#\}$  [Buss 86].

$$x \# y \stackrel{\text{def}}{=} 2^{|x| \cdot |y|}$$

The idea: if  $t(x_1, \dots, x_n)$  is a term of  $L_2$  then

$$|t(x_1, \dots, x_n)| \leq \text{poly}(|x_1| + \dots + |x_n|).$$

Bounded quantifiers:  $(\forall x \leq t)$   $(\exists x \leq t)$

Sharply bounded quantifiers:  $(\forall x \leq |t|)$   $(\exists x \leq |t|)$

Bounded formula: only bounded quantifiers occur

The hierarchy  $\Sigma_i^b, \Pi_i^b$ : sharply bounded quantifiers are allowed for free.

The central system  $S_2^1$  #2.12.  
 The induction is on  $\Sigma_1^k$   
 The language is  $L_1$

$$\text{BASIC}_2 + \sum_1^b \text{-PIND}$$

$$\text{PIND-free} \quad \sum_1^b \text{-LIND}$$

$$\left[ A(0) \wedge \forall x (A(x) \supset A(x+1)) \right] \Rightarrow \forall x A(|x|).$$

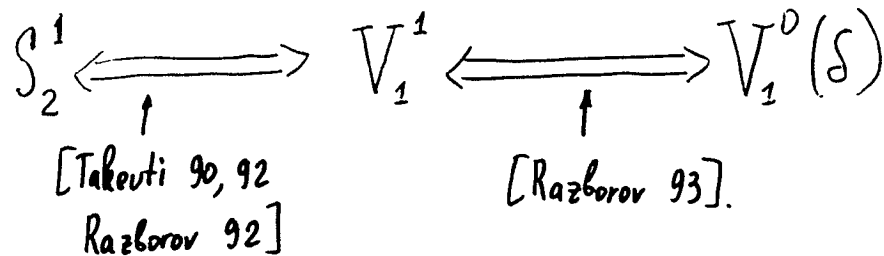
Warning:  $\forall x A(|x|)$  and  $\forall x A(x)$  are different  
 since B.A. does not prove that the exponentiation is total.

Theorem [Buss 86] Assume that  $S_2^1 \vdash \exists x \leq t A(a_1, \dots, a_n, x)$   
 where  $A(a_1, \dots, a_n, b) \in \sum_1^b$ . Then there is a polynomial  
 time computable  $f(a_1, \dots, a_n)$  s.t.

$$\mathbb{N} \models A(a_1, \dots, a_n, f(a_1, \dots, a_n)).$$

And vice versa.

Putting some cosmetics on...



RSUV-isomorphism

$V_1^0(\mathcal{L})$ : the basic language is  $L_1$

Append it with a new predicate variable  $\mathcal{L}$ .

Append  $L_1(\mathcal{L})$  with new predicate variables  $\delta_A$   
 having defining axioms (A is bounded)

$$\forall x \left( \delta_A(x) \equiv A(\mathcal{L}, x, \delta_A(0), \dots, \delta_A(x-1)) \right) \quad (\text{DEF})$$

The notation is sloppy.

$V_1^0(\mathcal{L})$ :  $\text{BASIC}_1 + (\text{DEF}) + \text{induction on all bounded}$   
 formulae containing  $\mathcal{L}, \delta_A$ . \(\equiv\)

## Krajicek's Formalization.

The size of the objects we are working with is polynomial in  $n$ .

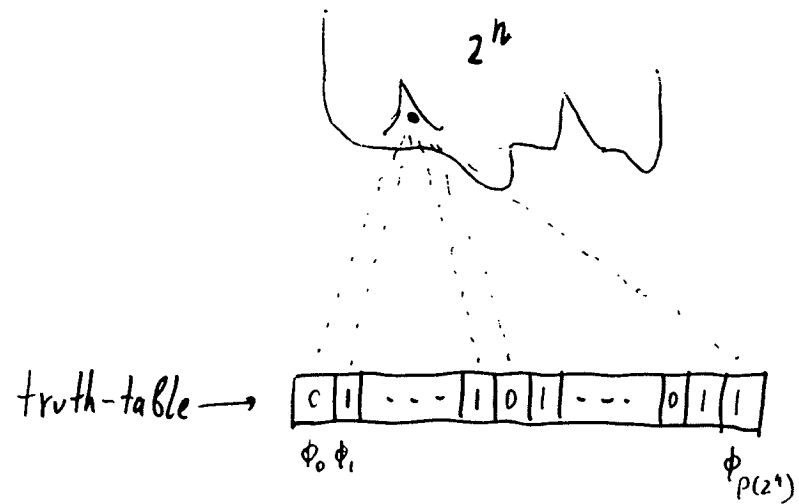
$\phi \in \text{SAT}$ :  $\exists \mathcal{I} \forall C (C \text{ is a clause in } \phi) \exists \text{ a literal } l \text{ in } C \text{ such that } \mathcal{I} \text{ satisfies } l$ .

$k > 0$  is fixed.

$\forall (\text{circuit } C \text{ of size } n^k) \exists \phi \text{ such that } C(\phi) \neq \text{SAT}(\phi)$ .

The problem with this: we (apparently) can do to little...

The solution: go to skies:



but not too high...

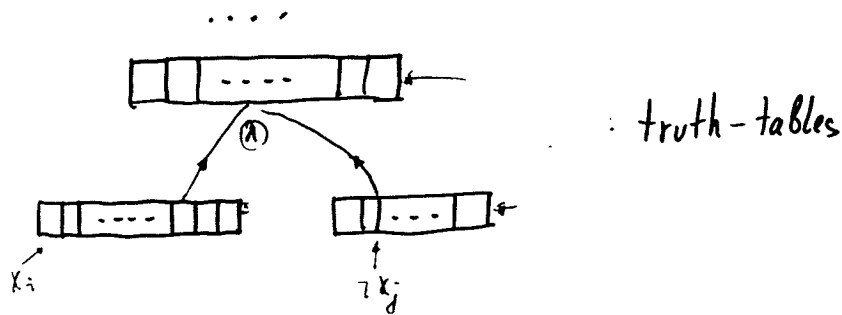
$N = 2^n$  is our main parameter.

Reason 1. We do not need more in the existing proofs.

Reason 2. I do not have a least idea how to handle proofs using objects of size, say,  $2^{2^n}$ .

# Formalization. (optional).

$\alpha$  is the most primitive encoding of Boolean circuits:



Circuit  $(t, N, \alpha)$  -  $\alpha$  is a legitimate picture like this with  $t$  nodes;  $N = 2^n$  is the length of the truth-table  
 Output  $(t, N, x, \alpha)$  - the output of  $\alpha$  on a string  $x$

$$\text{Circuit}(t(N), N, \alpha) \supset \exists x \in \{0,1\}^{|N|} (\text{Output}(t(N), N, x, \alpha) \neq \text{SAT}_{|N|}(x)).$$

$$\forall N \exists t(N) \geq (\log N)^{\omega(1)}$$

CHALLENGE. Find a proof for an explicit function in non-uniform complexity which can not be done in  $V_1^1 = V_1^0(\delta)$

Good News. Counting arguments are presumably beyond the strength of  $V_1^1$ .

Direction for research. Can  $V_1^1$  prove superpolynomial lower bounds for general circuits or not?

Hope. We understand metamathematics of  $V_1^1$  much better than that of PA or ZF, and the concept responsible for this metamathematics is polynomial size circuits!!!

# Natural Proofs

(joint work with Steven Rudich)

$$N = 2^n$$

$\mathcal{A}$  is a class of circuits

$G_n$  is a predicate on  $\{0,1\}^N =$  Boolean functions in  $n$  variables.

$G_n$  is natural against  $\mathcal{A}$  iff:

Usefulness: for every choice of  $f_n \in \mathcal{A}$ ,  $\{f_n\} \notin \mathcal{A}$

Constructivity:  $G_n$  is polynomially time (in  $N$ ) computable.

Largeness:  $\text{Prob}(G_n(f_n) = 1) \geq 2^{-O(n)} (= N^{-O(1)})$

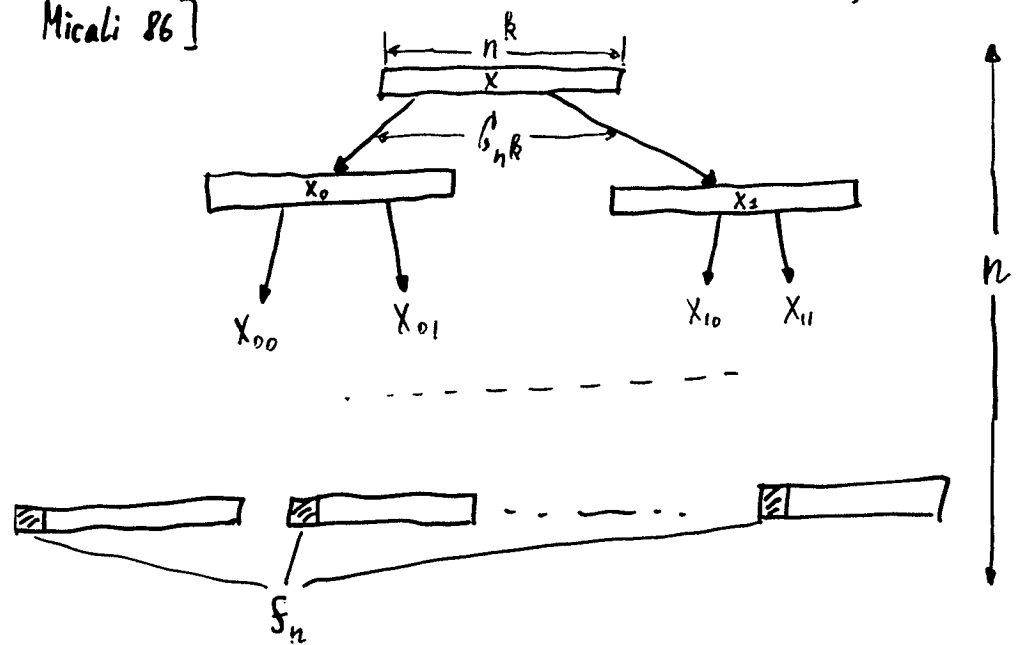
CHALLENGE. Find a proof for an explicit function in non-uniform complexity which does not lead to  $G_n$  with these properties.

Hardness assumption. There exist a polynomial time computable pseudorandom number generator  $G_n: \{0,1\}^n \rightarrow \{0,1\}^{2n}$  and  $\epsilon > 0$  such that  $G_n$  is secure against attack by  $2^{n^\epsilon}$ -size circuits  $C$ :

$$|\text{Prob}(C(G_n(x)) = 1) - \text{Prob}(C(y) = 1)| \leq 2^{-n}$$

Theorem. Under this assumption there is no natural proof against general circuits

Proof: variation on the theme of [Goldreich, Goldwasser, Micali 86]



Good news. Standard hardness assumptions in principle allow us to rule out the existence of  $2^{o(n)}$ -size circuits with sunny properties.

$t(n)$  is a superpolynomially growing function

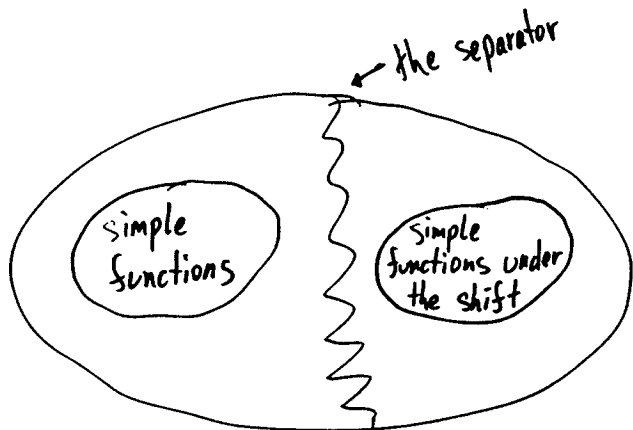
$\mathcal{U}$  - the set of (truth-tables of) simple functions (with circuit size  $\leq t(n)$ ).  $\mathcal{U} = \text{NP}$

Fact. If  $S_n$  is a complex Boolean function then

$$\mathcal{U} \cap (\mathcal{U} \oplus S_n) = \emptyset$$

Under the same hardness assumption  $\mathcal{U}$  and  $\mathcal{U} \oplus S_n$  can not be separated by a (quasi)polynomial time algorithm.

Proof.



Do there exist two disjoint NP-sets  $(\mathcal{U}, \mathcal{V})$  which can not be separated by a (quasi)polynomial algorithm?

This is equivalent to the existence of cryptosystems secure against worst-case attack

It is (apparently) open whether there exists a pair  $(\mathcal{U}, \mathcal{V})$  complete for the class of all disjoint NP-pairs.

Def.  $(\mathcal{U}, \mathcal{V})$  is representable in a theory  $\mathcal{T}$  if

$\mathcal{T} \vdash (w_u \text{ is a witness to } u \in \mathcal{U}) \ \& \ (w_v \text{ is a witness to } v \in \mathcal{V}) \Rightarrow u \neq v.$

Observation. If  $\mathcal{T}$  proves superpolynomial lower bounds for  $S_n$  then  $(\mathcal{U}, \mathcal{U} \oplus S_n)$  is representable in  $\mathcal{T}$ .



The question. Is it true that every NP-pair representable in  $\mathbb{T}$  can be separated by a (quasi)polynomial time algorithm?

Good news. For "decent" theories  $\mathbb{T}$  of Bounded Arithmetic the class of NP-pairs representable in  $\mathbb{T}$  does contain pairs complete w.r.t. (quasi)polynomial many-one reductions.

$P$  - a proof system for propositional calculus

$l_P$  - the length of the optimal proof in  $P$ .

$$\text{SAT}' = \{ \langle \phi, 1^t \rangle \mid \phi \in \text{SAT} \}$$

$$V_P = \{ \langle \phi, 1^t \rangle \mid l_P(\phi) \leq t \}$$

$(\text{SAT}', V_P)$

## The table.

88

Theory	Proof system	Type of reducibility
$S_2^1(d)$	resolutions with every clause containing $\leq \log n$ literals	quasipolynomial
$S_2^3(d)$	resolutions	quasipolynomial
$I\Delta_0(d)$	bounded-depth Frege systems	polynomial
$U_1^1$	Frege systems	quasipolynomial
$V_1^1$	extended Frege systems	polynomial

Fact. If  $P$  are resolutions with every clause containing  $\leq \log n$  literals then  $V_P$  is decidable in quasipolynomial time.

# Unprovability of Lower Bounds on Circuit Size in Certain Fragments of Bounded Arithmetic

Alexander A. Razborov\*  
School of Mathematics  
Institute for Advanced Study  
Princeton, NJ 08540

and

Steklov Mathematical Institute  
Vavilova 42, 117966, GSP-1  
Moscow, RUSSIA

To appear in *Izvestiya of the RAN*

## Abstract

We show that if strong pseudorandom generators exist then the statement “ $\alpha$  encodes a circuit of size  $n^{(\log^* n)}$  for SATISFIABILITY” is not refutable in  $S_2^2(\alpha)$ . For refutation in  $S_2^1(\alpha)$ , this is proven under the weaker assumption of the existence of generators secure against the attack by small depth circuits, and for another system which is strong enough to prove exponential lower bounds for constant-depth circuits, this is shown without using any unproven hardness assumptions.

These results can be also viewed as direct corollaries of interpolation-like theorems for certain “split versions” of classical systems of Bounded Arithmetic introduced in this paper.

---

\*Supported by the grant # 93-6-6 of the Alfred P. Sloan Foundation and by the grant # 93-011-16015 of the Russian Foundation for Fundamental Research

## 1. Introduction

Proving lower bounds on the complexity of explicitly given Boolean functions is one of the most challenging tasks in the computational complexity. This theory met with a remarkable success at least twice: in the 60's (see e.g. [34, 29, 30, 35, 36]) and in more recent time ([11, 1, 26, 12, 31, 32, 27, 2, 24, 28, 33, 21, 4, 15, 17]). Both times, however, the period of enthusiasm was followed by understanding that it is not quite clear to which extent the methods developed so far can be useful for attacking central open problems in Boolean complexity.

A logical analysis of this situation should start with understanding what is the right "minimal" fragment of *ZFC* which is really needed for formalizing all these methods, and this question was raised in [19]. It was argued there that the conceivable answer is the second order theory of Bounded Arithmetic  $V_1^1$ , and no example of a lower bound for explicit function not provable in  $V_1^1$  has been found since that. The next goal is to develop machinery for understanding whether  $V_1^1$  can prove superpolynomial lower bounds on the size of unrestricted circuits or not.

In this paper we present first partial results in this direction. Namely, we show that the existence of a pseudorandom generator secure against the attack by circuits of size  $2^{n^\epsilon}$  (for some fixed  $\epsilon > 0$ ) implies that for any explicit Boolean function  $f_n$  and any integer-valued  $t(n)$  such that  $t(n) \geq n^{\omega(1)}$ , the theory  $S_2^2(\alpha)$  can not refute that  $\alpha$  encodes a Boolean circuit of size  $t(n)$  for  $f_n$ . For the theory  $S_2^1(\alpha)$  the same statement holds under the weaker assumption of the existence of a generator secure against  $n^\epsilon$ -depth circuits.

A few remarks concerning these results should be made immediately.

- Following [19], we work in the strongest possible framework in which  $\alpha$  includes encodings of *truth-tables* of all Boolean functions appearing in the circuit as intermediate results.
- We do *not* require that Bounded Arithmetic would *prove*  $t(n) \geq n^{\omega(1)}$ , we only need this to be true on integers. Thus, our results are still applicable to e.g.  $t(n) = n^{\log^* n}$ .
- Since we are mostly interested in the provability in  $V_1^1$ , this is also natural to consider the hierarchy of its subtheories and wonder whether we can do better for them. The strongest theory in this hierarchy to which our method applies is  $IE_1(f)$  (see [25] for the definition of  $IE_1$ ), and for this theory we indeed can prove a slightly stronger result. Namely, we may replace  $t(n)$  by  $n^k$  for a *fixed* constant  $k > 0$  depending only on the quality of the generator. This improvement, however, is really marginal, so we prefer to work all the time in the language  $L_2$  containing the smash function  $\#$ .

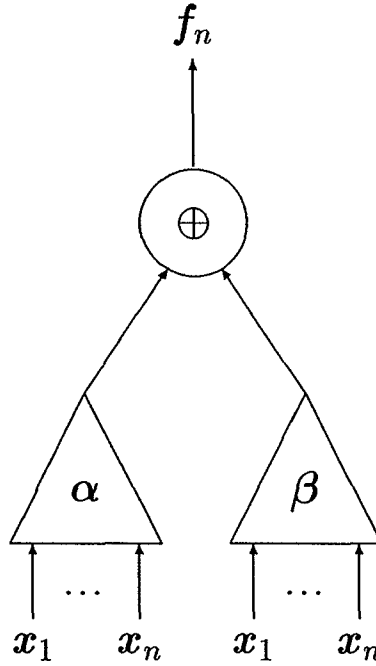


Figure 1: The framework for split versions

For proving these results we define the *split* version  $\mathcal{S}(S_2)$  of  $S_2$  as the theory in the language  $L_2(\alpha, \beta)$  which allows induction on arbitrary bounded formulae in  $L_2(\alpha)$  and arbitrary bounded formulae in  $L_2(\beta)$ . We consider the pair  $(\alpha, \beta)$  as an encoding of a Boolean circuit with the PARITY gate at the top so that  $\alpha$  encodes the left-hand side of the rest, and  $\beta$  encodes the right-hand side (see Figure 1).

$\mathcal{S}(S_2)$  proves in this framework exponential lower bounds on the size of constant-depth circuits over the standard basis. We show that on the other hand it can not prove super-polynomial lower bounds for depth-3 circuits with PARITY gates. We derive the above-mentioned results about  $S_2^1(\alpha)$  and  $S_2^2(\alpha)$  as direct consequences of similar statements concerning  $\mathcal{S}(S_2)$  appended with the corresponding induction schemes.

The proofs consist of several fairly independent pieces. One of essential ingredients is the characterization of the circuit depth by a communication game [15], and a characterization of the circuit size in these terms based upon local search problems (Theorem 3.1 of this paper). These characterizations are non-uniform in their very nature, and this suggests

that our results might be extended to stronger theories allowing more computational power for both players.

To this end we define the split version  $\mathcal{S}(V_2)$  of the second order theory  $V_2$  in the same fashion as  $\mathcal{S}(S_2)$ , and extend our three results to this theory (appended with the appropriate induction scheme for the first two). These extensions follow from general interpolation-like theorems, and this is a close indication that  $\mathcal{S}(V_2)$  and its extensions *exactly* capture Karchmer-Wigderson game and its analogue for the circuit size. Unfortunately, these second order versions are somewhat technical. Thus, for the convenience of the reader interested only in classical fragments of Bounded Arithmetic, we start with the simpler first order case.

The paper is organized as follows. In Section 2 we recall necessary definitions from Complexity Theory. In Section 3 we present the new characterization of the circuit size (Theorem 3.1). In Section 4 we briefly survey results from Bounded Arithmetic needed for our purposes. In Section 5 we recall the framework from [19] and introduce its split variant. In Section 6 we present first order versions of our main results, and in Section 7 show that they can be actually derived as corollaries of interpolation-like theorems for split versions of second order theories. The paper is concluded by some remarks and open problems in Section 8.

## 2. Background from Complexity Theory

In this section we recall necessary definitions and facts from Complexity Theory.

### 2.1. Boolean Complexity

We address the reader to [5] for an excellent treatment of the subject; the sole purpose of this section is to agree upon notation.

We denote by  $F_n$  the set of all Boolean functions in  $n$  variables  $x_1, \dots, x_n$ . Let  $x_i^1 \equiv x_i$  and  $x_i^0 \equiv (\neg x_i)$ . Most of the time, it will be convenient to think of  $f_n \in F_n$  as of a binary string of length  $2^n$  called the *truth-table* of  $f_n$ . We will denote by  $S(f_n)$  the circuit size of  $f_n$  (over the standard basis  $\{\wedge, \vee, \neg\}$  with negations appearing only at variables; all computational nodes must have fan-in 2).  $D(f_n)$  is the minimal depth needed for computing  $f_n$  in the same model.  $S_{mon}(f_n)$  and  $D_{mon}(f_n)$  are, respectively, the monotone circuit size and the monotone depth of a monotone  $f_n$ .  $S_d(f_n)$  is the circuit size with respect to depth- $d$  (unbounded fan-in) circuits.  $S_d^\oplus(f_n)$  is the same as  $S_d(f_n)$ , only now we additionally allow PARITY gates.

$SIZE(t(n))$  is the complexity class consisting of all functions  $\{f_n\}$  for which  $S(f_n) \leq O(t(n))$ ;  $DEPTH(d(n))$  has a similar meaning. The notation  $DEPTH, SIZE(d(n), t(n))$  and  $DEPTH, SIZE^\oplus(d(n), t(n))$  corresponds to unbounded fan-in circuits with simultaneous restrictions  $d(n)$  on their depth and  $O(t(n))$  on their size.  $DEPTH, SIZE(O(1), n^{O(1)})$  is the (non-uniform) class  $AC^0$ ,  $DEPTH, SIZE^\oplus(O(1), n^{O(1)})$  will be denoted by  $AC^0[2]$ , and  $DEPTH, SIZE^\oplus(d, n^{O(1)})$  will be denoted by  $AC^{0,d}[2]$ .

All these complexity measures can be in a natural way extended to the case of *partial* Boolean functions  $f_n : \{0, 1\}^n \rightarrow \{0, 1, *\}$  (\* stands for “undefined”). E.g.  $S(f_n)$  for a partial  $f_n$  is the minimum of  $S(\bar{f}_n)$  taken over all total extensions  $\bar{f}_n$  of  $f_n$  etc.

## 2.2. Karchmer-Wigderson game

This game was introduced in [15].

Let  $U, V, I$  be finite sets, and  $R \subseteq U \times V \times I$  be a ternary relation such that

$$\forall u \in U \forall v \in V \exists i \in I ((u, v, i) \in R). \quad (1)$$

Assume that we have two players with unlimited computational power. Let player I receive  $u \in U$ , and player II receive  $v \in V$ . Their common task is to find some  $i \in I$  such that  $(u, v, i) \in R$  exchanging messages between each other. The minimal number of bits (taken over all possible protocols achieving this goal) to be exchanged in the worst case is called the *communication complexity* of  $R$  and denoted by  $C(R)$ .

Now, for a (possibly, partial) Boolean function  $f_n$  in  $n$  variables consider the relation  $R_{f_n} \subseteq f_n^{-1}(0) \times f_n^{-1}(1) \times [n]$  given by  $R_{f_n} \equiv \{(u, v, i) \mid u_i \neq v_i\}$ . If  $f_n$  is monotone (that is, has at least one total monotone extension in  $F_n$ ), define also its monotone analogue  $R_{f_n}^{mon}$  by  $R_{f_n}^{mon} \equiv \{(u, v, i) \mid u_i = 0, v_i = 1\}$ .

**Proposition 2.1** ([15]). a) For every (partial) Boolean function  $f$ ,  $C(R_f) = D(f)$ ,

b) For every (partial) monotone Boolean function  $f$ ,  $C(R_f^{mon}) = D_{mon}(f)$ .

Denote by  $C_d(R)$  the modification of  $C(R)$  in which only  $d$  rounds are allowed. The following is a slightly refined version of the result implicitly contained in [14, Definition 3.5.2]:

**Proposition 2.2.** For every (partial) Boolean function  $f$  and every  $d > 0$ ,

$$2^{\left(\frac{C_d(R_f)}{d} - 1\right)} \leq S_d(f) \leq 2^{C_d(R_f)}.$$

### 2.3. Polynomial local search problems

This concept was originally considered in [13]. We reproduce here the variant of the definition given in [8].

**Definition 2.3.** A *local search problem*  $L$  consists of a set  $F_L(x) \subseteq \mathbf{N}$  of *solutions* for every instance  $x \in \mathbf{N}$ , an integer-valued *cost function*  $c_L(s, x)$  and a *neighborhood function*  $N_L(s, x)$  such that:

- a)  $0 \in F_L(x)$ ;
- b) for all  $s \in F_L(x)$ ,  $N_L(s, x) \in F_L(x)$ ;
- c) for all  $s \in F_L(x)$ , if  $N_L(s, x) \neq s$  then  $c_L(s, x) < c_L(N_L(s, x), x)$ .

A *local optimum* for the problem  $L$  on  $x$  is an  $s$  such that  $s \in F_L(x)$  and  $N_L(s, x) = s$ . A local search problem  $L$  is *polynomial* if the binary predicate  $s \in F_L(x)$  and the functions  $c_L(s, x), N_L(s, x)$  are polynomially time computable, and also there exists a polynomial  $p_L(n)$  such that  $|s| \leq p_L(|x|)$  for all  $s \in F_L(x)$ .

Note that the concept of a polynomial local search (PLS) problem can be relativized in a standard way.

### 2.4. Natural proofs

This concept was introduced in [20].

Let  $\Gamma$  and  $\Lambda$  be complexity classes. Slightly altering the notation from [20], we call a sequence  $\{C_n \mid n \in \omega\}$  of subsets  $C_n \subseteq F_n$  a  $\Gamma$ -*natural combinatorial property useful against*  $\Lambda$  if it satisfies the following three conditions:

**Constructivity:** The predicate  $f_n \overset{?}{\in} C_n$  is computable in  $\Gamma$  (note that the bit size of an input to this problem is  $2^n$  which will be denoted further on by  $N$ ),

**Largeness:**  $|C_n| \geq 2^{-O(n)} \cdot |F_n|$ ,

**Usefulness:** For any sequence of functions  $f_n \in C_n$ ,  $\{f_n\} \notin \Lambda$

(our  $C_n$  corresponds to  $C_n^*$  from [20]). A lower bound proof that some explicit function is not in  $\Lambda$  is called  $\Gamma$ -*natural against*  $\Lambda$  if it leads to a  $\Gamma$ -natural combinatorial property which is useful against  $\Lambda$ .

For a pseudo-random generator  $G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$  define its *hardness*  $H(G_n)$  as the minimal  $S$  for which there exists a circuit  $C$  of size  $\leq S$  with the property

$$|\mathbf{P}[C(G_n(\mathbf{x})) = 1] - \mathbf{P}[C(\mathbf{y}) = 1]| \geq \frac{1}{S}. \quad (2)$$

Here  $\mathbf{x}$  is taken at random from  $\{0, 1\}^n$ , and  $\mathbf{y}$  is taken at random from  $\{0, 1\}^{2n}$ .

The following is a minor improvement on [20, Theorem 4.1] which is proved in the same way:

**Proposition 2.4.** *Assume that there exists a  $SIZE(2^{(\log N)^{O(1)}})$ -natural combinatorial property which is useful against  $P/poly$  ( $= SIZE(n^{O(1)})$ ). Then for every polynomial time computable  $G_k : \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$ ,  $H(G_k) \leq 2^{k^{O(1)}}$ .*

We define *depth hardness*  $DH(G_n)$  of  $G_n$  as the minimal  $S$  for which there exists a circuit  $C$  of depth  $\leq \log_2 S$  such that (2) holds. The following is analogous to Proposition 2.4:

**Proposition 2.5.** *Assume that there exists a  $DEPTH((\log N)^{O(1)})$ -natural combinatorial property which is useful against  $P/poly$ . Then for every polynomial time computable  $G_k : \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$ ,  $DH(G_k) \leq 2^{k^{O(1)}}$ .*

Note that the classes  $SIZE(2^{(\log N)^{O(1)}})$ ,  $DEPTH((\log N)^{O(1)})$  appearing in the above two propositions are simply non-uniform analogues of quasipolynomial time and  $POLY\text{-LOGSPACE}$ , respectively.

Finally, we improve along the same lines upon [20, Theorem 4.3]:

**Proposition 2.6.** *There is no  $DEPTH, SIZE(O(1), 2^{(\log N)^{O(1)}})$ -natural combinatorial property useful against  $AC^{0,3}[2]$ .*

### 3. A new characterization of circuit size

Let  $U, V, I$  be finite sets, and  $R \subseteq U \times V \times I$  be a ternary relation such that (1) holds. We will be considering those local search problems whose instances  $x$  are (encodings of) pairs  $(u, v)$ ;  $u \in U, v \in V$ .

For any such problem  $L = \langle F_L, c_L, N_L \rangle$ , let  $C(F_L, c_L)$  be the communication complexity of computing simultaneously the predicate  $s \in F_L(u, v)$  and the function  $c_L(s, u, v)$



in the model when the first player gets  $(s, u)$ , and the second gets  $(s, v)$  (thus,  $s$  is in the public domain).  $C(N_L)$  is defined similarly. The *size* of  $L$ , by definition, is

$$\left| \bigcup_{\substack{u \in U \\ v \in V}} F_L(u, v) \right| \cdot 2^{2C(F_L, c_L) + C(N_L)}$$

(the meaning of the coefficient 2 in front of  $C(F_L, c_L)$  will become clear from the proof of Theorem 3.1).

We say that  $R$  *reduces* to  $L$  if there exists a function  $p : \mathbf{N} \rightarrow I$  such that for any  $(u, v) \in U \times V$  and any local optimum  $s$  for  $L$  on  $(u, v)$ , we have  $(u, v, p(s)) \in R$ . We define  $size(R)$  as

$$\min \{ size(L) \mid R \text{ reduces to } L \}.$$

**Theorem 3.1.** a) For every partial Boolean function  $f$ ,  $size(R_f) = \theta(S(f))$ ,

b) For every monotone partial Boolean function  $f$ ,  $size(R_f^{mon}) = \theta(S_{mon}(f))$ .

**Proof.** Since the proofs of the two parts are practically identical, we prove only part a).

Let  $f$  be a partial Boolean function in  $n$  variables, let  $t \equiv S(f)$ , and let  $C$  be a size- $t$  circuit computing  $f$ . Denote  $f^{-1}(0)$  by  $U$ , and  $f^{-1}(1)$  by  $V$ . We want to reduce  $R_f$  to a local search problem  $L$  of size  $O(t)$ . Disregarding all inessential variables not appearing in  $C$ , we may assume w.l.o.g. that

$$t \geq n - 1. \tag{3}$$

We arrange nodes  $w_1, \dots, w_t$  of the circuit  $C$  in such a way that a wire can go from  $w_\mu$  to  $w_\nu$  only when  $\mu < \nu$ . Let  $f_\nu$  be the function computed at  $w_\nu$ . Note for the record that  $f_t$  is an extension of  $f$  that is  $f_t(u) = 0, f_t(v) = 1$  for all  $u \in U, v \in V$ .

We construct  $L$  as follows. Encode nodes  $w_1, \dots, w_t$  by integers  $n_1, \dots, n_t$  so that  $n_t = 0$  and  $\{1, \dots, n\} \cap \{n_1, \dots, n_t\} = \emptyset$ . Let

$$\begin{aligned} F_L(u, v) &\equiv \{i \mid 1 \leq i \leq n \ \& \ u_i \neq v_i\} \cup \{n_\nu \mid 1 \leq \nu \leq t \ \& \ f_\nu(u) = 0 \ \& \ f_\nu(v) = 1\}, \\ c_L(i, u, v) &\equiv 0 \text{ for } 1 \leq i \leq n, \\ N_L(i, u, v) &\equiv i \text{ for } 1 \leq i \leq n, \\ c_L(n_\nu, u, v) &\equiv \nu \text{ for } 1 \leq \nu \leq t. \end{aligned}$$

$N_L(n_\nu, u, v)$  is defined as follows. If  $n_\nu \notin F_L(u, v)$ , let  $N_L(n_\nu, u, v) \equiv 0$ . Otherwise, that is when  $f_\nu(u) = 0$  and  $f_\nu(v) = 1$ , we choose one of the two sons of the node  $w_\nu$  for which this

property is preserved. If this son is a computational node  $w_\mu$ , we let  $N_L(n_\nu, u, v) \rightleftharpoons n_\mu$ ; if this is a leaf  $x_i^\epsilon$ , we let  $N_L(n_\nu, u, v) \rightleftharpoons i$ .

It is straightforward to check that so defined  $L$  is a local search problem, and that  $R_f$  reduces to  $L$ . Also,  $C(F_L, C_L) \leq 2$  and  $C(N_L) \leq 3$ . Hence  $\text{size}(L) \leq O(n + t)$  which is  $O(t)$  due to (3).

For another (non-trivial) direction, assume that  $R_f$  reduces via a function  $p$  to a local search problem  $L$ . Let  $h_0 \rightleftharpoons 2^{C(F_L, C_L)}$  and  $h_1 \rightleftharpoons 2^{C(N_L)}$ . Then for every fixed  $s \in \bigcup_{u \in U, v \in V} F_L(u, v)$  we have a communication protocol  $P_s$  for computing the binary relation  $s \in F_L(u, v)$  and the cost function  $c_L(s, u, v)$  which has at most  $h_0$  different histories. These histories define a partition of  $U \times V$  into rectangles  $U_{s,1} \times V_{s,1}; \dots; U_{s,h_0} \times V_{s,h_0}$  such that  $F_L, c_L$  are fully determined on  $U_{s,i} \times V_{s,i}$ . That is to say, for some predicates  $\alpha_s \subseteq [h_0]$  and some functions  $\eta_s : [h_0] \rightarrow \mathbb{N}$  the following is true for all  $i \in [h_0]$  and for all  $(u, v) \in U_{s,i} \times V_{s,i}$ :

$$s \in F_L(u, v) \text{ iff } i \in \alpha_s$$

and

$$c_L(s, u, v) = \eta_s(i).$$

We call those rectangles  $U_{s,i} \times V_{s,i}$  for which  $i \in \alpha_s$  *good*. We call  $\eta_s(i)$  the *cost* of rectangle  $U_{s,i} \times V_{s,i}$ . We order all good rectangles in such a way that their costs are non-decreasing:

$$U^1 \times V^1; \dots; U^{H_0} \times V^{H_0}.$$

Here  $H_0 \leq \left| \bigcup_{u \in U, v \in V} F_L(u, v) \right| \cdot h_0$ .

We construct by induction on  $\nu \leq H_0$  a circuit  $C_\nu$  which has the following property. For every  $\mu \leq \nu$  there exists a node  $w_\mu$  of  $C_\nu$  computing a function  $f_\mu$  such that  $f_\mu|_{U^\mu} \equiv 0$  and  $f_\mu|_{V^\mu} \equiv 1$ . Assume that we already have  $C_{\nu-1}$ .  $C_\nu$  will be obtained from it by adding at most  $h_0 h_1$  new nodes for computing a  $f_\nu$  with required properties from already available  $f_1, \dots, f_{\nu-1}$ .

Let  $U^\nu \times V^\nu = U_{s,i} \times V_{s,i}$ . Consider the following communication protocol  $P_s^*$  of complexity at most  $C(F_L, C_L) + C(N_L)$ . First we run the optimal protocol for computing  $N_L(s, u, v)$ . Let  $s' \rightleftharpoons N_L(s, u, v)$  be its outcome. Then we run  $P_{s'}$ .

We introduce Boolean variables  $y_1, \dots, y_H$  for those histories of  $P_s^*$  which actually correspond to at least one instance  $(u, v) \in U_{s,i} \times V_{s,i}$ . For every  $u \in U_{s,i}$  let  $\bar{u}$  be the assignment on  $\{0, 1\}^H$  defined by letting  $\bar{u}_h$  be 0 if there exists  $v \in V_{s,i}$  such that the computation of  $P_s^*$  on  $(u, v)$  develops according to the history  $h$ , and 1 otherwise. Dually,  $\bar{v}_h = 1$  iff there exists  $u \in U_{s,i}$  so that the pair  $(u, v)$  leads to the history  $h$ . For every pair  $(u, v) \in U_{s,i} \times V_{s,i}$

we have  $\bar{u}_h = 0, \bar{v}_h = 1$ , where  $h$  is the history of  $P_s^*$  corresponding to this pair. Hence, the partial Boolean function  $\hat{f}_\nu(y_1, \dots, y_H)$  outputting 0 on  $\{\bar{u} \mid u \in U_{s,i}\}$ , outputting 1 on  $\{\bar{v} \mid v \in V_{s,i}\}$  and undefined elsewhere, is monotone, and, moreover, the protocol  $P_s^*$  finds a solution to  $R_{\hat{f}_\nu}^{mon}$ . Hence, by Proposition 2.1 b),  $D_{mon}(\hat{f}_\nu) \leq C(F_L, c_L) + C(N_L)$ , and the same bound holds for some total monotone extension  $\bar{f}_\nu$  of  $\hat{f}_\nu$ . Note for the record that this implies  $S_{mon}(\bar{f}_\nu) \leq h_0 h_1$ .

Consider now a particular history of  $P_s^*$ ,  $h$ . Let  $(s', j)$  be the corresponding output (here  $s'$  is the output of computing  $N_L$ , and  $j$  is the subhistory corresponding to the subprotocol  $P_{s'}$ ). By Definition 2.3 b), the rectangle  $U_{s',j} \times V_{s',j}$  is good. By part c) of this definition, either  $s' = s$  or the cost of  $U_{s',j} \times V_{s',j}$  is strictly less than the cost of  $U_{s,i} \times V_{s,i}$ .

In the first case  $s$  is a local optimum for  $L$  on every  $(u, v) \in U_{s,i} \times V_{s,i}$  belonging to the non-empty rectangle which corresponds to  $h$ . Since  $R_f$  reduces to  $L$ , this means that  $u_{p(s)} \neq v_{p(s)}$  for every such pair, and this implies that actually  $u_{p(s)} = \epsilon, v_{p(s)} = (\neg\epsilon)$  for some fixed  $\epsilon \in \{0, 1\}$ . Let  $y'_h \equiv x_{p(s)}^{(\neg\epsilon)}$ .

In the second case  $U_{s',j} \times V_{s',j} = U^\mu \times V^\mu$  for some  $\mu < \nu$ . Let  $y'_h \equiv f_\mu$ .

Finally, let  $f_\nu \equiv \bar{f}_\nu(y'_1, \dots, y'_H)$ .  $f_\nu$  can be computed by appending to  $C_{\nu-1}$  at most  $h_0 h_1$  new nodes.

Since for every  $u \in U^\nu$ ,  $\bar{f}_\nu(\bar{u}_1, \dots, \bar{u}_H) = 0$ , and  $\bar{f}_\nu$  is monotone, in order to check that  $f_\nu(u) = 0$  for  $u \in U^\nu$ , we only have to check that  $y'_h(u) \leq \bar{u}_h$  for any history  $h$ . For doing this simply note that if  $\bar{u}_h = 0$ , then for some  $v \in V^\nu$  the computation on  $(u, v)$  proceeds along  $h$ , which, due to our choice of  $y'_h$ , implies  $y'_h(u) = 0$ . By the dual argument,  $f_\nu(v) = 1$  for all  $v \in V^\nu$ .

This completes the construction of  $C_\nu$ .

Now,  $C_{H_0}$  has size at most  $H_0 h_0 h_1$ . Also, due to Definition 2.3 a), all rectangles  $U_{0,i} \times V_{0,i}$  are good. Thus, applying the same argument as above and adding to  $C_{H_0}$  at most  $h_0$  new nodes, we finally compute  $f$  by a circuit of size  $O(\text{size}(L))$ . This completes the proof of Theorem 3.1. ■

## 4. Background from Bounded Arithmetic

We assume the familiarity with [6] and use the now-standard notation for denoting various hierarchies and fragments of Bounded Arithmetic from that book. We denote by  $L_2$  Buss's first order language which consists of the constant 0, function symbols  $S, +, \cdot, \lfloor \frac{1}{2}x \rfloor, |x|, x \# y$  and of the predicate symbol  $\leq$ .  $BASIC_2$  is the set of 32 open axioms in the language  $L_2$  from [6, §2.2] describing basic properties of its symbols.  $\Sigma^b \equiv \bigcup_{i \geq 0} \Sigma_i^b$  is the set of all

(first-order) bounded formulae of  $L_2$ .

In [19] a convenient technical notion of a regular theory was introduced. The meaning of this notion is that many proofs in Bounded Arithmetic which do not involve the smash function  $\#$  can be generalized to arbitrary regular theories. In this paper we need a stronger notion which is good also for  $\#$ -involving proofs.

**Definition 4.1.** A first order theory  $R$  in a language  $L \supseteq L_2$  is *strongly regular* if it possesses the following properties:

- a)  $BASIC_2 \subseteq R$ ,
- b)  $R$  can be axiomatized by  $\Sigma_0^b$ -formulae,
- c) every function symbol (and hence every term) of the language  $L$  can be bounded from above in the theory  $R$  by a term of the language  $L_2$ .

For a strongly regular theory  $R$  in a language  $L$  we denote by  $S_R^i$  the theory  $R + \Sigma_i^b(L) - PIND$ , and by  $T_R^i$  the theory  $R + \Sigma_i^b(L) - IND$ . Let also  $S_R = \bigcup_{i=0} S_R^i$ ; this is the same theory as  $T_R = \bigcup_{i=0} T_R^i$ .

If  $L = L_2$  and  $R = BASIC_2$  then  $S_R^i$  is simply  $S_2^i$ , and  $T_R^i$  is  $T_2^i$ . Another important example is  $L = L_2(\gamma)$ ,  $R = BASIC_2$  ( $\gamma$  is a new predicate variable). In this case  $S_R^i$  and  $T_R^i$  coincide with ordinary theories  $S_2^i(\gamma)$  and  $T_2^i(\gamma)$ . A less trivial example is provided by  $L = L_{PV}$ ,  $R = "BASIC_2 + \Pi_1^b$ -defining axioms for  $PV$ -symbols" (see [6, §6.2]), where  $PV$  is Cook's equational system [10]. In this case  $S_R^1$  is the theory  $S_2^1(L_{PV})$  as defined in [6]. One more example of this sort will be given in Section 6.

As we already mentioned, the meaning of this definition is that many (if not all) results proven for  $S_2^i$ ,  $T_2^i$  relativize to arbitrary strongly regular theories  $R$ . For example, the (weaker form of) the main theorem from [6] in this setting looks like this:

**Proposition 4.2.** *Let  $R$  be a strongly regular theory in a language  $L$  extending  $L_2$ . Suppose  $S_R^1 \vdash \exists y A(\vec{a}, y)$ , where  $A(\vec{a}, b)$  is a  $\Sigma_1^b(L)$ -formula with all its free variables displayed. Then there is a polynomial time oracle Turing machine  $M$  allowed to ask queries of the form  $\vec{n} \stackrel{?}{\in} P$  or  $f(\vec{n}) = ?$ , where  $P$  is a predicate symbol of  $L \setminus L_2$ , and  $f$  is a function symbol of  $L \setminus L_2$ , such that the following holds.*

*For every model  $(\mathbf{N}, \Omega)$  of the theory  $R$  expanding the standard model of  $BASIC_2$  and every tuple  $\vec{n} \in \mathbf{N}$ ,*

$$(\mathbf{N}, \Omega) \models A(\vec{n}, M^\Omega(\vec{n})).$$

*Here  $\Omega$  is the interpretation of symbols from  $L \setminus L_2$ , and  $M^\Omega(\vec{n})$  is the result of the computation of  $M$  on  $\vec{n}$  when  $M$  is fed with the oracle  $\Omega$ .*

We also need the following conservation result from [7]:

**Proposition 4.3.** *For any strongly regular theory  $R$  in a language  $L \supseteq L_2$ ,  $S_R^2$  is  $\Sigma_2^b(L)$ -conservative over  $T_R^1$ .*

Finally, we recall the characterization of  $\Sigma_1^b$ -defined in  $T_2^1$  functions in terms of PLS-problems [8]. Once again, we present the relativized version.

**Proposition 4.4.** *Let  $R$  be a strongly regular theory in a language  $L \supseteq L_2$ . Suppose  $T_R^1 \vdash \exists y A(a, y)$ , where  $A(a, b)$  is a  $\Sigma_1^b(L)$ -formula with all its free variables displayed. Then there is an oracle PLS-problem  $K$ , where the associated oracle computations of  $F_K, c_K, N_K$  are allowed to ask queries of the form  $\vec{n} \stackrel{?}{\in} P$  or  $f(\vec{n}) = ?$ ;  $P, f$  being symbols of  $L \setminus L_2$ , and  $a$  a (polynomial-time computable) function  $p(s)$  such that the following holds.*

*For every model  $(\mathbf{N}, \Omega)$  of the theory  $R$  expanding the standard model of  $BASIC_2$ , every  $x \in \mathbf{N}$ , and every local optimum  $s$  for  $K^\Omega$  on  $x$ ,*

$$(\mathbf{N}, \Omega) \models A(x, p(s)).$$

## 5. Boolean Complexity and Bounded Arithmetic: split framework

In our formalization of problems studied in Boolean complexity within the framework provided by Bounded Arithmetic we follow [19, Appendix A]. Namely, let  $Circuit(t, N, \gamma)$  be a  $\Sigma^b(\gamma)$ -formula asserting that  $\gamma$  encodes the protocol of computation by a circuit of size  $t$  in  $|N|$  variables. Similarly, for a fixed  $d > 0$ , let  $Circuit_d(t, N, \gamma)$  and  $Circuit_d^\oplus(t, N, \gamma)$  assert that  $Circuit(t, N, \gamma)$  and, moreover,  $\gamma$  is a depth- $d$  circuit or depth- $d$  circuit with PARITY gates, respectively. Let  $Output(t, N, x, \gamma)$  be a  $\Sigma^b(\gamma)$ -formula which represents the output of  $\gamma$  (viewed as a circuit of size  $t$  in  $|N|$  variables) on a Boolean string  $x$ . The exact details of these encodings are unimportant; the only extra property which we require (and which is shared by all reasonable schemes) is that we can easily combine in this framework two circuits to compute PARITY of their outputs as shown on Figure 1. More precisely, we require that there exists a  $\Delta_1^b(\alpha, \beta)$  (with respect to  $S_2^1(\alpha, \beta)$ ) abstract

$PARITY(t, N, \alpha, \beta)$  such that

$$\left. \begin{aligned} S_2^1(\alpha, \beta) \vdash & \left( Circuit(\lfloor (t-3)/4 \rfloor, N, \alpha) \wedge Circuit(\lfloor (t-3)/4 \rfloor, N, \beta) \right) \supset \\ & \left( Circuit(t, N, PARITY(t, N, \alpha, \beta)) \wedge \forall x \in \{0, 1\}^{|N|} \right. \\ & \left. \left( Output(\lfloor (t-3)/4 \rfloor, N, x, \alpha) \oplus Output(\lfloor (t-3)/4 \rfloor, N, x, \beta) \equiv \right. \right. \\ & \left. \left. Output(t, N, x, PARITY(t, N, \alpha, \beta)) \right) \right) \end{aligned} \right\} (4)$$

Like in [19], we are mostly interested in the provability of the formula

$$Circuit(t(N), N, \gamma) \supset \exists x \in \{0, 1\}^{|N|} (Output(t(N), N, x, \gamma) \neq S(N, x)), \quad (5)$$

where  $t(N)$  is a  $\Sigma^b$ -definable function such that  $\mathbf{N} \models t(N) \geq (\log N)^{\omega(1)}$ , and  $S(N, a)$  is in  $\Sigma^b$ . (5) asserts that there is no circuit of size  $t(N)$  (remember that  $N \approx 2^n$ ) computing the Boolean function  $\{x\}S(N, x)$ ; we denote this formula by  $LB(t, S, \gamma)$ .  $LB_d(t, S, \gamma)$  and  $LB_d^\oplus(t, S, \gamma)$  are obtained from  $LB_d(t, S, \gamma)$  after replacing  $Circuit(t, N, \gamma)$  by  $Circuit_d(t, N, \gamma)$  and  $Circuit_d^\oplus(t, N, \gamma)$ , respectively.

One of the main results of this paper (Corollary 6.5) says that if sufficiently strong pseudorandom generators exist, then  $S_2^2(\gamma) \not\vdash LB(t, S, \gamma)$  for *any* choice of  $t, S$  with the above properties. We can, however, prove a stronger result at the same cost and better explain the mechanism of the proof if we split our circuit into two pieces as shown on Figure 1. The corresponding statement, denoted by  $SLB(t, S, \alpha, \beta)$  is

$$\begin{aligned} & (Circuit(t(N), N, \alpha) \wedge Circuit(t(N), N, \beta)) \supset \\ & \exists x \in \{0, 1\}^{|N|} (Output(t(N), N, x, \alpha) \oplus Output(t(N), N, x, \beta) \neq S(N, x)). \end{aligned}$$

$SLB_d(t, S, \alpha, \beta)$  and  $SLB_d^\oplus(t, S, \alpha, \beta)$  have the obvious meaning.

We are going to allow unlimited reasoning about each of the two halves  $\alpha, \beta$  alone. In this and the next section we do as much as we can within the first order framework, and, with this restriction, we implement our idea as follows.

Denote by  $\mathcal{S}(L_2)$  the language  $L_2(\alpha, \beta)$  obtained from  $L_2$  by appending to it two new unary predicate variables  $\alpha$  and  $\beta$ , and define the *split hierarchy*  $\mathcal{S}\Sigma_i^b, \mathcal{S}\Pi_i^b$  of bounded formulae in this language similarly to the ordinary hierarchy  $\Sigma_i^b, \Pi_i^b$  (see [6, §2.1]) with the exception of the base case. Namely,  $\mathcal{S}\Sigma_0^b = \mathcal{S}\Pi_0^b$  is the set of *all* bounded formula in the language  $L_2(\alpha)$  *plus* the set of all bounded formulae in  $L_2(\beta)$ . The inductive definition of  $\mathcal{S}\Sigma_{i+1}^b, \mathcal{S}\Pi_{i+1}^b$  is the same as for  $\Sigma_{i+1}^b, \Pi_{i+1}^b$ . Note that  $\mathcal{S}\Sigma_0^b$  is *not* closed under applying

the connectives  $\wedge, \vee$  or sharply bounded quantifiers although all  $\mathcal{S}\Sigma_i^b, \mathcal{S}\Pi_i^b$  for  $i > 0$  are so closed.

Our “base” theory  $\mathcal{S}(S_2)$  is the theory in the language  $\mathcal{S}(L_2)$  with the set of axioms  $BASIC_2 + \mathcal{S}\Sigma_0^b - IND$ . Another, more expressive description of  $\mathcal{S}(S_2)$  (which also justifies the notation) is that it is axiomatized by  $S_2(\alpha) + S_2(\beta)$ .

We conclude this section by showing that  $\mathcal{S}(S_2)$  is already capable of proving some non-trivial lower bounds.

**Theorem 5.1.** *For every fixed  $d \geq 2$ ,*

$$\mathcal{S}(S_2) \vdash SLB_d(t, S, \alpha, \beta),$$

where  $t(N) \doteq \lfloor 2^{\frac{1}{50}|N|^{1/(2d-3)}} \rfloor$  and  $S(N, x) \doteq x_1 \oplus \cdots \oplus x_{|N|}$ .

**Proof.** Arguing informally in  $\mathcal{S}(S_2)$ , let  $\alpha$  and  $\beta$  be depth- $d$  circuits of size at most  $t(N)$ . Since Håstad Switching Lemma is available in  $S_2(\alpha)$  (see [19, Appendix E.4]), we can find a restriction  $\rho$  assigning at least  $\frac{1}{5}|N|^{\frac{d-1}{2d-3}}$  stars and reducing the output of  $\alpha$  to a constant.  $\rho$ , however, is coded by an integer, thus we can apply in  $\mathcal{S}(S_2)$  the same argument to  $\beta|_\rho$  and find an extension  $\rho'$  of  $\rho$  assigning at least two stars and reducing  $\beta$  to a constant as well. Now we take any two adjacent inputs compatible with  $\rho'$ ; one of them will satisfy  $Output(t(N), N, x, \alpha) \oplus Output(t(N), N, x, \beta) \neq x_1 \oplus \cdots \oplus x_{|N|}$ . ■

## 6. Main results: first order versions

Throughout the rest of the paper,  $t(N)$  will stand for a  $\Sigma^b$ -definable in  $S_2$  function such that  $\mathbb{N} \models t(N) \geq (\log N)^{\omega(1)}$ , and  $S(N, a)$  will stand for an arbitrary bounded formula.

We start with our base theory  $\mathcal{S}(S_2)$  and show that it can not prove superpolynomial lower bounds for depth-3 circuits allowing PARITY gates. This, together with Theorem 5.1, provides some formal evidence toward the remark made in [20, Section 3.2] that [33, 21, 4] had to require arguments from a stronger class than those of [11, 26, 12].

**Theorem 6.1.** *For any  $t(N), S(N, a)$  with the above properties,*

$$\mathcal{S}(S_2) \not\vdash SLB_3^\oplus(t, S, \alpha, \beta).$$

The next theory of interest to us is  $\mathcal{S}(S_2) + \mathcal{S}\Sigma_1^b - PIND$ .

**Theorem 6.2.** *Assume that there exists a polynomial time computable generator  $G_k : \{0, 1\}^k \rightarrow \{0, 1\}^{2^k}$  with  $DH(G_k) \geq 2^{k^{\Omega(1)}}$ . Then for any  $t(N), S(N, a)$  as above,*

$$S(S_2) + \mathcal{S}\Sigma_1^b - PIND \not\vdash SLB(t, S, \alpha, \beta).$$

**Corollary 6.3.** *Under the same assumption as in Theorem 6.2,*

$$S_2^1(\alpha) \not\vdash LB(t, S, \alpha).$$

**Proof of Corollary 6.3 from Theorem 6.2.** Assume the contrary, that is  $S_2^1(\alpha) \vdash LB(t, S, \alpha)$ . Substitute in this proof the  $\Delta_1^b(\alpha, \beta)$ -abstract  $PARITY(t(N), N, \alpha, \beta)$  for  $\alpha$ . Then we will have  $S_2^1(\alpha, \beta) \vdash SLB(t', S, \alpha, \beta)$ , where  $t'(N) \equiv \lfloor (t(N) - 3)/4 \rfloor$ . This contradicts Theorem 6.2 (applied to  $t := t'$ ) since  $S_2^1(\alpha, \beta)$  is a subtheory of  $S(S_2) + \mathcal{S}\Sigma_1^b - PIND$ . ■

Our main result is similar to Theorem 6.2.

**Theorem 6.4.** *Assume that there exists a polynomial time computable generator  $G_k : \{0, 1\}^k \rightarrow \{0, 1\}^{2^k}$  with  $H(G_k) \geq 2^{k^{\Omega(1)}}$ . Then for any  $t(N), S(N, a)$  with the properties stated in the beginning of this section,*

$$S(S_2) + \mathcal{S}\Sigma_2^b - PIND \not\vdash SLB(t, S, \alpha, \beta).$$

**Corollary 6.5.** *Under the same assumption as in Theorem 6.4,*

$$S_2^2(\alpha) \not\vdash LB(t, S, \alpha).$$

**Proof** is the same as that of Corollary 6.3. ■

We begin proving these results with a straightforward definition of the *skolemization*  $\widehat{S_2}(\gamma)$  of the theory  $S_2(\gamma)$ . Firstly, we define the language  $\widehat{L_2}(\gamma)$  as the extension of  $L_2(\gamma)$  obtained by recursively appending to it new function symbols  $f_{A,t}(\vec{b})$  for every open formula  $A(a, \vec{b})$  and term  $t(\vec{b})$  of the language  $L_2(\gamma)$ ; all occurrences of free variables in  $A, t$  are explicitly displayed.

$\widehat{S_2}(\gamma)$  is the open theory in the language  $\widehat{L_2}(\gamma)$  axiomatized by  $BASIC_2$  and the following *defining axioms* for  $f_{A,t}$ :

$$\begin{aligned} & \forall \vec{y} \, f_{A,t}(\vec{y}) \leq t(\vec{y}); \\ & \forall x, \vec{y} \, ((x \leq t(\vec{y}) \wedge A(x, \vec{y})) \supset (A(f_{A,t}(\vec{y}), \vec{y}) \wedge f_{A,t}(\vec{y}) \leq x)); \\ & \forall \vec{y} \, (\neg A(f_{A,t}(\vec{y}), \vec{y}) \supset f_{A,t}(\vec{y}) = 0). \end{aligned}$$



Thus, the intended meaning of  $f_{A,t}(\vec{b})$  is simply  $\mu x \leq t(\vec{b})A(x, \vec{b})$ . The following summarizes some easy properties of this theory:

**Lemma 6.6.** a) For every  $A \in \Sigma^b(\gamma)$  there exists  $A' \in \text{Open}(L_2(\widehat{\gamma}))$  such that  $\widehat{S}_2(\gamma) \vdash A \equiv A'$ , and vice versa;

b)  $\widehat{S}_2(\gamma)$  is a strongly regular open extension of  $S_2(\gamma)$  by definitions.

We define the extension  $\widehat{\mathcal{S}}(L_2)$  of  $\mathcal{S}(L_2)$  as  $L_2(\widehat{\alpha}) + L_2(\widehat{\beta})$ , where we assume, of course, that all non-logical symbols in  $L_2(\widehat{\alpha})$  and  $L_2(\widehat{\beta})$  other than those of  $L_2$  are pairwise distinct. Finally, let  $\widehat{\mathcal{S}}(S_2)$  be the theory  $\widehat{S}_2(\alpha) + \widehat{S}_2(\beta)$  in the language  $\widehat{\mathcal{S}}(L_2)$ . The following properties are inherited from Lemma 6.6:

**Lemma 6.7.** a) For every  $A \in \mathcal{S}\Sigma_0^b$  there exists  $A' \in \text{Open}(L_2(\widehat{\alpha})) \cup \text{Open}(L_2(\widehat{\beta}))$  such that  $\widehat{\mathcal{S}}(S_2) \vdash A \equiv A'$ , and vice versa;

b)  $\widehat{\mathcal{S}}(S_2)$  is a strongly regular open extension of  $\mathcal{S}(S_2)$  by definitions. Thus,  $\widehat{\mathcal{S}}(S_2)$  is conservative over  $\mathcal{S}(S_2)$ , and every model of  $\mathcal{S}(S_2)$  has a unique extension to a model of  $\widehat{\mathcal{S}}(S_2)$ .

The following observation provides a crucial link between the theory  $\widehat{\mathcal{S}}(S_2)$  and the communication game from Section 2.2.

**Lemma 6.8.** Let  $s(a_1, \dots, a_r, \alpha, \beta)$  be a term of the language  $\widehat{\mathcal{S}}(L_2)$  with all its free variables displayed. Consider the following communication problem: player I receives  $n_1, \dots, n_r \in \mathbb{N}$  and a language  $A \subseteq \mathbb{N}$ ; player II receives the same  $n_1, \dots, n_r$  and  $B \subseteq \mathbb{N}$ , and they want to compute  $s(n_1, \dots, n_r, A, B)$  in the extension of the model  $(\mathbb{N}, A, B)$  of  $\mathcal{S}(S_2)$  to a model of  $\widehat{\mathcal{S}}(S_2)$ . Then there exists a constant  $d$  depending only on the term  $s$  and a  $d$ -round communication protocol solving this problem whose complexity is polynomial in  $|n_1| + \dots + |n_r|$ .

**Proof.** Obvious induction on the logical depth of  $s$  (every function symbol of the language  $\widehat{\mathcal{S}}(L_2)$  can be evaluated by one of the two players alone, and results of all intermediate evaluations are of polynomial length).■

Now we are ready to prove the results stated in the beginning of this section.

**Proof of Theorem 6.1.** Assume the contrary, that is  $\mathcal{S}(S_2) \vdash SLB_3^\oplus(t, S, \alpha, \beta)$ . Then also  $\widehat{\mathcal{S}}(S_2) \vdash SLB_3^\oplus(t, S, \alpha, \beta)$ . But the theory  $\widehat{\mathcal{S}}(S_2)$  is open, and, by Lemma 6.7

a), the formulae  $Circuit_3^\oplus(t(N), N, \alpha)$ ,  $Circuit_3^\oplus(t(N), N, \beta)$ ,  $Output(t(N), N, x, \alpha)$  and  $Output(t(N), N, x, \beta)$  are equivalent in  $\widehat{\mathcal{S}}(S_2)$  to open formulae. Thus, by Herbrand's theorem, there exist terms  $s_1(N, \alpha, \beta), \dots, s_r(N, \alpha, \beta)$  of the language  $\widehat{\mathcal{S}}(L_2)$  such that

$$\begin{aligned} \widehat{\mathcal{S}}(S_2) \vdash & \left( Circuit_3^\oplus(t(N), N, \alpha) \wedge Circuit_3^\oplus(t(N), N, \beta) \right) \supset \\ & \bigvee_{i=1}^r \left( s_i(N, \alpha, \beta) \in \{0, 1\}^{|N|} \wedge \right. \\ & \left. (Output(t(N), N, s_i(N, \alpha, \beta), \alpha) \oplus Output(t(N), N, s_i(N, \alpha, \beta), \beta)) \neq \right. \\ & \left. S(N, s_i(N, \alpha, \beta))) \right). \end{aligned}$$

Let  $n$  be an integer, and  $N \equiv 2^n - 1$ . By Lemma 6.8, there exists a communication protocol in which the first player receives  $n$  and a depth-3 size- $t(N)$  circuit  $C_1$  in  $n$  variables allowing PARITY gates, the second player receives  $n$  and a circuit  $C_2$  of the same kind, and they produce an input string  $x$  such that

$$C_1(x) \oplus C_2(x) \neq S(N, x) \quad (6)$$

within  $O(1)$  rounds and  $n^{O(1)}$  bits exchanged. For doing this they simply compute

$$s_1(N, C_1, C_2), \dots, s_r(N, C_1, C_2)$$

and find among this list some  $x$  satisfying (6).

But this protocol also gives rise to a similar protocol in which the players, instead of circuits, receive only Boolean functions  $f_1, f_2 \in F_n$  such that  $S_3^\oplus(f_1) \leq t(N)$  and  $S_3^\oplus(f_2) \leq t(N)$ . In fact, the players, using their unlimited power, simply reconstruct some  $C_1, C_2$  computing  $f_1$  and  $f_2$ , respectively, and then run the protocol above.

Let us now consider the partial Boolean function  $\mathcal{F}_n$  in  $2^n$  variables (we will call it a *functional*) which outputs a 1 on  $f$  if  $S_3^\oplus(f) \leq t(N)$ , outputs a 0 if  $S_3^\oplus(f \oplus s_n) \leq t(N)$  (here  $s_n(x) \equiv S(N, x)$ ) and is undefined elsewhere. Then our protocol for every  $f_1, f_2$  such that  $\mathcal{F}_n(f_1) = 1$  and  $\mathcal{F}_n(f_2) = 0$  finds a position  $x$  where  $f_1(x) \neq f_2(x)$  (note that the second player should modify his  $f_2$  to  $f_2 \oplus s_n$  before entering the protocol from the previous paragraph). Hence, by Proposition 2.2, there exists  $E_n \subseteq F_n$  in  $DEPTH, SIZE(O(1), 2^{(\log N)^{O(1)}})$  such that  $\mathcal{F}_n^{-1}(1) \subseteq E_n$  and  $\mathcal{F}_n^{-1}(0) \cap E_n = \emptyset$ . If  $|E_n| \geq \frac{1}{2}F_n$  then  $E_n \oplus s_n$  makes a  $DEPTH, SIZE(O(1), 2^{(\log N)^{O(1)}})$ -natural combinatorial property useful against  $AC^{0,3}[2]$  since  $t(N) \geq n^{\omega(1)}$  and for every  $f_n \in E_n \oplus s_n$  we have the bound  $S_3^\oplus(f_n) > t(N)$ . Otherwise,  $F_n \setminus E_n$  is such a property. We have arrived at a contradiction with Proposition 2.6. ■

**Proof of Theorem 6.2.** Suppose  $\mathcal{S}(S_2) + \mathcal{S}\Sigma_1^b - PIND \vdash SLB(t, S, \alpha, \beta)$ . By Lemma 6.7 a), the class of  $\mathcal{S}\Sigma_1^b$ -formulae is equivalent in  $\widehat{\mathcal{S}}(S_2)$  to the class of  $\Sigma_1^b(\widehat{\mathcal{S}}(L_2))$ -formulae. Denoting  $\widehat{\mathcal{S}}(S_2)$  by  $R$ , we see that  $\widehat{\mathcal{S}}(S_2) + \mathcal{S}\Sigma_1^b - PIND$  is actually equivalent to  $S_R^1$ . In particular,  $S_R^1 \vdash SLB(t, S, \alpha, \beta)$ . But  $R$  is strongly regular by Lemma 6.7 b), hence we can apply to it Proposition 4.2. We find a polynomial time (in  $n$ ) oracle Turing machine  $M$  asking queries which depend either only on  $C_1$  or only on  $C_2$ ;  $C_1, C_2$  being this time size- $t(N)$  circuits, and producing a length  $n$  string  $x$  with the property (6). But the two players, one holding  $(n, C_1)$  and another holding  $(n, C_2)$ , can simulate  $M$  exchanging only  $n^{O(1)}$  bits between each other. Now the proof is completed by the same argument as in the proof of Theorem 6.1 on the base of Propositions 2.1 a) and 2.5. ■

**Proof of Theorem 6.4.** Suppose  $\mathcal{S}(S_2) + \mathcal{S}\Sigma_2^b - PIND \vdash SLB(t, S, \alpha, \beta)$ . Let, once again,  $R \equiv \widehat{\mathcal{S}}(S_2)$ . Then  $\mathcal{S}(S_2) + \mathcal{S}\Sigma_2^b - PIND$  is equivalent to  $S_R^2$ , and  $S_R^2 \vdash SLB(t, S, \alpha, \beta)$ . By Proposition 4.3,  $T_R^1 \vdash SLB(t, S, \alpha, \beta)$ . By Proposition 4.4, there is an oracle PLS-problem  $K$  and a function  $p(s)$  such that for any two circuits  $C_1, C_2$  of size at most  $t(N)$ , and any local optimum  $s$  for  $K^{C_1, C_2}$  on  $N$ ,  $p(s)$  is a binary string  $x$  of length  $n$  for which (6) holds.

Now we change our view and consider  $C_1, C_2$  simply as extra inputs to  $K$  rather than as oracles, and let  $K_n$  be its subproblem obtained by fixing  $n$  to a particular value. Then the relation  $R_{\mathcal{F}_n}$  corresponding to  $\mathcal{F}_n$  ( $\mathcal{F}_n$  is the functional defined as in the proof of Theorem 6.1) reduces to  $K_n$  if we encode a pair  $(f_1, f_2)$  by  $(C_1, C_2)$ , where  $C_1$  is a size- $t(N)$  circuit computing  $f_1$ , and  $C_2$  is a size- $t(N)$  circuit computing  $f_2 \oplus s_n$ . Also,  $size(K_n) \leq 2^{(\log N)^{O(1)}}$ . Thus, by Theorem 3.1,  $\mathcal{F}_n$  is computable by circuits of size  $2^{(\log N)^{O(1)}}$ , and we can apply Proposition 2.4 to complete the proof. ■

## 7. Interpolation-like theorems in the second order setting

The proof of Proposition 2.1, as well as of Theorem 3.1 in the non-trivial direction involves a highly non-constructive step of deciding whether a rectangle is empty (cf. the sentence “those histories of  $P_s^*$  which actually correspond to at least one instance  $(u, v) \in U_{s,i} \times V_{s,i}$ ” on page 9). This step seems to be intractable if we want to prove syntactic analogues of the results from the previous section within the framework provided by first order theories. In this section we briefly outline how to extend this framework to second order theories,

and present in this more general setting interpolation-like theorems which actually imply these results.

Let  $\mathcal{L}_2$  be the second order extension of  $L_2$  obtained by augmenting it with second order variables  $\gamma_1, \gamma_2, \dots$  (for simplicity we allow only unary variables). Let  $\mathcal{S}(\mathcal{L}_2)$  be the second order language which has one sort for first order variables and two different sorts for second order variables. We will be denoting second order variables of the first sort by  $\alpha_1, \alpha_2, \dots$  (free variables) and  $\phi_1, \phi_2, \dots$  (bound variables); second order variables of the second sort will be denoted by  $\beta_1, \beta_2, \dots, \psi_1, \psi_2, \dots$ . We fix the notation  $\mathcal{L}_2^\alpha$  [ $\mathcal{L}_2^\beta$ ] for the sublanguage of  $\mathcal{S}(\mathcal{L}_2)$  (isomorphic to  $\mathcal{L}_2$ ) which allows second order variables only of the first sort [of the second sort, respectively]. For a formula  $A(\gamma_1, \dots, \gamma_r)$  of  $\mathcal{L}_2$  with all free second order variables displayed, we denote by  $A^\alpha(\alpha_1, \dots, \alpha_r)$  and  $A^\beta(\beta_1, \dots, \beta_r)$  its isomorphic copies in  $\mathcal{L}_2^\alpha$  and  $\mathcal{L}_2^\beta$ , respectively.

We form the hierarchy  $\Sigma_i^{w1,b}$  of second order bounded formulae similarly to the ordinary hierarchy  $\Sigma_i^{1,b}$  (see [6, §9.1]) with the exception that the forming rule “if  $A$  is in  $\Sigma_i^{1,b}$  then  $(\forall x \leq t)A$  is in  $\Sigma_i^{1,b}$ ” is weakened to “if  $A$  is in  $\Sigma_i^{w1,b}$  then  $(\forall x \leq |t|)A$  is in  $\Sigma_i^{w1,b}$ ”, and similarly for the dual case. In plain words, we allow sharply bounded first order quantifiers for free, whereas all other first order quantifiers are counted exactly as second order quantifiers.

We define the split versions  $\mathcal{S}\Sigma_i^{w1,b}$  similarly to  $\mathcal{S}\Sigma_i^b$ . That is,  $\mathcal{S}\Sigma_0^{w1,b} \equiv \mathcal{S}\Pi_0^{w1,b} \equiv (\Sigma^{1,b})^\alpha \cup (\Sigma^{1,b})^\beta$ , and the inductive definition of  $\mathcal{S}\Sigma_{i+1}^{w1,b}, \mathcal{S}\Pi_{i+1}^{w1,b}$  is the same as for  $\Sigma_{i+1}^{w1,b}, \Pi_{i+1}^{w1,b}$  (the case  $(\exists \eta)A$  gets split into two, depending on the sort of the second order bound variable  $\eta$ ).

**Definition 7.1.** For a class  $\Phi$  of bounded formulae in  $\mathcal{L}_2$ , we denote by  $\Phi - SIM$  the following principle:

$$\bigwedge_{i=1}^r (\forall x(\alpha_i(x) \equiv \beta_i(x))) \supset (A^\alpha(\alpha_1, \dots, \alpha_r) \equiv A^\beta(\beta_1, \dots, \beta_r)),$$

where  $A(\gamma_1, \dots, \gamma_r)$  is in  $\Phi$ .

Let  $Cl_2$  be the class of bounded formulae without free second order variables. Note that  $Cl_2 - SIM$  is simply  $A^\alpha \equiv A^\beta$ , where  $A \in Cl_2$ . This principle states that isomorphic internal computations run by the two parties (whatever complex) lead to the same result.

Our base theory,  $\mathcal{S}(V_2)$  in the language  $\mathcal{S}(\mathcal{L}_2)$  is, by definition, axiomatized by  $(V_2)^\alpha + (V_2)^\beta + Cl_2 - SIM$ .

For a class  $\Phi$  of formulae in the language  $\mathcal{L}_2$  we denote by  $\Phi^+$  the closure of  $\Phi$  under the operation of substituting  $Cl_2$ -abstracts for second order variables.

**Lemma 7.2.**  $\mathcal{S}(V_2) \vdash (\Sigma_0^{1,b})^+ - SIM$ .

**Proof.** Let  $A(\gamma_1, \dots, \gamma_r, V_1, \dots, V_s) \in (\Sigma_0^{1,b})^+$ , where  $A(\gamma_1, \dots, \gamma_r, \gamma_{r+1}, \dots, \gamma_{r+s})$  is in  $\Sigma_0^{1,b}$ , and  $V_1, \dots, V_s$  are  $Cl_2$ -abstracts. In order to show  $A(\gamma_1, \dots, \gamma_r, V_1, \dots, V_s) - SIM$ , we apply an obvious induction on the logical complexity of  $A$ ;  $Cl_2 - SIM$  takes care of the base case  $A \equiv \gamma_i(t)$ ;  $r + 1 \leq i \leq r + s$ . ■

**Lemma 7.3.**  $\mathcal{S}(V_2) + \mathcal{S}\Sigma_1^{w1,b} - PIND \vdash (\Delta_1^{1,b}(U_2^1))^+ - SIM$ , where  $\Delta_1^{1,b}(U_2^1)$  is the set of formulae which are  $\Delta_1^{1,b}$  with respect to  $U_2^1$ .

**Proof.** It is an immediate corollary of the main result in [19] that every  $A(\vec{a}, \vec{\gamma})$  in  $\Delta_1^{1,b}(U_2^1)$  is equivalent to the result of evaluating a  $\Sigma_0^{1,b}$ -definable circuit  $\delta(\vec{a}, \vec{\gamma})$  of depth  $|\vec{a}|^{O(1)}$ . Thus, we only have to show in  $\mathcal{S}(V_2) + \mathcal{S}\Sigma_1^{w1,b} - PIND$  that  $\bigwedge_i \forall x (\alpha_i(x) \equiv \beta_i(x)) \supset (\delta^\alpha(\vec{a}, \vec{\alpha}, \vec{V}) \equiv \delta^\beta(\vec{a}, \vec{\beta}, \vec{V}))$  for any circuit  $\delta$  of this kind and any abstracts  $\vec{V}$  in  $Cl_2$ . This is done by  $\mathcal{S}\Pi_1^{w1,b} - PIND$  on  $d$  applied to the formula “every node of  $\delta$  at the  $d$ th level outputs the same value in  $\delta^\alpha(\vec{a}, \vec{\alpha}, \vec{V})$  and in  $\delta^\beta(\vec{a}, \vec{\beta}, \vec{V})$ ”. ■

The following is proved in exactly the same way.

**Lemma 7.4.**  $\mathcal{S}(V_2) + \mathcal{S}\Sigma_1^{w1,b} - IND \vdash (\Delta_1^{1,b}(V_2^1))^+ - SIM$ .

Now we are in position to formulate and prove interpolation-like theorems generalizing the results of the previous section.

**Theorem 7.5.** Let  $A(\vec{\gamma}), B(\vec{\gamma}'), C(a, \vec{\gamma}), D(a, \vec{\gamma}')$  be  $\Sigma^{1,b}$ -formulae, where all occurrences of  $a$  and of all free second order variables are explicitly displayed. Then  $\mathcal{S}(V_2)$  proves the formula

$$\forall \vec{\phi} \forall \vec{\psi} ((A^\alpha(\vec{\phi}) \wedge B^\beta(\vec{\psi})) \supset \exists x (C^\alpha(x, \vec{\phi}) \not\equiv D^\beta(x, \vec{\psi}))) \quad (7)$$

if and only if there exists  $E(\gamma) \in (\Sigma_0^{1,b})^+$  such that

$$V_2 \vdash \forall \vec{\phi} (A(\vec{\phi}) \supset E(\{x\}C(x, \vec{\phi}))) \quad (8)$$

and

$$V_2 \vdash \forall \vec{\psi} (B(\vec{\psi}) \supset \neg E(\{x\}D(x, \vec{\psi}))). \quad (9)$$

**Theorem 7.6.** *Let  $A, B, C, D$  have the same meaning as in Theorem 7.5. Then the formula (7) is provable in  $\mathcal{S}(V_2) + \mathcal{S}\Sigma_1^{\omega_1, b} - PIND$  if and only if there exists  $E(\gamma) \in (\Delta_1^{1, b}(U_2^1))^+$  with the properties (8), (9).*

**Theorem 7.7.** *For the same  $A, B, C, D$ ,  $\mathcal{S}(V_2) + \mathcal{S}\Sigma_2^{\omega_1, b} - PIND$  proves (7) if and only if there exists  $E(\gamma) \in (\Delta_1^{1, b}(V_2^1))^+$  satisfying (8), (9).*

These theorems, combined with the material from Section 2.4, indeed generalize the results of the previous section if we notice that  $E(\gamma)$  with properties (8), (9) encodes a circuit from the class needed in each of the three cases separating functions  $\{\{x\}C(x, \vec{\alpha}) \mid A(\alpha)\}$  from functions  $\{\{x\}D(x, \vec{\beta}) \mid B(\beta)\}$ . The output of this circuit corresponds to  $E_n$  in the proof of Theorem 6.1, and the  $Cl_2$ -abstracts provide non-uniformity.

The proofs of Theorems 7.5, 7.6, 7.7 in the easy direction are based on Lemmas 7.2, 7.3, 7.4, respectively. Namely, assume that we have (8), (9) for some  $E(\gamma)$  from the class  $\Phi$  prescribed in each of the three cases. We lift these proofs to  $(V_2)^\alpha$  and  $(V_2)^\beta$ , and find that  $\mathcal{S}(V_2) \vdash \forall \vec{\phi} \forall \vec{\psi} \left( (A^\alpha(\vec{\phi}) \wedge B^\beta(\vec{\psi})) \supset (E(\{x\}C^\alpha(x, \vec{\phi})) \neq E(\{x\}D^\beta(x, \vec{\psi}))) \right)$ . Now we only have to apply  $\Phi - SIM$  to the formula  $E(\gamma)$ .

The proofs in another direction can be viewed as formalized analogues of Propositions 2.2, 2.1 and Theorem 3.1. In the rest of this section we briefly outline those aspects of this formalization which may appear less obvious.

Firstly, we, similarly to [18], treat  $V_2$  simply as a two-sorted *first order* theory. This allows us to define a language  $\widehat{\mathcal{L}}_2$  and the skolemization  $\widehat{V}_2$  of  $V_2$  in this language similarly to  $\widehat{L}_2(\gamma)$ ,  $\widehat{S}_2(\gamma)$ . Namely, behind function symbols  $f_{A, t}$  already known to us from the previous section, we introduce function symbols  $\theta_A(\vec{b}, \vec{\gamma})$  and  $\pi_B(\vec{b}, \vec{\gamma})$  taking values in the sort for second order variables with the intended meaning  $\theta_A(\vec{b}, \vec{\gamma}) \equiv \mu \phi A(\phi, \vec{b}, \vec{\gamma})$  and  $\pi_B(\vec{b}, \vec{\gamma}) \equiv \{x\}B(x, \vec{b}, \vec{\gamma})$ . Here  $A(\gamma_0, \vec{b}, \vec{\gamma}), B(a, \vec{b}, \vec{\gamma})$  are in  $Open(\widehat{\mathcal{L}}_2)$ , and the operator  $\mu$  corresponds to the ordering of second order objects  $\gamma$  given by  $\gamma \mapsto \sum 2^{-n} \gamma(n)$ . The definition of  $\theta_A$  makes sense in  $\widehat{V}_2$  since there always exists a term  $t_A(\vec{b})$  such that  $\widehat{V}_2 \vdash \forall x \leq t_A(\vec{b}) (\gamma_0(x) \equiv \gamma'_0(x)) \supset (A(\gamma_0, \vec{b}, \vec{\gamma}) \equiv A(\gamma'_0, \vec{b}, \vec{\gamma}))$ . We omit the exact details.

Then we define  $\widehat{\mathcal{S}}(\mathcal{L}_2)$  and  $\widehat{\mathcal{S}}(V_2)$  analogously to  $\widehat{\mathcal{S}}(L_2)$  and  $\widehat{\mathcal{S}}(S_2)$ . We will be denoting terms of  $\widehat{\mathcal{S}}(\mathcal{L}_2)$  taking values in the second order variables of the first sort by  $\mathcal{A}_1, \mathcal{A}_2, \dots$ , and terms taking values in the second order variables of the second sort by  $\mathcal{B}_1, \mathcal{B}_2, \dots$ .

Now, suppose  $\mathcal{S}(V_2)$  proves (7). Then  $\widehat{\mathcal{S}}(V_2)$  also proves this formula. Applying Herbrand's theorem (for the three-sorted case) as in the proof of Theorem 6.1, we find witnesses

$s_1(\vec{\alpha}, \vec{\beta}), \dots, s_r(\vec{\alpha}, \vec{\beta})$  to this fact, and it is easy to see that actually they can be combined into one term  $s(\vec{\alpha}, \vec{\beta})$  such that

$$\widehat{\mathcal{S}}(V_2) \vdash (A^\alpha(\vec{\alpha}) \wedge B^\beta(\vec{\beta})) \supset (C^\alpha(s(\vec{\alpha}, \vec{\beta}), \vec{\alpha}) \neq D^\beta(s(\vec{\alpha}, \vec{\beta}), \vec{\beta})). \quad (10)$$

Next, we make an easy observation that the term  $s(\vec{\alpha}, \vec{\beta})$  can be represented in an equivalent form  $s'(\mathcal{A}(\vec{\alpha}), \mathcal{B}(\vec{\beta}))$ , where all occurrences of second order variables are explicitly displayed, and  $s'(\alpha, \beta)$  is a term of  $\widehat{\mathcal{S}}(L_2)$ .

In order to find  $E(\gamma) \in (\Sigma_0^{1,b})^+$  with the required properties (8), (9), we apply induction on the logical complexity of  $s'$ .

**Base case**  $s' \equiv a$ . We have  $\mathcal{S}(V_2) \vdash (A^\alpha(a, \vec{\alpha}) \wedge B^\beta(a, \vec{\beta})) \supset (C^\alpha(a, \vec{\alpha}) \neq D^\beta(a, \vec{\beta}))$ . Applying the sort-erasing interpretation, we find

$$V_2 \vdash (A(a, \vec{\alpha}) \wedge B(a, \vec{\beta})) \supset (C(a, \vec{\alpha}) \neq D(a, \vec{\beta})).$$

The formula  $E(a, \gamma)$  defined by

$$E(a, \gamma) \equiv \begin{cases} \gamma(a) \equiv \exists \vec{\phi} (A(a, \vec{\phi}) \wedge C(a, \vec{\phi})) & \text{if } \exists \vec{\phi} A(a, \vec{\phi}) \wedge \exists \vec{\psi} B(a, \vec{\psi}) \\ \top & \text{if } \exists \vec{\phi} A(a, \vec{\phi}) \wedge \forall \vec{\psi} \neg B(a, \vec{\psi}) \\ \perp & \text{if } \forall \vec{\phi} \neg A(a, \vec{\phi}) \wedge \exists \vec{\psi} B(a, \vec{\psi}) \\ \text{arbitrary} & \text{if } \forall \vec{\phi} \neg A(a, \vec{\phi}) \wedge \forall \vec{\psi} \neg B(a, \vec{\psi}) \end{cases}$$

has the required properties. Note that the case analysis in the definition of  $E(a, \gamma)$  is exactly the place where we use the power of our base theory not available in the first order setting.

**Inductive step.**  $s'(\alpha, \beta) \equiv s''(f^\alpha(\alpha), \alpha, \beta)$ , where  $f(\gamma)$  is a function symbol of  $\widehat{L}_2(\gamma)$ , and we are guaranteed the existence of  $E$  with the desired properties anytime when (10) is true for the term  $s''(a, \mathcal{A}(\vec{\alpha}), \mathcal{B}(\vec{\beta}))$  and *any* choice of  $A, B, C, D$ .

(10) implies

$$\begin{aligned} \widehat{\mathcal{S}}(V_2) \vdash & (A^\alpha(\vec{\alpha}) \wedge f^\alpha(\mathcal{A}(\vec{\alpha})) = a \wedge B^\beta(\vec{\beta})) \supset \\ & (C^\alpha(s''(a, \mathcal{A}(\vec{\alpha}), \mathcal{B}(\vec{\beta})), \vec{\alpha}) \neq D^\beta(s''(a, \mathcal{A}(\vec{\alpha}), \mathcal{B}(\vec{\beta})), \vec{\beta})), \end{aligned}$$

and we can use our inductive assumption (with  $A(a, \vec{\alpha}) := A(\vec{\alpha}) \wedge f(\mathcal{A}(\vec{\alpha})) = a$ ) to find  $E'(a, \gamma) \in (\Sigma_0^{1,b})^+$  such that

$$V_2 \vdash (A(\vec{\alpha}) \wedge f(\mathcal{A}(\vec{\alpha})) = a) \supset E'(a, \{x\}C(x, \vec{\alpha}))$$

and

$$V_2 \vdash B(\vec{\beta}) \supset \neg E'(a, \{x\}D(x, \vec{\beta})).$$

We simply set  $E(\gamma) \equiv \exists x \leq t E'(x, \gamma)$ , where  $t$  is a term such that  $V_2 \vdash f(\alpha) \leq t$ . This completes the inductive step and the proof of Theorem 7.5.

Coming to Theorem 7.6, we notice that in the theory  $\widehat{\mathcal{S}}(V_2) + \mathcal{S}\Sigma_1^b - PIND$  every  $\mathcal{S}\Sigma_1^{w1,b}$ -formula is equivalent to a  $\Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2))$ -formula of the form  $\exists x \leq t (A^\alpha(x) \wedge B^\beta(x))$ , where  $A(a), B(a) \in \Sigma^{1,b}$ . Indeed, the class of such formulae is closed under applying second order quantifiers:

$$\exists \phi \exists x \leq t (A^\alpha(\phi, x) \wedge B^\beta(x)) \equiv \exists x \leq t (\exists \phi A^\alpha(\phi, x) \wedge B^\beta(x)),$$

and (in the presence of  $\Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2)) - PIND$ ) under applying sharply bounded universal quantifiers<sup>1</sup>:

$$\begin{aligned} \forall y \leq |s| \exists x \leq t (A^\alpha(x, y) \wedge B^\beta(x, y)) &\equiv \\ \exists w (Seq(w) \wedge Len(w) \leq |s| + 1 \wedge Size(w) \leq t \wedge \forall y \leq |s| &A^\alpha((w)_{y+1}, y) \wedge \\ \forall y \leq |s| B^\beta((w)_{y+1}, y)). & \end{aligned}$$

Thus,  $\mathcal{S}(V_2) + \mathcal{S}\Sigma_1^{w1,b} - PIND$  is equivalent to  $\widehat{\mathcal{S}}(V_2) + \Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2)) - PIND$ . But it is straightforward to establish for  $\widehat{\mathcal{S}}(V_2) + \Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2)) - PIND$  the cut elimination theorem and extend to it the syntactic version of Proposition 4.2; in fact, this theory more resembles the first order theory  $S_R^1$  for what might be called “a many-sorted strongly regular theory  $R$ , where no quantifiers other than those on first order variables are allowed” than a second order theory. We skip the details.

The proof of Theorem 7.6 is completed by formalizing the standard proof of Proposition 2.1 in the same fashion as we did above with the proof of Proposition 2.2. We omit exact and somewhat tedious details.

The same ideas work for the weaker version of Theorem 7.7 in which  $\mathcal{S}\Sigma_2^{w1,b} - PIND$  is replaced by  $\mathcal{S}\Sigma_1^{w1,b} - IND$ : extending the syntactic variant of Proposition 4.4 to this case and formalizing the proof of Theorem 3.1 is more or less straightforward.

The analogue of Proposition 4.3 is, however, much less straightforward since we in general can not eliminate second order quantifiers from  $\mathcal{S}\Sigma_2^{w1,b}$ -formulae. We circumvent this as follows.

---

<sup>1</sup>to avoid collision with another usage of  $\beta$ , we denote the  $i$ th member of a sequence  $w$  by  $(w)_i$  rather than by  $\beta(i, w)$



For  $A(\vec{a}, \vec{\alpha}, \vec{\beta}) \in \mathcal{S}\Sigma_2^{w1,b}$  we introduce a *family*  $\mathcal{W}_A^{2,\vec{a},\vec{\alpha},\vec{\beta}}$  of witnessing formulae

$$\text{Witness}_A^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta}) \bullet \Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2)) \cup \Pi_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2))$$

rather than a single formula. All old cases in the standard definition of *Witness* (see [7, Section 4]) are modified in an obvious way, e.g. we say “if  $A$  is  $B \wedge C$  then  $\mathcal{W}_A^{2,\vec{a},\vec{\alpha},\vec{\beta}}$  consists of all formulae of the form  $\text{Witness}_B^{2,\vec{a},\vec{\alpha},\vec{\beta}}((w)_1, \vec{a}, \vec{\alpha}, \vec{\beta}) \wedge \text{Witness}_C^{2,\vec{a},\vec{\alpha},\vec{\beta}}((w)_2, \vec{a}, \vec{\alpha}, \vec{\beta})$ , where  $\text{Witness}_B^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta}) \in \mathcal{W}_B^{2,\vec{a},\vec{\alpha},\vec{\beta}}$ , and  $\text{Witness}_C^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta}) \in \mathcal{W}_C^{2,\vec{a},\vec{\alpha},\vec{\beta}}$ .”

The only case when the branching really occurs is the following new case:

(8) If  $A \notin \Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2)) \cup \Pi_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2))$  and  $A$  is  $\exists \phi B(\vec{a}, \vec{\alpha}, \phi, \vec{\beta})$  then  $\mathcal{W}_A^{2,\vec{a},\vec{\alpha},\vec{\beta}}$  consists of all formulae  $\text{Witness}_A^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta})$  of the form

$$\text{Seq}(w) \wedge \text{Len}(w) = 2 \wedge (w)_1 \leq t(\vec{a}) \wedge \text{Witness}_{B(\vec{a},\vec{\alpha},\alpha_0,\vec{\beta})}^{2,\vec{a},\vec{\alpha},\alpha_0,\vec{\beta}}((w)_2, \vec{a}, \vec{\alpha}, \mathcal{A}(\vec{a}, (w)_1, \vec{\alpha}, \vec{\beta}), \vec{\beta}),$$

where  $t(\vec{a})$  and  $\mathcal{A}(\vec{a}, w, \vec{\alpha}, \vec{\beta})$  run over all terms of the language  $\widehat{\mathcal{S}}(\mathcal{L}_2)$ , and

$$\text{Witness}_{B(\vec{a},\vec{\alpha},\alpha_0,\vec{\beta})}^{2,\vec{a},\vec{\alpha},\alpha_0,\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \alpha_0, \vec{\beta}) \in \mathcal{W}_B^{2,\vec{a},\vec{\alpha},\alpha_0,\vec{\beta}}.$$

The case  $A \equiv \exists \psi B(\psi)$  is treated in the same way.

For every  $\text{Witness}_A^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta}) \in \mathcal{W}_A^{2,\vec{a},\vec{\alpha},\vec{\beta}}$ ,

$$\widehat{\mathcal{S}}(V_2) \vdash \exists w \text{Witness}_A^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta}) \supset A(\vec{a}, \vec{\alpha}, \vec{\beta}).$$

Due to the very limited nature of witnessing second order variables, we can not hope to reverse this implication in any reasonable sense. But we actually do not need this. We simply show the straightforward analogue of [7, Theorem 17] in the following form:

if

$$\widehat{\mathcal{S}}(V_2) + \mathcal{S}\Sigma_2^{w1,b} - \text{PIND} \vdash G(\vec{a}, \vec{\alpha}, \vec{\beta}) \supset H(\vec{a}, \vec{\alpha}, \vec{\beta}),$$

where  $G, H$  are in  $\mathcal{S}\Sigma_2^{w1,b}$  then for every  $\text{Witness}_G^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta}) \bullet \mathcal{W}_G^{2,\vec{a},\vec{\alpha},\vec{\beta}}$  there exist  $\text{Witness}_H^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta}) \in \mathcal{W}_H^{2,\vec{a},\vec{\alpha},\vec{\beta}}$  and a  $Q_2$ -defined function  $f(w, \vec{a}, \vec{\alpha}, \vec{\beta})$  of  $\widehat{\mathcal{S}}(V_2) + \Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2)) - \text{IND}$  such that

$$\begin{aligned} \widehat{\mathcal{S}}(V_2) + \Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2)) - \text{IND} \vdash \text{Witness}_G^{2,\vec{a},\vec{\alpha},\vec{\beta}}(w, \vec{a}, \vec{\alpha}, \vec{\beta}) \supset \\ \text{Witness}_H^{2,\vec{a},\vec{\alpha},\vec{\beta}}(f(w, \vec{a}, \vec{\alpha}, \vec{\beta}), \vec{a}, \vec{\alpha}, \vec{\beta}). \end{aligned}$$

This allows us to conclude that  $\widehat{\mathcal{S}}(V_2) + \mathcal{S}\Sigma_2^{w1,b} - \text{PIND}$  is  $\mathcal{S}\Sigma_2^{w1,b}$ -conservative over  $\widehat{\mathcal{S}}(V_2) + \Sigma_1^b(\widehat{\mathcal{S}}(\mathcal{L}_2)) - \text{IND}$  and complete the proof of Theorem 7.7.

## 8. Conclusion

Naturally, the most interesting question is to which extent the techniques developed in this paper can advance us toward the main goal of understanding the strength of  $V_1^1$ . Let us first point out that the hierarchy of second order theories introduced in the previous section collapses already at the next level. Indeed,

$$\widehat{S}(V_2) + \mathcal{S}\Sigma_2^{w1,b} - IND \vdash \forall \phi \exists \psi \forall x \leq t(\phi(x) \equiv \psi(x)) \wedge \forall \psi \exists \phi \forall x \leq t(\phi(x) \equiv \psi(x)).$$

Thus, at least with respect to bounded formulae,  $\widehat{S}(V_2) + \mathcal{S}\Sigma_2^{w1,b} - IND$  is simply equivalent to  $V_2$ . So, we restrict our discussion to first order theories.

What we actually did in the proof of Theorem 6.4 (this is also a direct corollary of Theorem 7.7) was to show the following *separation* theorem. Whenever

$$S_R^2 \vdash (A(N, \alpha) \wedge B(N, \beta)) \supset \exists x(C(N, x, \alpha) \neq D(N, x, \beta)), \quad (11)$$

where  $R = \mathcal{S}(S_2)$ , the sets  $\{\{x\}C(N, x, \alpha) \mid A(N, \alpha)\}$  and  $\{\{x\}D(N, x, \beta) \mid B(N, \beta)\}$  can be separated by a size- $2^{(\log N)^{O(1)}}$  circuit. An informal reformulation of this is that every two  $NP$ -sets which are provably disjoint in  $S_R^2$  can actually be separated by a set computable in quasipolynomial time. Is it possible to improve this by replacing  $S_R^2$  in (11) with a stronger theory like  $T_R^2$ ,  $S_2(\alpha, \beta)$ ,  $U_2^1$  or  $V_2^1$ ? This seems to be open even under any reasonable complexity assumption. Note for the comparison that even for the case of  $V_2^1$ , the affirmative answer to a similar question in which we are interested in separating *co- $NP$*  sets is a straightforward corollary of Proposition 4.2 and RSUV-isomorphism [22, 23, 18].

There are several examples showing that for  $NP$ -sets the situation may be different. A couple of them originated from a discussion with Steven Rudich are based upon the lower bound proof for voting polynomials [3] and one-way functions, respectively. In these examples, however, in order to prove the formula (11) one apparently needs at least the strength of  $U_1^1$ . Also, their impact on the future research in this direction is still to be understood. Thus, we confine ourselves here with a simpler combinatorial example which gives a new unexpected proof of a known result from [9] and raises several immediate open questions.

**Example 1.** The proof of the separation theorem works for the monotone case as well. That is to say, if

$$S_R^2 \vdash (A(N, \alpha) \wedge B(N, \beta)) \supset \exists x(C(N, x, \alpha) \wedge \neg D(N, x, \beta))$$

then there exists a *monotone* size- $2^{(\log N)^{O(1)}}$  circuit outputting 1 on all  $\{x\}C(N, x, \alpha)$  with  $A(N, \alpha)$ , and outputting 0 on all  $\{x\}D(N, x, \beta)$  with  $B(N, \beta)$ . We will show that this is no longer the case if we replace  $S_R^2$  with  $T_R^3$ .

Indeed, denote by  $WPHP(f)$  the *weak pigeon hole principle* taken in the following form:

$$(a \geq 2 \wedge \forall x \leq a^2 (f(x) < a)) \supset \exists x_1, x_2 < a^2 (f(x_1) = f(x_2) \wedge x_1 \neq x_2).$$

Note that, contrary to the common belief, it is open whether  $T_2^2(f) \vdash WPHP(f)$ . But the proof in [16] lets us conclude at least that  $T_2^3(f) \vdash WPHP(f)$ , and this (naturally) extends to showing that  $T_R^3(f) \vdash WPHP(f)$  for every  $\Sigma_1^b$ -definable  $f$ .

Now, let  $A(N, f_\alpha)$  say “ $f_\alpha$  is an injective mapping from  $[N^2]$  to  $[N^4]$ ”. Let  $B(N, f_\beta)$  say “ $f_\beta$  is a mapping from  $[N^4]$  to  $[N]$ ”. Then, applying  $WPHP(f_\beta \circ f_\alpha)$  (available in  $T_R^3$ ), we see that

$$T_R^3 \vdash (N \geq 2 \wedge A(N, f_\alpha) \wedge B(N, f_\beta)) \supset \exists x_1 < x_2 < N^4 (x_1, x_2 \in \text{im}(f_\alpha) \wedge f_\beta(x_1) = f_\beta(x_2)).$$

But  $\{x_1, x_2\} (x_1 < x_2 < N^4 \wedge x_1, x_2 \in \text{im}(f_\alpha))$  taken over all possible injective  $f_\alpha : [N^2] \rightarrow [N^4]$  is simply the set of all  $N^2$ -cliques.  $\{x_1, x_2\} (x_1 < x_2 < N^4 \wedge f_\beta(x_1) = f_\beta(x_2))$  is the set of all  $N$ -partite complete subgraphs. These two sets can not be separated by a subexponential size *monotone* circuit [2].

This example suggests several open questions. Is it true that  $T_2^2(f) \vdash WPHP(f)$ ? Is it true that  $T_R^2(f) \vdash WPHP(f_\beta \circ f_\alpha)$ ? Is the monotone version of the separation theorem true for  $T_R^2$ ?

In connection with the last question the following observation made by J. Krajíček may turn out useful. Let the weaker principle  $WPHP_1(f, g)$  state that  $f$  and  $g$  do not form two inverse *bijections* between  $[a^2]$  and  $[a]$ , for  $a \geq 2$ . Then this principle is already provable in  $T_2^2(f)$ .

In general, we lack a decent characterization of  $\Sigma_1^b$ -theorems of  $T_2^2$ . In particular, it is still open whether  $S_2(\alpha)$  is  $\Sigma_1^b(\alpha)$ -conservative over  $T_2^2(\alpha)$  or not. Obtaining such a characterization and understanding its meaning in the context of split versions seems to be the most immediate accessible question. The first part of this question is undoubtedly interesting in its own right, irrespectively of the application to particular problems from Boolean complexity.

It is also worth noting that the reasoning in Example 1 can be reversed: since we have the monotone separation theorem for  $S_R^2$ , we also have the independence result  $S_R^2 \not\vdash WPHP(f_\beta \circ f_\alpha)$ . This implies the result from [9] that  $S_2^2(f) \not\vdash WPHP(f)$ .

In the formal sense, Example 1 can not be used for refuting the separation theorem for nonmonotone circuits. Indeed, É. Tardos [24] noticed that the classes of graphs  $G$  with  $\omega(G) \geq s$  and of graphs  $G$  with  $\chi(G) < s$  can be separated by (non-monotone) polynomial size circuits. Still, her proof involves highly nontrivial combinatorial argument known as Lovasz lower bound for Shannon capacity, and it hardly can be expected that this argument would follow from a separation theorem in Bounded Arithmetic.

## 9. Acknowledgement

I am indebted to Sam Bass, Steven Cook, Mauricio Karchmer, Jan Krajíček, Steven Rudich, Avi Wigderson, and Andy Yao for their useful remarks concerning various aspects of this patchwork paper.

## References

- [1] M. Ajtai.  $\Sigma_1^1$ -formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, May 1983.
- [2] N. Alon and R. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.
- [3] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. In *Proceedings of the 23rd ACM STOC*, pages 402–409, 1991. Journal version to appear in *Combinatorica*.
- [4] D. A. Barrington. A note on a theorem of Razborov. Technical report, University of Massachusetts, 1986.
- [5] R. B. Boppana and M. Sipser. The complexity of finite functions. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. A (Algorithms and Complexity)*, chapter 14, pages 757–804. Elsevier Science Publishers B.V. and The MIT Press, 1990.
- [6] S. R. Buss. *Bounded Arithmetic*. Bibliopolis, Napoli, 1986.
- [7] S. R. Buss. Axiomatizations and conservations results for fragments of Bounded Arithmetic. In *Logic and Computation, Contemporary Mathematics 106*, pages 57–84. American Math. Society, 1990.

- [8] S. R. Buss and J. Krajíček. An application of Boolean complexity to separation problems in Bounded Arithmetic. To appear in *Proceedings of the London Mathematical Society*, 1992.
- [9] M. Chiari and J. Krajíček. Witnessing functions in Bounded Arithmetic and search problems. Manuscript in preparation, 1994.
- [10] S. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the 7th Annual ACM Symposium on the Theory of Computing*, pages 83–97, 1975.
- [11] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. *Math. Syst. Theory*, 17:13–27, 1984.
- [12] J. Håstad. *Computational limitations on Small Depth Circuits*. PhD thesis, Massachusetts Institute of Technology, 1986.
- [13] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.
- [14] M. Karchmer. *Communication complexity: A new approach to circuit depth*. PhD thesis, Massachusetts Institute of Technology, 1989.
- [15] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. on Disc. Math.*, 3(2):255–265, May 1990.
- [16] J. B. Paris, A. J. Wilkie, and A. R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *Journal of Symbolic Logic*, 53(4):1235–1244, 1988.
- [17] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. In *Proceedings of the 22th Ann. ACM Symposium on the Theory of Computing*, pages 287–292, 1990.
- [18] A. Razborov. An equivalence between second order bounded domain bounded arithmetic and first order bounded arithmetic. In P. Clote and J. Krajíček, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 247–277. Oxford University Press, 1992.
- [19] A. Razborov. Bounded Arithmetic and lower bounds in Boolean complexity. Submitted to the volume *Feasible Mathematics II*, 1993.

- [20] A. Razborov and S. Rudich. Natural proofs. To appear in the 26th ACM STOC, 1994.
- [21] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 77–82, 1987.
- [22] G. Takeuti.  $S_3^i$  and  $\hat{V}_2^i(BD)$ . *Archive for Math. Logic*, 29:149–169, 1990.
- [23] G. Takeuti.  $RSUV$  isomorphisms. In P. Clote and J. Krajíček, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 364–386. Oxford University Press, 1992.
- [24] É. Tardos. The gap between monotone and nonmonotone circuit complexity is exponential. *Combinatorica*, 8:141–142, 1988.
- [25] G. Wilmer. Bounded existential induction. *The Journal of Symbolic Logic*, 50(1):72–90, March 1985.
- [26] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th IEEE FOCS*, pages 1–10, 1985.
- [27] А.Е. Андреев. Об одном методе получения нижних оценок сложности индивидуальных монотонных функций. *ДАН СССР*, 282(5):1033–1037, 1985. A.E. Andreev, On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Math. Dokl.* 31(3):530–534, 1985.
- [28] А.Е. Андреев. Об одном методе получения эффективных нижних оценок монотонной сложности. *Алгебра и логика*, 26(1):3–21, 1987. A.E. Andreev, On one method of obtaining effective lower bounds of monotone complexity. *Algebra i logika*, 26(1):3–21, 1987. In Russian.
- [29] А. А. Марков. О минимальных контактно-вентильных двухполюсниках для монотонных симметрических функций. In *Проблемы кибернетики*, volume 8, pages 117–121. Наука, 1962. A. A. Markov, On minimal switching-and-rectifier networks for monotone symmetric functions, *Problems of Cybernetics*, vol. 8, 117–121 (1962).
- [30] Э. И. Нечипорук. Об одной булевой функции. *ДАН СССР*, 169(4):765–766, 1966. E. I. Nechiporuk, On a Boolean function, *Soviet Mathematics Doklady* 7:4, pages 999–1000.

- [31] А. А. Разборов. Нижние оценки монотонной сложности некоторых булевых функций. *ДАН СССР*, 281(4):798–801, 1985. A. A. Razborov, Lower bounds for the monotone complexity of some Boolean functions, *Soviet Math. Dokl.*, 31:354–357, 1985.
- [32] А. А. Разборов. Нижние оценки монотонной сложности логического перманента. *Матем. Зам.*, 37(6):887–900, 1985. A. A. Razborov, Lower bounds of monotone complexity of the logical permanent function, *Mathem. Notes of the Academy of Sci. of the USSR*, 37:485–493, 1985.
- [33] А. А. Разборов. Нижние оценки размера схем ограниченной глубины в полном базисе, содержащем функцию логического сложения. *Матем. Зам.*, 41(4):598–607, 1987. A. A. Razborov, Lower bounds on the size of bounded-depth networks over a complete basis with logical addition, *Mathem. Notes of the Academy of Sci. of the USSR*, 41(4):333–338, 1987.
- [34] Б. А. Субботовская. О реализации линейных функций формулами в базисе  $\&, \vee, -$ . *ДАН СССР*, 136(3):553–555, 1961. B.A. Subbotovskaya, Realizations of linear functions by formulas using  $+, *, -$ , *Soviet Mathematics Doklady* 2(1961), 110–112.
- [35] В. М. Храпченко. О сложности реализации линейной функции в классе  $\Pi$ -схем. *Матем. заметки*, 9(1):35–40, 1971. V.M. Khrapchenko, Complexity of the realization of a linear function in the class of  $\pi$ -circuits, *Math. Notes Acad. Sciences USSR* 9(1971), 21–23.
- [36] В. М. Храпченко. Об одном методе получения нижних оценок сложности  $\Pi$ -схем. *Матем. заметки*, 10(1):83–92, 1971. V.M. Khrapchenko, A method of determining lower bounds for the complexity of  $\Pi$ -schemes, *Math. Notes Acad. Sciences USSR* 10(1971), 474–479.

**Lecturer: Russell Impagliazzo**

**Title: Hard-core distributions for somewhat hard functions**

**Material:** R. Impagliazzo: *Hard-core distributions for somewhat hard problems: (Preliminary informal version)*

Computer Science Department  
University of California at San Diego  
San Diego, CA  
USA

E-mail: russell@cs.ucsd.edu



# Hard-core distributions for somewhat hard problems: (Preliminary informal version)

Russell Impagliazzo

August 16, 1994

## Abstract

This work is motivated by the following general problem also looked at under various models by many others (see the bibliography for a small sample.) If you have a problem that is difficult for a certain model on a certain input distribution, in that any algorithm in the model taking less than  $R$  resources has failure probability at least  $\delta$ , is it always the case that combining several independent instances of the problem makes the failure probability proportionally greater? Here, combining can mean asking the algorithm to output answers for each input, or some predicate (e.g., the parity) depending on all the answers. One classical example of such a result is the Yao exclusive-or lemma ([Y2]), which says that if we have a Boolean function  $f$  that is  $\delta$ -hard for circuits of size  $C$ , and  $(1 - 2\delta)^k < \epsilon/2$ , then the function  $f(x_1, \dots, x_k) = f(x_1) \oplus f(x_2) \oplus \dots \oplus f(x_k)$  is  $1/2(1 - \epsilon)$ -hard for circuits of size  $O(\epsilon^2 C)$ . (For a complete proof, see Levin [L]).

I have been interested in this problem mainly from the viewpoint, is full independence between the different  $x_i$ 's necessary, or do the same results pertain if the  $x_i$ 's were chosen in some suitable pseudo-random fashion? I have not been terribly successful at answering this question, but in thinking about these issues I have come up with a Lemma that I think might be of independent interest, and at least gives a new proof of the Yao XOR Lemma (up to some changes in the formula above, effectively requiring  $k$  to double), and some weak results along the lines I was pursuing. [GILVZ] answers a similar question in amplifying the difficulty of inverting functions.

Another interesting slant to this problem is, is it possible to get (in at least some models) some similar results without decreasing the resource bounds? (i.e., a real increase in difficulty, rather than just a time/ probability of correctness trade-off). This direction is pursued in [NRS], but I won't talk about it any more here.

I am already tardy in submitting my seminar contribution. So in the following informal seminar contribution, I will limit myself to stating and

proving the following Lemma, and leave extensions and applications to the future, and to my fellow seminar participants.

**Lemma 1** *Let  $f$  be a Boolean function on  $n$ -bit inputs that is  $\delta$ -hard for circuits of size  $g$  on the uniform distribution, and let  $\epsilon > 0$ . Then there is a set  $S \subseteq \{0, 1\}^n$  so that  $|S| \geq \delta 2^n$  and  $f$  is  $1/2(1 - \epsilon)$  hard on an input uniformly chosen from  $S$  for circuits of size  $d\epsilon^2\delta^2g$ , where  $d$  is an absolute constant.*

A translation into intuitive terms is that for any yes/no problem that is hard to solve almost all the time the instances can be divided up into a set of "easy" instances and a "hard-core" of difficult instances where it is impossible to do significantly better than a random guess. Note that in the above lemma, we are actually off by a constant factor from what we'd expect, since if there were a hard-core set of size  $2\delta$ , we might still be able to predict the function with probability  $(1 - 2\delta) + 1/2(2\delta) = 1 - \delta$ . By recursively applying the lemma, we can make  $|S|$  closer and closer to  $2\delta 2^n$ , but at the expense of reducing the resource bound more and more.

The above actually holds for arbitrary starting distribution, and for any non-uniform model of computation closed under taking majorities.

## 1 Basic Definitions

**Definition 1** *Let  $f$  be a Boolean function on  $n$  bit inputs, and  $D$  a distribution on  $n$  bit strings. Let  $1/2 > \delta > 0$  and let  $n \leq g \leq 2^n/n$ . We say  $f$  is  $\delta$ -hard on  $D$  for size  $g$  if for any circuit  $C$  with at most  $g$  gates, and for  $x$  chosen according to  $D$ ,  $\text{Prob}[C(x) = f(x)] \leq 1 - \delta$ . For a circuit  $C$  and an input  $x$  define  $R_C(x) = 1$  if  $f(x) = C(x)$ ,  $-1$  otherwise. A measure on strings of length  $n$  is a function  $M$  with  $M(x) \in [0, 1]$  (think of it as defining a "fuzzy" set of strings, where instead of definitely being in or out of the set,  $x$  is in the set with probability  $M(x)$ .) The size of a measure  $M$  is written  $\mu(M)$  and is defined by  $\mu(M) = 1/2^n \sum_x M(x)$ . The distribution induced by  $M$  is defined by  $D_M(x) = M(x)/\mu(M)$ . The advantage of  $C$  on  $M$  is defined by  $\text{Adv}_C(M) = 1/2^n \sum_x M(x)R_C(x)$ . It is easy to see that if  $x$  is chosen according to  $D_M$ ,  $\text{Prob}[C(x) = f(x)] \geq 1/2 + \epsilon$  if and only if  $\text{Adv}_C(M) \geq 2\epsilon\mu(M)$ .*

We will use the above definitions in the following way. We will first show that  $f$  is  $1/2 - \epsilon/4$  hard on some  $D_M$  with  $\mu(M) \geq \delta$ . We then use a counting argument to show that a randomly chosen subset  $S$  where  $x \in S$  with probability  $M(x)$  must be almost as hard a distribution as  $D_M$ . This last step seems just a technicality; the hard-core measure will be sufficient for all applications (I believe).

## 2 Intuition

Consider a problem like inverting a one-way function, where if we have a correct solution, then it is easy to verify it. Finding a "hard-core" set of problems for such a distribution is easy. Either there is no circuit of size  $1/2\epsilon\delta g$  that solves the problem on a  $\epsilon$  fraction of instances, or there is. If not, our hard-core distribution is the uniform distribution. If so, this circuit weeds out an  $\epsilon$  fraction of inputs as easy, and we look for a circuit that does well on the remaining inputs. This process continues until either we find a hard-core distribution, or the set of remaining inputs is smaller than  $\delta$ . Note that, since we weed out at least a  $\delta\epsilon$  fraction of inputs each time, this process continues at most  $1/\delta\epsilon$  iterations. (This is overcounting some.) So if we don't find a hard-core distribution, we could piece all of the circuits we found into one circuit that tries them all and outputs the first correct solution. This circuit has size  $g$  and solves the problem  $1 - \delta$  of the time, contradicting the assumed hardness of the problem.

We will follow the same outline, except that in general, we won't be able to tell when a circuit solves a particular instance, so we won't be able to just eliminate those instances where our first circuit solves the problem correctly. Instead, we gradually reduce the importance of those inputs where the circuits we have found so far do well, until we have reached a certain "comfort level" where the margin of success is high enough that we don't have to worry about that input for a while. If the margin of success is large for almost all inputs, the circuit that computes the majority of the circuits we have found computes  $f$  correctly on almost all inputs.

## 3 Proof of Main Lemma

**Lemma 2** *Let  $f$  be  $\delta$ -hard for size  $g$  on the uniform distribution on  $n$ -bit strings, and let  $1 > \epsilon > 0$ . Then there is a measure  $M$  with  $\mu(M) \geq \delta$  so that  $f$  is  $1/2(1 - \epsilon)$ -hard for size  $1/4\epsilon^2\delta^2g$  on  $D_M$ .*

Proof: Assume not, i.e., that on every measure  $M$  with  $\mu(M) \geq \delta$ , we can find a circuit  $C_M$  of size  $g' = 1/4\epsilon^2\delta^2g$  so that  $\text{Prob}[f(x) = C_M(x)] \geq 1/2(1 + \epsilon)$  when  $x$  is chosen according to  $D_M$ . Let  $\epsilon' = \epsilon\delta$ . Then for each such  $M$ ,  $\text{Adv}_{C_M}(M) \geq \epsilon\mu(M) \geq \epsilon'$ .

For a set of circuits  $C_1, \dots, C_i$ , let  $N_i(x) = \sum_{1 \leq j \leq i} R_{C_j}(x)$  (i.e., the margin by which we are predicting  $f$  on  $x$  correctly), and let  $M_i(x) = 1$  if  $N_i(x) \leq 0$ , 0 if  $N_i(x) \geq 1/\epsilon'$  and  $1 - \epsilon'N_i(x)$  otherwise. (In other words, if we've guessed more incorrectly than correctly on  $x$  we definitely want to include  $x$  in our next candidate hard-core distribution, if we have a comfortable margin on  $x$ , we don't, and if we are somewhere in between, include  $x$  with a probability decreasing linearly with our margin.) For the empty set of circuits,  $N_0(x) = 0$  so  $M_0(x) = 1$  (i.e., we start in the uniform distribution.)

Let  $C_1 = C_{M_0}$ . If  $\mu(M_1) \geq \delta$ , let  $C_2 = C_{M_1}$ , and so on. Note that  $\text{maj}(C_1, \dots, C_i)$  is correct on all inputs except those with  $M_i(x) = 1$ , so  $\text{Prob}[\text{maj}(C_1, \dots, C_i) = f(i)] \geq 1 - \mu(M_i)$ . So if the above process halts before  $i = g/2g' = 2\epsilon'^{-2}$  then this defines a circuit of size at most  $g'i + O(i) < 2g'i = g$  gates that computes  $f$  on  $1 - \delta$  of the inputs, a contradiction to the assumed hardness of  $f$ .

On the other hand, let  $x$  be any fixed input. Let  $A(x) = \sum_{0 \leq j < i-1} R_{C_j}(x)M_{i-1}(x)$ . We claim  $A(x) \leq \min\{N_i(x), 1/\epsilon'\} + 1/2\epsilon'i$ . To see this, for each  $k$ , match up the times  $j$  so that  $N_j(x) = k, N_{j+1}(x) = k + 1$  with those where  $N_j(x) = k + 1, N_{j+1}(x) = k$ , with possibly one time left out for each  $0 \leq k < N_i(x)$ , or  $N_i(x) \leq k < 0$ . (In other words, if you ride an elevator starting at the ground floor, you will go up from floor  $k$  to floor  $k + 1$  at most one more time than you go down from floor  $k + 1$  to floor  $k$ , and that one time will occur if and only if you get off at a floor greater than  $k$ . P.S. The analagous principle applies for health conscious people who take the stairs. ) For each such pair of times  $a, b$ ,  $R_{C_{a+1}}(x)M_a(x) + R_{C_{b+1}}(x)M_b(x) = M_a(x) - M_b(x)$ . If  $0 \leq k < 1/\epsilon'$ , this is  $1 - k(\epsilon') - (1 - (k + 1)\epsilon') = \epsilon'$ , and otherwise it is 0. Thus, each pair together contribute at most  $\epsilon'$  to the sum, so all pairs contribute at most  $i/2\epsilon'$ . Each unmatched edge with  $k < 0$  contributes  $-1$ , each unmatched edge with  $0 \leq k < 1/\epsilonpsilon'$  contributes at most 1, and each unmatched edge with  $k \geq 1/\epsilonpsilon'$  contributes 0, so the total contribution of the unmatched edges is at most  $\min\{N_i(x), 1/\epsilonpsilon'\}$ . Thus, we have proved the claim.

On the other hand,  $\sum_x A(x) = 2^n \sum_{0 \leq j < i-1} \text{Adv}_{C_{j+1}}(M_j) \geq 2^n i \epsilon'$ . So combining these, we have  $\sum_x \min\{N_i(x), 1/\epsilonpsilon'\} \geq 2^n i / 2\epsilon' = 2^n 1/\epsilon'$  so  $N_i(x) \geq 1/\epsilonpsilon' > 0$  for all  $x$ . But this gives us a circuit of size  $g$  that always computes  $f$  correctly. This contradiction proves the Lemma.

## 4 Getting a hard core set from a hard-core measure

**Lemma 3** *Let  $f$  be  $1/2(1 - \epsilon/2)$  hard for size  $2n < g < (1/8)(2^n/n)(\epsilon\delta)^2$  on  $D_M$ , where  $M$  is a measure with  $\mu(M) \geq \delta$ . Then there is a set  $S$  with  $|S| \geq \delta 2^n$  and  $f$  is  $1/2(1 - \epsilon)$  hard for size  $g$  on  $U_S$  (the uniform distribution on  $S$ .)*

*Proof:* First, note that the number of circuits of size  $g$  is at most  $(2(2n + g))^{2g} \leq 2^{2ng} \leq 1/4e^{2^n \epsilon^2 \delta^2 / 2}$ . Let  $C$  be any circuit of size  $g$ , and pick  $S$  by placing  $x \in S$  with probability  $M(x)$ . Then  $\text{Adv}_C(M) = \text{Exp}[\text{Adv}_C(U_S)] \leq \epsilon\mu(M)$ , and  $\text{Adv}_C(U_S)$  is the sum of  $2^n$  independent random variables that are in the interval  $[0, 2^{-n}]$ . Hence, the probability that  $\text{Adv}_C(U_S) \geq 2\epsilon\mu(M)$  is at most  $e^{-2^n \epsilon^2 \delta^2 / 2}$ , by Chernoff bounds. Thus, the probability that there is such a  $C$  is at most  $1/4$ . On the other hand, the probability that  $|S| \geq \text{Exp}|S| = \mu(M)$  is about  $1/2$  (I'm fudging a bit here, I should check this more carefully.) Therefore, there is a set  $S$  with  $|S| \geq \mu(M)$  and  $\text{Adv}_C(U_S) \leq 2\epsilon\mu(M)$  for every circuit  $C$

with at most  $g$  gates. Therefore,  $\text{Prob}[C(x) = f(x)] \leq 1/2 + \epsilon$  for  $x$  uniformly selected from  $S$  for any such circuit  $C$ , and so  $f$  is  $1/2 - \epsilon$ -hard for size  $g$  on  $U_S$ .

Lemma 1 then follows from combining Lemma 2 and Lemma 3 with the observation that if  $f$  is  $\delta$ -hard on any distribution for any  $\delta$  for size  $g$ , then  $g < 2^n/n$ , since any function can be computed with  $2^n/n$  gates.

## References

- [ABG] A. Amir, R. Beigel, W. Gasarch, "Some connections between bounded query classes and nonuniform complexity", *5th Structures in Complexity Theory Conference*, 1990.
- [Bsh] N.H. Bshouty, "On the extended direct sum conjecture", *Proceedings of 21<sup>st</sup> STOC*, pp. 177-185 (1989).
- [FKN] T. Feder, E. Kushilevitz, M. Naor, "Amortized Communication Complexity", *32nd FOCS*, pp. 239-248, 1991.
- [J] J. Ja'Ja', "On the Validity of the Direct Sum Conjecture", *SIAM J. Comput.*, 15, 4, pp. 1004-1020, 1986.
- [KKN] M. Karchmer, E. Kushilevitz, N. Nisan, "Fractional Covers and Communication Complexity", *7th Structures in Complexity Theory Conference*, pp. 262-274, 1992.
- [KRW] M. Karchmer, R. Raz, A. Wigderson, "On Proving Super-Logarithmic Depth Lower Bounds via the Direct Sum in Communication Complexity", *Structures in Complexity Theory '91*, pp. 299-304 (1991).
- [L] L. Lovász, "On the ratio of optimal integral and fractional cover", *Discrete Mathematics*, 13, pp. 383-390, 1975.
- [NRS] N. Nisan, S. Rudich M. Saks, Manuscript.
- [Y] A. C.-C. Yao, "Some complexity questions related to distributive computing", *Proceedings of 11<sup>th</sup> STOC*, pp. 209-213 (1979).
- [BM] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM J. on Computing*, Vol. 13, 1984, pp. 850-864. A preliminary version appeared in *23<sup>rd</sup> FOCS* 1982.
- [GILVZ] Goldreich, O., Impagliazzo, R., Levin, L., Venkatesan, R., Zuckerman, D., "Security Preserving Amplification of Harness", *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pp. 318-326, 1990.

- [GL] Goldreich, O., and L.A. Levin, "A Hard-Core Predicate for any One-way Function", 21<sup>st</sup> *STOC*, 1989, pp 25-32.
- [GM] Goldwasser, S. and Micali, S., "Probabilistic Encryption," *JCSS*, Vol. 28, No. 2, April 1984, pp. 270-299. A preliminary version appeared in 14<sup>th</sup> *STOC*, 1982.
- [L] Levin, L.A., "One-way Function and Pseudorandom Generators", *Combinatorica*, Vol. 7, No. 4, 1987, pp. 357-363. A preliminary version appeared in 17<sup>th</sup> *STOC*, 1985.
- [Y2] Yao, A.C., "Theory and Applications of Trapdoor Functions", 23<sup>rd</sup> *FOCS*, 1982, pp. 80-91.



**Lecturer: Amnon Ta-Shma**

**Title: Symmetric Log-space is Closed Under Complement**

**Material:** Slides &  
Noam Nisan and Amnon Ta-Shma: *Symmetric Logspace is Closed Under Complement*

Institute of Computer Science  
Hebrew University  
Jerusalem, Israel  
E-mail: {noam, am}@CS.HUJI.AC.IL



# Symmetric Logspace is Closed Under Complement

Noam Nisan  
Amnon Ta-Shma  
The Hebrew University

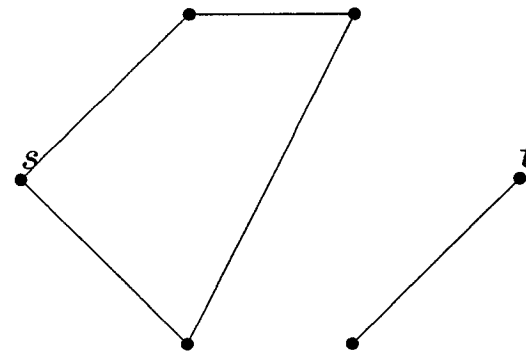
Definition [Lewis,Papadimitriou]:

$SL$  is the class of languages accepted by symmetric non-deterministic logspace Turing machines.

Theorem [Lewis,Papadimitriou]:

$USTCON$  is complete for  $SL$  (under many-one logspace reductions).

$USTCON$  is the undirected  $s, t$  connectivity problem.



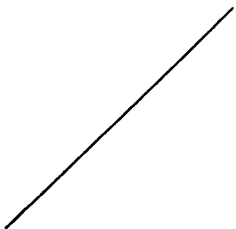
# Complexity map

$L$  \_\_\_\_\_  $SL$  \_\_\_\_\_ \_\_\_\_\_  $NL$

$RL$   
[AKLLR]

[AKLLR] - Aleliunas, Karp, Lipton,  
Lovasz, Rackof

*CO - SL*      *CO - RL*    *CO - NL*



[I,S]

[I,S]      - Immerman / Szelepcsényi

? [BCDRT] ?

[BCDRT] - Borodin, Cook, Dymond,  
Ruzzó, Tompa

## Main result

Theorem:  $SL = CO - SL$

Corollary:  $SL^{SL} = SL$

We need to show that given  $(G, s, t)$  we can build  $(G', s', t')$  s.t:

$G$  is  $s, t$  connected  $\iff$

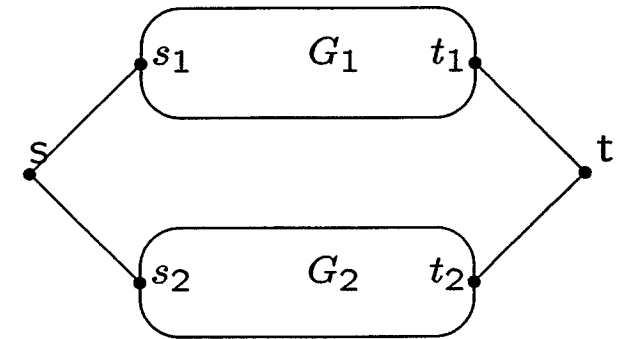
$G'$  is NOT  $s', t'$  connected.

## How to combine results

### Overview of proof

- Combining  $s$ - $t$  non-connectivity problems.
- Counting the number of connected components of  $G$ ,  $\#CC(G)$ .
- $s$  is connected to  $t$  in  $G \iff \#CC(G) = \#CC(G \cup (s, t))$

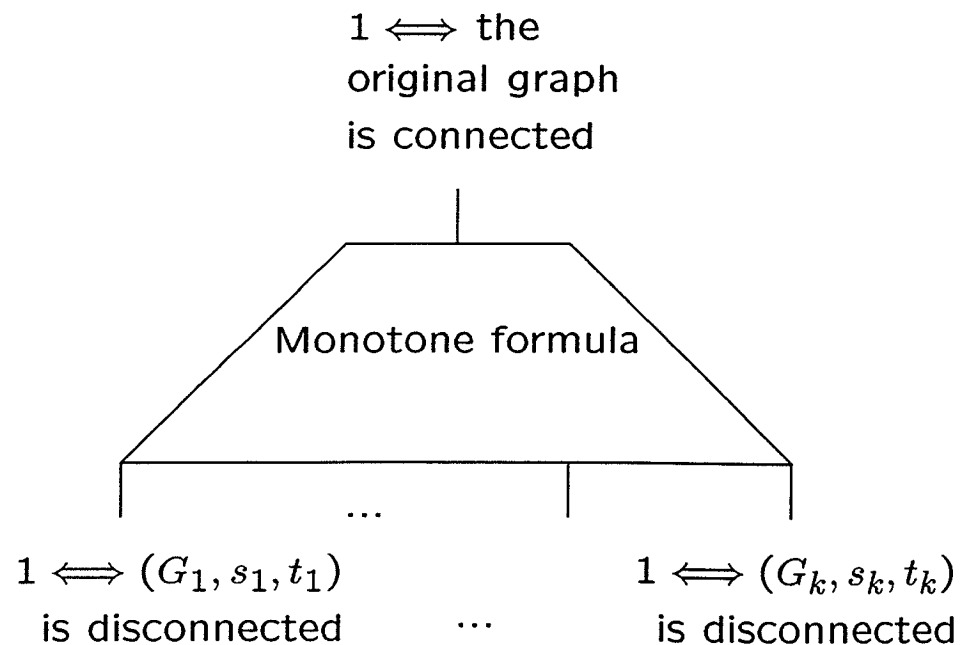
$(G_1, s_1, t_1)$  is disconnected AND  $(G_2, s_2, t_2)$  is disconnected.



$(G_1, s_1, t_1)$  is disconnected OR  $(G_2, s_2, t_2)$  is disconnected.



Thus, if we can solve the connectivity problem with a monotone formula whose inputs are disconnectivity problems, we can express the connectivity problem as one disconnectivity problem.



7

## Computing $\#CC(G)$

We will build:

1. A Vector  $A$  with  $r$  1's, where  $r$  is the number of edges in a spanning forest of  $G$ .
2. A Vector  $B$  with  $k$  1's, where  $k = \#CC(G)$ .

Then, we will show how we can solve the connectivity problem using these vectors.

8

## #edges in a spanning forest

Idea: Consider greedy MSF algorithm.

Index the edges from 1 to  $m$ . There is a unique lexicographically first spanning forest  $T$ .

Notation:  $G_e$  is the subgraph of  $G$  containing all edges with index  $< \text{Index}(e)$ .

Lemma:  $e = (i, j) \in T \iff i$  is disconnected from  $j$  in  $G_e$ .

Let  $A_e = 1 \iff e \in T$ .

The vector  $A$  has exactly  $|T|$  1's.

## Choosing a representative<sup>128</sup> from each $CC$

Idea: transitive closure

For every  $i \in V$ , define:

$B_i = \bigwedge_{j>i} (j \text{ is not connected to } i)$ .

$B_i = 1 \iff i$  is the vertex with the biggest index in its component.

Thus,  $B$  has exactly  $\#CC(G)$  1's.

## Finding $\#CC(G)$

Given  $G$ :

1) Construct  $A$  and  $B$ .  $A$  contains  $r = |T|$  1's,  $B$  contains  $k = \#CC(G)$  1's.

Notice that  $k + r = n$ .

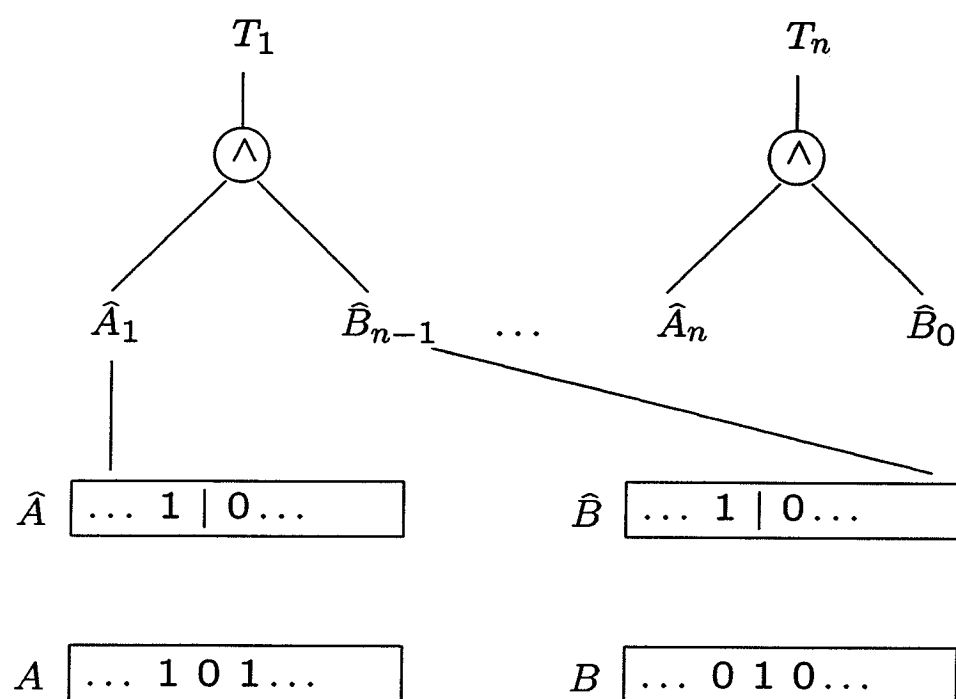
2) Construct a vector  $\hat{A}$  s.t.  $\hat{A}_i = 1 \iff$  there are at least  $i$  1's in  $A$  (using [AKS] sorting networks).

3) Construct  $\hat{B}$

4) Define  $T_i = \hat{B}_i \wedge \hat{A}_{n-i}$

Then:  $T_i = 1 \iff i = \#CC(G)$

## Finding $\#CC(G)$





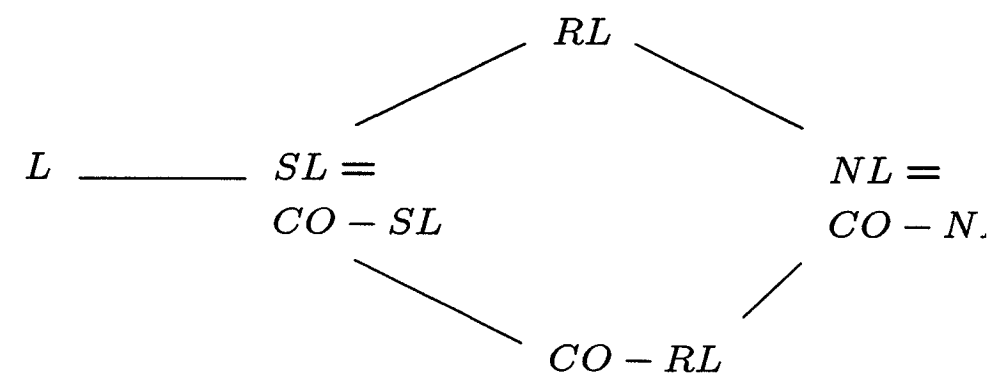
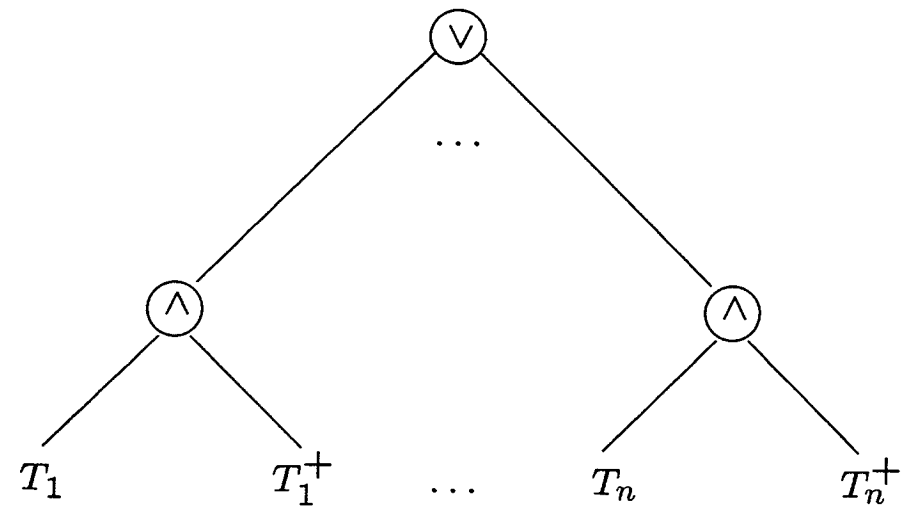
New complexity map

Solving *USTCON*

$s$  is connected to  $t$  in  $G \iff$   
 $\#CC(G) = \#CC(G \cup \{(s,t)\})$ .

Build the vectors:

- 1)  $T_i = 1 \iff \#CC(G) = i$
- 2)  $T_i^+ = 1 \iff \#CC(G \cup \{(s,t)\}) = i$ .



?

?

?

?

?

?

14-a

# Symmetric Logspace is Closed Under Complement \*

Noam Nisan  
noam@cs.huji.ac.il

Amnon Ta-Shma  
am@cs.huji.ac.il

September 28, 1994

## Abstract

We present a Logspace, many-one reduction from the undirected st-connectivity problem to its complement. This shows that  $SL = co - SL$ .

## 1 Introduction

This paper deals with the complexity class symmetric Logspace,  $SL$ , defined by Lewis and Papadimitriou in [LP82]. This class can be defined in several equivalent ways:

1. Languages which can be recognised by symmetric nondeterministic Turing Machines that run within logarithmic space. See [LP82].
2. Languages that can be accepted by a uniform family of polynomial size contact schemes (also sometimes called switching networks.) See [Raz91].
3. Languages which can be reduced in Logspace via a many-one reduction to  $USTCON$ , the undirected st-connectivity problem.

A major reason for the interest in this class is that it captures the complexity of  $USTCON$ . The input to  $USTCON$  is an undirected graph  $G$  and two vertices in it  $s, t$ , and the input should be accepted if  $s$  and  $t$  are connected via a path in  $G$ . The similar problem,  $STCON$ , where the graph  $G$  is allowed to be directed is complete for  $NL$ , non-deterministic Logspace. Several combinatorial problems are known to be in  $SL$  or  $co - SL$ , e.g. 2-colourability is complete in  $co - SL$  [Rei82].

The following facts are known regarding  $SL$  relative to other complexity classes in “the vicinity”:

$$L \subseteq SL \subseteq RL \subseteq NL.$$

Here,  $L$  is the class deterministic Logspace and  $RL$  is the class of problems that can be accepted with one-sided error by a randomized Logspace machine running in polynomial

---

\*This work was supported by BSF grant 92-00043 and by a Wolfson award administered by the Israeli Academy of Sciences. The work was revised while visiting BRICS, Basic Research in Computer Science, Centre of the Danish National Research Foundation.

time. The containment  $SL \subseteq RL$  is the only non-trivial one in the line above and follows directly from the randomized Logspace algorithm for  $USTCON$  of [AKL<sup>+</sup>79]. It is also known that  $SL \subseteq SC$  [Nis92],  $SL \subseteq \oplus L$  [KW93] and  $SL \subseteq DSPACE(\log^{1.5} n)$  [NSW92].

After the surprising proofs that  $NL$  is closed under complement were found [Imm88, Sze88], Borodin et al [BCD<sup>+</sup>89] asked whether the same is true for  $SL$ . They could prove only the weaker statement, namely that  $SL \subseteq co-RL$ , and left “ $SL = co-SL$ ?” as an open problem. In this paper we solve the problem in the affirmative by exhibiting a Logspace, many-one reduction from  $USTCON$  to its complement. Quite surprisingly the proof of our theorem does not use inductive counting, as do the proofs of  $NL = co-NL$ , and is in fact even simpler than them, however it uses the [AKS83] sorting networks.

**Theorem 1**  $SL = co-SL$ .

It should be noted that the monotone analogues (see [GS91]) of  $SL$  and  $co-SL$  are known to be different [KW88].

As a direct corollary of our theorem, we get that  $L^{SL} = SL^{SL} = SL$  where  $L^{SL}$  is the class of languages accepted by *Logspace* oracle Turing machines with oracle from  $SL$ , and  $SL^{SL}$  is defined similarly, being careful with the way we allow queries (see [RST82]).

**Corollary 1.1**  $L^{SL} = SL^{SL} = SL$

This also shows that the “symmetric Logspace hierarchy” defined in [Rei82] collapses to  $SL$ .

## 2 Proof of Theorem

### 2.1 Overview of proof.

We show that we can upper and lower bound the number of connected components of a graph, using connectivity problems. We upper bound this number using a “transitive-closure” method, which can be easily done since we are allowed to freely use connectivity problems. However, trying to lower-bound the number of connected components this way requires negation. The heart of the proof lies in lower-bounding the number of connected components, and we achieve this in a surprisingly easy way, by computing a spanning forest.

In subsection 2.2 we show how to combine many connectivity problems to one single connectivity problem. In subsection 2.3 we show how to find a spanning forest using connectivity problems. In subsection 2.4 we show how to use this spanning forest to find the number of connected components of a graph, and how we solve the *st* non-connectivity problem with it.

### 2.2 Projections to $USTCON$ .

In this paper we will use only the simplest kind of reductions, i.e. *LogSpace* uniform projection reductions [SV85]. Moreover, we will be interested only in reductions to  $USTCON$ . In this subsection we define this kind of reduction and we show some of its basic properties.

**NOTATION 2.1** Given  $f : \{0, 1\}^* \mapsto \{0, 1\}^*$  denote by  $f_n : \{0, 1\}^n \mapsto \{0, 1\}^*$  the restriction of  $f$  to inputs of length  $n$ . Denote by  $f_{n,k}$  the  $k$ 'th bit function of  $f_n$ , i.e. if  $f_n : \{0, 1\}^n \mapsto \{0, 1\}^{k(n)}$  then  $f_n = (f_{n,1}, \dots, f_{n,k(n)})$ .

**NOTATION 2.2** We represent an  $n$ -node undirected graph  $G$  using  $\binom{n}{2}$  variables  $\vec{x} = \{x_{i,j}\}_{1 \leq i < j \leq n}$  s.t.  $x_{i,j}$  is 1 iff  $(i, j) \in E(G)$ . If  $f(\vec{x})$  operates on graphs, we will write  $f(G)$  meaning that the input to  $f$  is a binary vector of length  $\binom{n}{2}$  representing  $G$ .

**DEFINITION 2.1** We say that  $f : \{0, 1\}^* \mapsto \{0, 1\}^*$  reduces to  $USTCON(m)$ ,  $m = m(n)$ , if there is a uniform family of  $Space(\log(n))$  functions  $\{\sigma_{n,k}\}$  s.t. for all  $n$  and  $k$ :

- $\sigma_{n,k}$  is a projection, i.e.:  $\sigma_{n,k}$  is a mapping from  $\{i, j\}_{1 \leq i < j \leq m}$  to  $\{0, 1, x_i, \neg x_i\}_{1 \leq i \leq n}$
- Given  $\vec{x}$  define  $G_{\vec{x}}$  to be the graph  $G_{\vec{x}} = (\{1, \dots, m\}, E)$  where  $E = \{(i, j) \mid \sigma_{n,k}(i, j) = 1 \text{ or } \sigma_{n,k}(i, j) = x_i \text{ and } x_i = 1 \text{ or } \sigma_{n,k}(i, j) = \neg x_i \text{ and } x_i = 0\}$ . It should hold that  $f_{n,k}(\vec{x}) = 1 \iff$  there is a path from 1 to  $m$  in  $G_{\vec{x}}$ .

If  $\sigma$  is restricted to the set  $\{0, 1, x_i\}_{1 \leq i \leq n}$  we say that  $f$  monotonically reduces to  $USTCON(m)$ .

**Lemma 2.1** If  $f$  has uniform monotone formulae of size  $s(n)$  then  $f$  is monotonically reducible to  $USTCON(O(s(n)))$ .

**Proof:** Given a formula  $\phi$  recursively build  $(G, s, t)$  as follows:

- If  $\phi = x_i$  then build a graph with two vertices  $s$  and  $t$ , and one edge between them labelled with  $x_i$ .
- If  $\phi = \phi_1 \wedge \phi_2$ , and  $(G_i, s_i, t_i)$  the graphs for  $\phi_i$ ,  $i = 1, 2$ , then identify  $s_2$  with  $t_1$  and define  $s = s_1, t = t_2$ .
- If  $\phi = \phi_1 \vee \phi_2$ , and  $(G_i, s_i, t_i)$  the graphs for  $\phi_i$ ,  $i = 1, 2$ , then identify  $s_1$  with  $t_1$  and  $s_2$  with  $t_2$  and define  $s = s_1 = t_1$  and  $t = s_2 = t_2$ .

□

Using the AKS sorting networks [AKS83], which belong to  $NC^1$ , we get:

**Corollary 2.2**  $Sort : \{0, 1\}^* \mapsto \{0, 1\}^*$  (which given a binary vector sorts it) is monotonically reducible to  $USTCON(poly)$ .

**Lemma 2.3** If  $f$  monotonically reduces to  $USTCON(m_1)$  and  $g$  reduces to  $USTCON(m_2)$  then  $f \circ g$  reduces to  $USTCON(m_1^2 \cdot m_2)$ , where  $\circ$  is the standard function composition operator.

**Proof:**  $f$  monotonically reduces to a graph with  $m_1$  vertices, where each edge is labelled with one of  $\{0, 1, x_i\}$ . In the composition function  $f \circ g$  each  $x_i$  is replaced by  $x_i = g_i(\vec{y})$  which can be reduced to a connectivity problem of size  $m_2$ . Replace each edge labelled  $x_i$  with its corresponding connectivity problem. □

### 2.3 Finding a spanning forest.

In this section we show how to build a spanning forest using *USTCON*. This construction was also noticed by Reif and independently by Cook [Rei82].

Given a graph  $G$  index the edges from 1 to  $m$ . We can view the indices as weights to the edges, and as no two edges have the same weight, we know that there is a unique minimal spanning forest  $F$ . In our case, where the edges are indexed, this minimal forest is the lexicographically first spanning forest.

It is well known that the greedy algorithm finds a minimal spanning forest. Let us recall how the greedy algorithm works in our case. The algorithm builds a spanning forest  $F$  which is at the beginning empty  $F = \emptyset$ . Then the algorithm checks the edges one by one according to their order, for each edge  $e$  if  $e$  does not close a cycle in  $F$  then  $e$  is added to the forest, i.e.  $F = F \cup \{e\}$ .

At first glance the algorithm looks sequential, however, claim 2.3 shows that the greedy algorithm is actually highly parallel. Moreover, all we need to check that an edge does not participate in the forest, is one  $st$  connectivity problem over a simple to get graph.

**DEFINITION 2.2** For an undirected graph  $G$  denote by  $LEF(G)$  the lexicographically first spanning forest of  $G$ . Let

$SF(G) \mapsto \{0, 1\}^{\binom{n}{2}}$  be:

$$SF_{i,j}(G) = \begin{cases} 0 & (i, j) \in LEF(G) \\ 1 & \text{otherwise} \end{cases}$$

**Lemma 2.4**  $SF$  reduces to  $USTCON$  (poly)

**Proof:** Let  $F$  be the lexicographically first spanning forest of  $G$ . For  $e \in E$  define  $G_e$  to be the subgraph of  $G$  containing only the edges  $\{e' \in E \mid \text{index}(e') < \text{index}(e)\}$ .

**Claim:**  $e = (i, j) \in F \iff e \in E \wedge i$  is not connected to  $j$  in  $G_e$ .

**Proof:** Let  $e = (i, j) \in E$ . Denote by  $F_e$  the forest which the greedy algorithm built at the time it was checking  $e$ . So  $e \in F \iff e$  does not close a cycle in  $F_e$ .

( $\implies$ )  $e \in F$  and therefore  $e$  does not close a cycle in  $F_e$ , but then  $e$  does not close a cycle in the transitive closure of  $F_e$ , and in particular  $e$  does not close a cycle in  $G_e$ .

( $\impliedby$ )  $e$  does not close a cycle in  $G_e$  therefore  $e$  does not close a cycle in  $F_e$  and  $e \in F$ .  $\square$

Therefore  $SF_{i,j}(G) = \neg x_{i,j} \vee i$  is connected to  $j$  in  $G_{(i,j)}$ .

Since  $\neg x_{i,j}$  can be viewed as the connectivity problem over the graph with two vertices and one edge labelled  $\neg x_{i,j}$  it follows from lemmas 2.1, 2.3 that  $SF$  reduces to  $USTCON$ . Notice, however, that the reduction is not monotone.  $\square$

## 2.4 Putting it together.

First, we want to build a function that takes one representative from each connected component. We define  $LI_i(G)$  to be 0 iff the vertex  $i$  has the largest index in its connected component.

**DEFINITION 2.3**  $LI(G) \mapsto \{0, 1\}^n$

$$LI_i(G) = \begin{cases} 0 & i \text{ has the largest index in its connected component} \\ 1 & \text{otherwise} \end{cases}$$

**Lemma 2.5**  $LI$  reduces to  $USTCON$ (poly)

**Proof:**

$$LI_i(G) = \bigvee_{j=i+1}^n (i \text{ is connected to } j \text{ in } G).$$

So  $LI$  is a simple monotone formula over connectivity problems, and by lemmas 2.1, 2.3  $LI$  reduces to  $USTCON$ . This is, actually, a monotone reduction. □

Using the spanning forest and the  $LI$  function we can exactly compute the number of connected components of  $G$ , i.e.: given  $G$  we can compute a function  $NCC_i$ ; which is 1 iff there are exactly  $i$  connected components in  $G$ .

**DEFINITION 2.4**  $NCC(G) \mapsto \{0, 1\}^n$

$$NCC_i(G) = \begin{cases} 1 & \text{there are exactly } i \text{ connected components in } G \\ 0 & \text{otherwise} \end{cases}$$

**Lemma 2.6**  $NCC$  reduces to  $USTCON$ (poly)

**Proof:**

Let  $F$  be a spanning forest of  $G$ . It is easy to see that if  $G$  has  $k$  connected components then  $|F| = n - k$ .

Define:

$$\begin{aligned} f(G) &= \text{Sort} \circ LI(G) \\ g(G) &= \text{Sort} \circ SF(G). \end{aligned}$$

Then:

$$\begin{aligned} f_i(G) = 1 &\implies k < i \\ g_i(G) = 1 &\implies n - k < i \implies k > n - i. \end{aligned}$$

and thus:  $NCC_i(G) = f_{i+1}(G) \wedge g_{n-i+1}(G)$

Therefore applying lemmas 2.1, 2.2, 2.3, 2.4, 2.5 proves the lemma. □

Finally we can reduce the non-connectivity problem to the connectivity problem, thus proving that  $SL = co - SL$ .

**Lemma 2.7**  $\overline{USTCON}$  reduces to  $USTCON(poly)$

**Proof:**

Given  $(G, s, t)$  define  $G^+$  to be the graph  $G \cup \{(s, t)\}$ .

Denote by  $\#CC(H)$  the number of connected components in the undirected graph  $H$ .

$$s \text{ is not connected to } t \text{ in } G \quad \iff$$

$$\#CC(G^+) = \#CC(G) - 1 \quad \iff$$

$$\bigvee_{i=2, \dots, n} NCC_i(G) \wedge NCC_{i-1}(G^+).$$

Therefore applying lemmas 2.1, 2.3, 2.6 proves the lemma.  $\square$

### 3 Extensions

Denote by  $L^{SL}$  the class of languages accepted by *Logspace* oracle Turing machines with oracle from  $SL$ . An oracle Turing machine has a work tape and a write-only query tape (with unlimited length) which is initialised after every query. We get:

**Corollary 3.1**  $L^{SL} = SL$ .

**Proof:**

Let  $Lang$  be a language in  $L^{SL}$  solved by an oracle Turing machine  $M$  running in  $L^{SL}$ , and fix an input  $\vec{x}$  to  $M$ .

Look at the configuration graph of  $M$ . In this graph we have query vertices with outgoing edges labelled “connected” and “not connected”. We would like to replace the edges labelled “connected” with their corresponding connectivity problems, and the edges labelled “not connected” with the connectivity problems obtained using our theorem that  $SL = co - SL$ .

However, there is a technical problem here, as the queries are determined by the edges and not by the query vertices. We can fix this difficulty by splitting each query vertex to its “yes” and “no” answers, and splitting each edge entering a query vertex to “connected” and “not connected” edges. Now we can easily replace each edge with a connectivity problem, obtaining an undirected graph which is  $st$  connected iff  $\vec{x} \in Lang$ , and therefore  $Lang \in SL$ .  $\square$

As can easily be seen the above argument applies to any undirected graph with  $USTCON$  query vertices, thus, if we carefully define  $SL^{SL}$  (see [RST82]) we get that:

**Corollary 3.2**  $SL^{SL} = SL$ .

In particular, the “symmetric Logspace hierarchy” defined in [Rei82] collapses to  $SL$ .



## 4 Acknowledgements

We would like to thank Amos Beimel, Allan Borodin, Robert Szelepcsényi, Assaf Schuster and Avi Wigderson for helpful discussions.

## References

- [AKL<sup>+</sup>79] R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal sequences and the complexity of maze problems. In *Proceedings of the 20th Annual IEEE Symposium on the Foundations of Computer Science*, 1979.
- [AKS83] M. Ajtai, J. Komlos, and E. Szemerédi. An  $O(n \log n)$  sorting network. In *Proc. 15th ACM Symposium on Theory of Computing (STOC)*, pages 1–9, 1983.
- [BCD<sup>+</sup>89] A. Borodin, S.A. Cook, P.W. Dymond, W.L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SIAM Journal on Computing*, 18(3):559–578, 1989.
- [GS91] Grigni and Sipser. Monotone separation of logspace from  $nc^1$ . In *Annual Conference on Structure in Complexity Theory*, 1991.
- [Imm88] Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17, 1988.
- [KW88] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proc. 20th ACM Symposium on Theory of Computing (STOC)*, pages 539–550, 1988.
- [KW93] Karchmer and Wigderson. On span programs. In *Annual Conference on Structure in Complexity Theory*, 1993.
- [LP82] Lewis and Papadimitriou. Symmetric space-bounded computation. *Theoretical Computer Science*, 19, 1982.
- [Nis92] N. Nisan.  $RL \subseteq SC$ . In *Proc. 24th ACM Symposium on Theory of Computing (STOC)*, pages 619–623, 1992.
- [NSW92] N. Nisan, E. Szemerédi, and A. Wigderson. Undirected connectivity in  $O(\log^{1.5} n)$  space. In *Proc. 33th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 24–29, 1992.
- [Raz91] A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *Proceedings of the 8th FCT, Lecture Notes in Computer Science*, 529, pages 47–60, New York/Berlin, 1991. Springer-Verlag.
- [Rei82] J. H. Reif. Symmetric complementation. In *Proc. 14th ACM Symposium on Theory of Computing (STOC)*, pages 201–214, 1982.

- [RST82] W. L. Ruzzo, J. Simon, and M. Tompa. Space-bounded hierarchies and probabilistic computations. In *Proc. 14th ACM Symposium on Theory of Computing (STOC)*, pages 215–223, 1982.
- [SV85] Skyum and Valiant. A complexity theory based on boolean algebra. *Journal of the ACM*, 1985.
- [Sze88] Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26, 1988.



**Lecturer: Peter Bro Miltersen**

**Title: On Cell Probe Complexity**

Material: Slides &  
Peter Bro Miltersen: *On the cell probe complexity of dynamic problems*

BRICS  
Computer Science Department  
University of Aarhus  
DK-8000 Aarhus C, Denmark  
E-mail: bromille@daimi.aau.dk

## Union-Split-Find

1      4 <sup>min</sup> 5 7      9      ~~10~~ 14  
           ↙      ↘

Maintain  $S \subseteq \{1, 2, \dots, n\}$  under

insert( $i$ )       $S := S \cup \{i\}$

delete( $i$ )       $S := S \setminus \{i\}$

pred( $i$ )      return  $\max(S \cap \{1, \dots, i\})$

Best known solution:

Worst case time per operation

$O(\log \log n)$

[Van Emde Boas, Kaas, Zijlstra '77]

The Cell probe complexity  
of Dynamic problems

Peter Bro Miltersen  
BRICS

## Maintaining partial sums

0 0 1 1 0 <sup>0</sup> ~~1~~ <sup>0</sup> ~~1~~ 0 1 0 0 0  
└──────────┘  
1

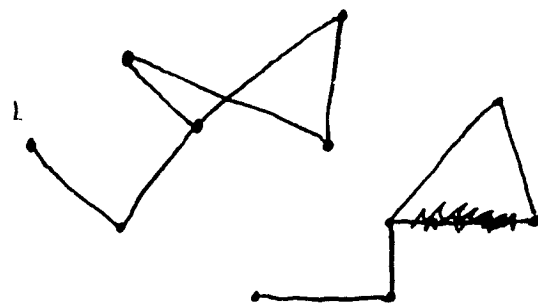
Maintain  $X \in \{0,1\}^n$  under  
change  $(i, a)$   $X_i := a$   
prefix  $(i)$  return  $\sum_{j=1}^i X_j \pmod 2$

Trivial solution:

Worst case time per operation  
 $O(\log n)$

Slightly more clever:  
 $O(\log n / \log \log n)$


## Maintaining connected components



Maintain  $G = (V, E)$  under  
insert  $(e)$   $E := E \cup \{e\}$   
delete  $(e)$   $E := E \setminus \{e\}$   
con  $(u, v)$  Are  $u$  and  $v$  connected?

Best known solution: Worst case  
time per operation  $O(\sqrt{n})$   
[Eppstein, Galil, Italiano, Spencer '93]

# Spreadsheet

	1	2	3
1	$C_{12} + C_{13}$ 0	$\neg C_{21}$ 0	0
2	$\neg C_{13}$ 0	1 1	.
3	$C_{22} + C_{11}$ 1	.	

Best known solution:  
Trivial  $O(n)$   
Patented !!

Union-Split-Find	$O(\log \log n)$
Partial Sums	$O(\log n / \log \log n)$
Connected Components	$O(\sqrt{n})$
Spreadsheet	$O(n)$

Wanted: Lower bounds  
Explanations

Common framework:

Dynamic language membership problems

Given  $L \subseteq \{0,1\}^*$

Maintain  $x \in \{0,1\}^n$  under

change  $(i,a) \quad x_i := a$

member  $\quad$  Is  $x \in L$  ?

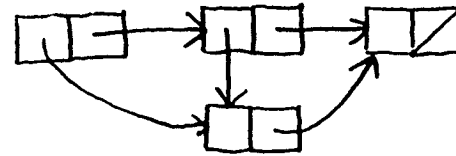
Maintaining connected components  
is captured by  $L = \text{USTCON}$

Partial sums and spreadsheet  
similarly captured

Union-Split-Find not captured  
exactly but we can replace it by

"Colour-Union-Split-Find".

## Machine models



Pointer Machine  
% No arrays

676	677	678
1236972	678	2

LD (677) (time:1)

ADD 676 (time:1)

Unit cost RAM  
% Too strong

676	677	678
1236972	678	2

LD (677) (time:3)

ADD 676 (time:7)

Log cost RAM  
% Too weak

676	677	678
1241	678	2

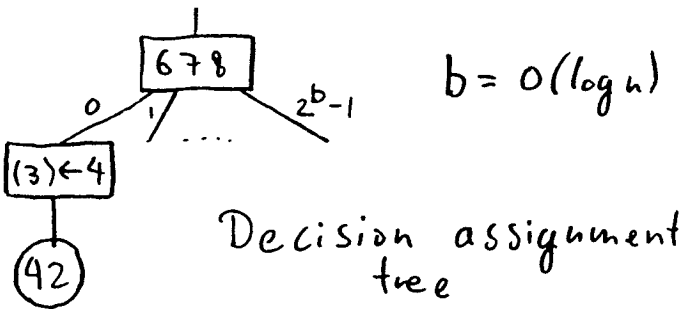
LD (677) (time:1)

ADD 676 (time:1)

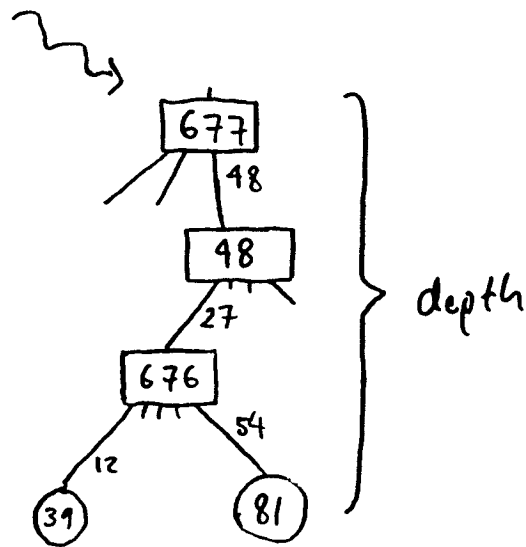
Random access  
Computer <sup>145</sup>  
RAC  $[ \log n ]$   
(Angluin & Valiant)



# Cell Probe Model



LD (677)  
ADD 676  
RTN



For all dynamic language membership problems, the complexity is at most  $O(\frac{n}{\log n})$ .

Proof: The trivial data structure, storing the string itself.

For most problems, the complexity is at least  $\Omega(\frac{n}{\log n})$

Proof: Counting

## Lower bound techniques ( $b = O(\log n)$ )

- Time stamp method  
[Fredman & Saks '89]
- Compression + Reduction from  
communication complexity  
[Ajtai '88, Willard '83, Miltersen '94  
Xiao '92]
- Reductions

## Time stamp method

Partial sums problem:

$ch(i_1, a_1) \ ch(i_2, a_2) \ \dots \ ch(i_m, a_m) \ pr(j)$

The  $i$ 's are fixed, the  $a$ 's and  $j$  random

ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	pr
----	----	----	----	----	----	----	----	----	----	----

Most queries have to consult cells  
belonging to most epochs

Maintaining partial sums requires  
time  $\Omega(\log n / \log \log n)$  per operation.

Efficient solution  
to dynamic problem



Efficient solution  
to static problem  
using small space



Fast protocol for  
communication game



Techniques from  
communication  
complexity

## Compression

Dynamic algorithm maintaining  $S \subseteq [n]$   
under insertions, deletions, queries



Static data structure storing  $S$  using  
small space (same queries answerable)

- Initialize dynamic algorithm

42	23	58	17	...	11
----	----	----	----	-----	----

- Insert elements in  $S$

42	<sup>2</sup> 11	58	<sup>4</sup> 19	...	<sup>717</sup> 70
----	-----------------	----	-----------------	-----	-------------------

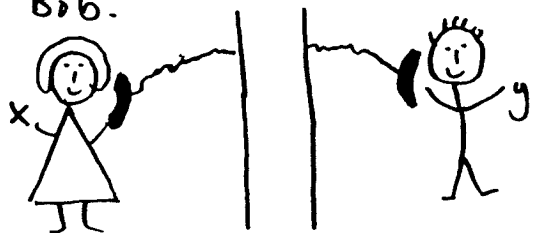
- Construct perfect hash table  
containing  $\{(2, 11), (4, 19), \dots, (717, 70)\}$

## Reduction from Communication Complexity

Solution to static problem with query set  $A$ , data set  $B$  (store  $y \in B$  using few cells so that any query  $x \in A$  can be answered efficiently)



Protocol for communication game: Alice gets  $x \in A$ , Bob gets  $y \in B$ . Alice must determine  $Q_x(y)$  through communication with Bob.



- Bob constructs data structure corresponding to  $y$
- repeat

Alice sends Bob the name of a cell

Bob sends Alice the content of the cell

## Polynomial evaluation with preprocessing of coefficients.

Store  $f(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$  so that  $f(a)$  can be found efficiently for any  $a$ .

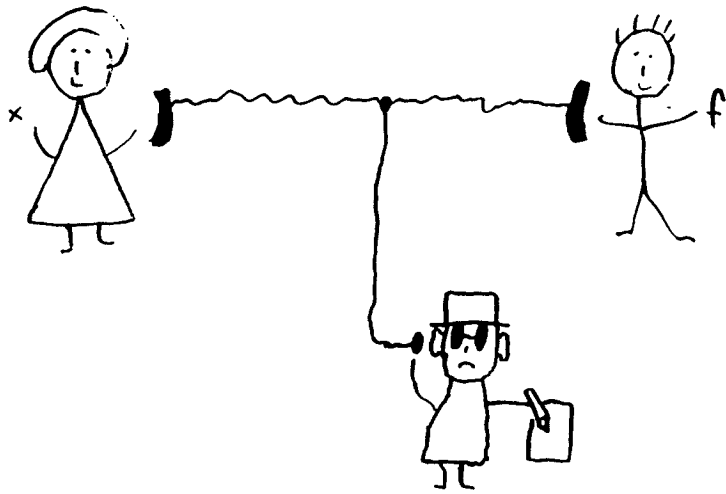
$$f \rightsquigarrow \begin{array}{|c|c|c|c|c|} \hline a_0 & a_1 & a_2 & \dots & a_n \\ \hline \end{array}$$

Evaluate using Horner's rule ( $2n$  ar.op.)

Pan '66: For  $\mathbb{R}$ , there is a scheme using  $\lfloor \frac{3n}{2} \rfloor + 2$  arithmetic operations.

Belega '61: For  $\mathbb{R}$ , any scheme uses at least  $\lfloor \frac{3n}{2} \rfloor + 1$  arithmetic operations

What about finite field  $|K|=k$  with probes instead of ar.op.?



Alice gets  $x \in K$

Bob gets  $f \in K[x]$

Alice sends Bob  $\alpha_1$

Bob sends Alice  $\beta_1$

$\vdots$

Bob sends Alice  $\beta_t$

Alice knows  $f(x)$ , what do we know?

	$x$	$2x$	$3x+3$	$\dots$	$x^2$	$\dots$	$f(x)$
0	<del>0</del>	<del>0</del>	<del>3</del>	<del>0</del>	<del>0</del>	<del>0</del>	<del>0</del>
1	<del>1</del>	<del>2</del>	<del>9</del>	<del>4</del>	<del>1</del>	<del>1</del>	<del>1</del>
2	<del>2</del>	<del>4</del>	<del>4</del>	<del>4</del>	<del>4</del>	<del>4</del>	<del>3</del>
3	<del>13</del>	<del>6</del>	<del>10</del>	<del>6</del>	<del>9</del>	<del>9</del>	<del>9</del>
4	<del>4</del>	<del>8</del>	<del>5</del>	<del>5</del>	<del>5</del>	<del>5</del>	<del>5</del>
$\vdots$	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>
$a$	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>
$\vdots$	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>	<del>1</del>

In the final matrix, every row is constant:

$$\text{width} > n \Rightarrow \text{length} \leq 1$$

For the "greedy" transcript:

$$\text{length} \geq |B|/k^t$$

$$\text{width} \geq |A|/s^t$$

Combining:

$$t \geq \min(n+1, \frac{\log k - \log n}{\log s})$$

Corollary:

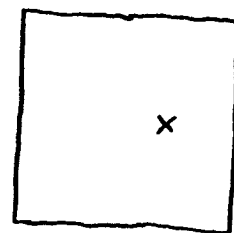
$\Omega(\log n / \log \log n)$  lower bound on the dynamic membership problem for a certain language  $L$ .

Another application:

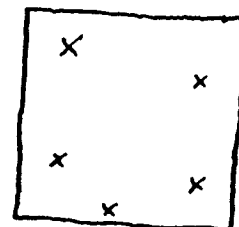
Union-Split-Find requires time  $\Omega(\log \log n / \log \log \log n)$  per operation.

Open problem: 2-dim. orthogonal range query problems.

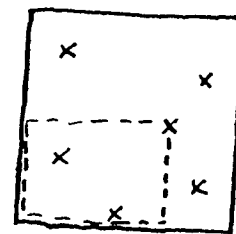
Alice gets  $x \in \{1, \dots, n\}^2$



Bob gets  $S \subseteq \{1, \dots, n\}^2$



How big is  $S_{nr}(x)$ ?



Alice's messages:  $\log |S|$  bits

Bob's messages:  $\log n$  bits

## Dynamic prefix problems

$M$  fixed finite monoid

Maintain  $x_1, x_2 \dots x_n \in M^n$  under

change  $(i, a)$        $x_i := a$   
 prefix  $(i)$       return  $x_1 \circ x_2 \circ \dots \circ x_i$

$M$  contains group       $O(\log n / \log \log n)$  [FS]  
                                   $\Omega(\log n / \log \log n)$

$M$  aperiodic       $O(\log \log n)$  [FMS]

$M$  contains no right absorbers       $\Omega(\log \log n / \log \log \log n)$  [M+BF+]

Very similar to circuit complexity

## Some nice open problems

$M = (\{0, 1\}, \vee)$

Can the dynamic prefix problem be done faster than  $O(\log \log n)$ ?

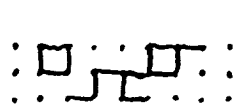
$M = \mathbb{Z}_2$

Can the dynamic prefix problem be solved with "black box" implementations of the dynamic prefix problem for  $\mathbb{Z}_3$ ?

(Analogous to Smolensky's result)

## Dynamic graph problems

### Dynamic connectivity for



Bounded  
width  
grids

$$O(\log \log n) \text{ [MS]} \\ \Omega(\log \log n / \log \log \log n)$$



Square  
grids

$$O(\log n) \\ \Omega(\log n / \log \log n)$$



General  
graphs

$$O(\sqrt{n}) \text{ [EGIS]} \\ \Omega(\log n / \log \log n)$$

Neither the time stamp method nor the compression + com. complexity methods are capable of proving larger lower bounds than  $\Omega(\log n / \log \log n)$  for dynamic language membership problems.

So we don't know any larger lower bounds. Can all of P be done in time  $O(\log n / \log \log n)$ ?

Very similar to circuit complexity: We don't know if all of P can be computed by polynomial size, depth  $O(\log n / \log \log n)$  unbounded fan-in circuits.



Alternative to lower bounds:  
Structure

We do not know if undirected  
graph connectivity is in  $NC^1$

We do not believe that undirected  
graph connectivity is in  $NC^1$   
because

undirected graph connectivity is  
complete for symmetric logspace  
and

$SL = NC^1$  would be surprising

The usual notions of reduction 154  
do not preserve dynamic  
complexity (apparently).

$$L_1 \leq_d L_2$$

$\Downarrow$  def

The dynamic membership problem  
for  $L_1$  can be solved in  
 $O(\log n)$  time per operation, given  
"black box" implementations of  
the DMP for  $L_2$ , running in  
time  $O(\log n)$  per operation.

The spreadsheet problem is complete for  $P$  with respect to  $\leq_d$  so it can't be solved in time  $O(\log n)$  ( $\log^{O(1)} n$ ,  $n^{O(1)}$ ) unless all of  $P$  can be solved in time  $O(\log n)$  ( $\log^{O(1)} n$ ,  $n^{O(1)}$ ).

Proof: Copy the usual  $P$ -completeness proof for the circuit value problem.

In fact, most natural problems which are  $P$ -complete with respect to logspace-reductions are so wrt.  $\leq_d$ .

Maybe maintenance of connected components is complete for  $SL$  wrt.  $\leq_d$ , dynamic transitive closure for  $NL$  etc. ? 😊

Apparently not ! 😞

In fact, most natural problems, complete for those classes, appear not to be so wrt.  $\leq_d$ .

Explanation: Natural complete problems tend to correspond to natural models of computation.


Circuit value problem ~ circuits  
 Directed graph reachability ~  
 branching programs

Models capturing P tend not to require each input given more than once (they can be stored)

Models capturing lower classes tend to require each input given several times.

Maybe we can find rich structure orthogonal to  $NC^1 \subseteq L \subseteq SL \subseteq NL \subseteq AC^1 \subseteq P$  ?

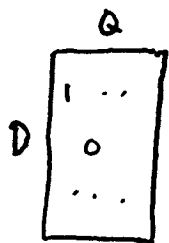


Apparently not! 

It seems that natural dynamic problems do not reduce to one another at all.

Another approach to lower bounds:  
Arithmetization.

Static problem  $f: D \times Q \rightarrow \{0,1\}^s$

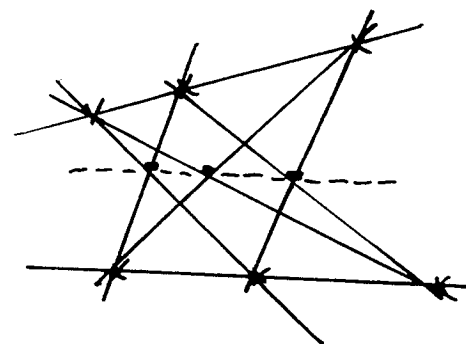


Efficient solution  $e: D \rightarrow \{0,1\}^s$   
and for each  $q \in Q$  a decision  
tree of depth  $t$  over  $\{0,1\}^s$

Arithmetization

Points  $x_1, x_2, \dots, x_{|D|}$  in  $\mathbb{A}^s$  and  
varieties  $V_1, V_2, \dots, V_{|Q|}$  of degree  $t$   
so that  $x_i \in V_j$  iff  $f(i,j) = 1$ .

Impossible configurations  
 $\Downarrow$   
Lower bounds



ex. Pappus' Theorem

But we need much stronger  
stuff than this!

# On the cell probe complexity of dynamic problems

Peter Bro Miltersen \*  
BRICS†

## 1 What are dynamic problems?

In this note, we give a survey of some results about dynamic problems with a complexity theoretic flavour. A survey on the same subject, but with a slightly different perspective, was recently given by Fredman [7].

A dynamic problem is the problem of maintaining an object in a data structure while certain operations are performed on the object, some of which change the object, and some of which answer questions about the object. Rather than a formal definition, let us look at some examples.

### The union-split-find problem

The *union-split-find* problem (on intervals) is the task of maintaining a set  $S \subseteq \{1, \dots, n\}$ , initially empty, under the following operations:

- For each  $i \in \{1, \dots, n\}$ , an operation `union( $i$ )`. It removes  $i$  from  $S$ .
- For each  $i \in \{1, \dots, n\}$ , an operation `split( $i$ )`. It inserts  $i$  into  $S$ .
- For each  $i \in \{1, \dots, n\}$ , an operation `find( $i$ )`. It returns the largest element of  $S$  which is smaller than or equal to  $i$  if such an element exists, otherwise 0 is returned.

Union-split-find is a generally useful abstract data type, for instance, it can be used to implement priority queues with small integers. The best known implementation has a worst case time per operation of  $O(\log \log n)$  [19].

---

\*This work was supported by a grant from the Danish Natural Science Research Council. It was partially supported by the ESPRIT II Basic Research Actions Program of the European Community under contract No. 7141 (project ALCOM II)

†Basic Research in Computer Science, a Centre of the Danish National Research Foundation at: Computer Science Department, Aarhus University, Ny Munkegade, DK-8000 Aarhus C, Denmark.

## Dynamic parity prefix

The dynamic parity prefix problem is the problem of maintaining a vector  $x \in \{0, 1\}^n$  under the following operations:

- For each  $i \in \{1, \dots, n\}$  and  $a \in \{0, 1\}$  an operation `change( $i, a$ )`. This operation changes  $x_i$  to  $a$ .
- For each  $j \in \{1, \dots, n\}$  an operation `prefix( $j$ )`, returning  $x_1 + x_2 + \dots + x_j \bmod 2$ .

The parity prefix problem is a one dimensional version of the range query problems, considered in computational geometry and database theory. A trivial solution gives time  $O(\log n)$  per operation. With a bit more thought, we can get time  $O(\log n / \log \log n)$  [9].

## Dynamic graph connectivity

The dynamic graph connectivity problem is the problem of maintaining an undirected graph with set of vertices  $V = \{1, \dots, n\}$  under the following operations:

- For each  $i, j \in V$  with  $i \neq j$ , an operation `insert( $i, j$ )`. This inserts an edge between  $i$  and  $j$  in the graph.
- For each  $i, j \in V$  with  $i \neq j$ , an operation `delete( $i, j$ )`. This removes the edge between  $i$  and  $j$  in the graph.
- For each  $i, j \in V$ , an operation `con( $i, j$ )`. This operation returns `true` if  $i$  and  $j$  are in the same connected component in the graph, `false` otherwise.

The best known solution to the graph connectivity problem is highly non-trivial, with a worst case time per operation of  $O(\sqrt{n})$  [4]. Dynamic graph problems had been studied intensively for a decade before this solution appeared.

## The spreadsheet problem

The (small-screen, Boolean) spreadsheet problem is the problem of maintaining  $n$  cells  $C_1, C_2, \dots, C_n$ , each cell containing either a Boolean constant (i.e. "0" or "1") or a constant size Boolean formula, with variables denoting other cells (e.g. " $C_{12} \vee C_{37}$ "), under the following operations:

- For each  $i \in \{1, \dots, n\}$ , and each formula or constant  $f$ , an operation `change( $i, f$ )` operation, which changes the contents of cell  $C_i$  to  $f$ .
- For each  $i \in \{1, \dots, n\}$ , an operation `screen( $i$ )`, which return the value of cell  $C_i$  (with "value" having the obvious semantics; if loops exist in the spreadsheet, the value is undefined).

It is well known that spreadsheets are generally useful. The trivial solution to the spreadsheet problem is to maintain the cells themselves, perform changes in constant time and when a `screen(i)` operation is called, make a topological sort of the cells, and evaluate the cells bottom up until the value of cell  $C(i)$  is known. This takes time  $O(n)$  in the worst case. This solution is patented [15]! No better solution is known.

### Our goal

We see that while the various problems seem similar, at least from a syntactical point of view, their best known solutions, some of which are trivial, some of which are deep, have very different complexities. This motivates looking at the problems from a complexity theoretic angle. The goal of doing complexity theory in this domain (as in others) are two-fold:

- Show lower bounds, hopefully establishing that the best known solutions are optimal, so that we do not have to keep on searching for better solutions.
- Gain an understanding about which properties of the problems make some of them difficult and some of them easy. This does not necessarily imply showing lower bounds, merely showing *structure* is often illuminating.

## 2 A complexity theoretic framework

### Dynamic language membership problems

In order to make systematic complexity theoretic investigations, we need a more well defined notion of dynamic problem. We will let the class of *dynamic language membership* problems be our subject of investigation:

A problem in this class is given by a language  $L \subseteq \{0, 1\}^*$ . We are supposed maintain a string  $x \in \{0, 1\}^*$  with operations:

- For each  $i \in \{1, \dots, n\}$ ,  $a \in \{0, 1\}$ , an operation `change(i, a)`. This operation changes the  $i$ 'th component of  $x$  to  $a$ .
- `query`. This operation returns `true` if  $x \in L$ , `false` otherwise.

Many naturally occurring problems can be phrased as dynamic language membership problems without changing their complexity. For instance, it is an easy exercise to see that the dynamic graph connectivity problem corresponds to the dynamic language membership problem for the language  $L = \text{USTCON}$ . The spreadsheet and prefix problems can be similarly captured. The union-split-find problem can not be captured exactly, because the find operation returns more than  $O(1)$  bits, but similar problems, like dynamic binary addition [12] can.

## The cell probe model

Various models of computation have been considered for dynamic problems:

- The *pointer* or *storage modification* machine, whose memory consists of a collection of records, each consisting of a bounded number of fields, each consisting of a pointer to other records. Interesting lower bounds [10, 16] have been shown in this model. However, since we know that arrays can be useful (e.g. for hashing), we would like our lower bounds to hold in stronger models.
- The *unit cost* RAM, where each cell in the random access memory holds an arbitrary integer. This model can simulate PRAMs [18], so it is a bit too strong.
- The *log cost* RAM, where the cost of an operation is proportional to the number of bits in the words accessed. This is a bit too weak, since we can then not follow a pointer in constant time.
- Our favourite model: The *random access computer* [2], where operations are unit cost, but each cell can only hold an integer of polynomial magnitude. Consensus seems to be emerging that this model has exactly the right level of generality. For instance, it captures all the upper bounds in Section 1, and the fact that they are the best known.

The cell probe model can be regarded as a strong (but not too strong), non-uniform version of the random access computer.

In this model, the complexity of a computation is the number of cells accessed in the random access memory containing the data structure during the computation, while the computation itself is for free. Each cell contains  $b$  bits, where  $b$  is a parameter of the model. There is no restriction on the number of cells in the memory.

Formally, the model is as follows: In an implementation of the dynamic problem we assign to each operation a decision assignment tree, i.e. a rooted tree containing *read* nodes and *write* nodes. When performing an operation we proceed from the root of its tree to one of the leaves. The read nodes are labelled with a location of the random access memory. Each has  $2^b$  sons, one for each possible content of the memory location. The write nodes, which are unary, are labelled with a memory location and a value between 0 and  $2^b - 1$ . When such a node is encountered, the value is written in the memory location. If the operation is to return an answer, this is found in the leaf finally encountered. The complexity of an operation in an implementation is the depth of its corresponding tree.

In the rest of the paper, we assume  $b = O(\log n)$ . With this setting, the cell probe model simulates the random access computer, no matter which instruction set the latter uses.



### 3 Lower bounds

#### Couting arguments

It is easy to show [14]:

**Theorem 1** *All dynamic language membership problems have complexity at most  $O(n/\log n)$ . Furthermore, almost all dynamic language membership problems have complexity at least  $\Omega(n/\log n)$ .*

So the cell probe complexity measure behaves much like more usual complexity measures for Boolean languages, like circuit size and depth. However, we want to know lower bounds for explicitly defined languages, which, for our purpose, are languages in  $P$  (The theorem above only gives us languages in  $EXPSPACE$ ). For this purpose, there seem to be only two techniques available for cell size  $b = O(\log n)$ :

- The time stamp method.
- The compression and communication complexity method.

For smaller cell sizes, there are at least three additional techniques [6, 5, 14], but they do not translate into RAC lower bounds.

#### The time stamp method

We are not going to go into the details of the time stamp method here, but only mention that the time stamp method shows a lower bound of  $\Omega(\log n / \log \log n)$  for the dynamic prefix problem mentioned above. By a reduction, the same bound holds for the dynamic graph connectivity problem [14, 17]. Furthermore, this seems to be the largest lower bound that can be shown for any problem using this technique.

#### The compression and communication complexity method

It is harder to give correct and fair citations for this method. Willard [20] used what is essentially the compression method for proving upper bounds on certain static data structure problems. Ajtai used what was essentially the communication complexity method for proving lower bounds on the static version of the union-split-find problem. He didn't phrase his proof in terms of communication complexity, which (in our view) made it hard to understand. Miltersen [11] used the compression and communication complexity technique in a weak form to show lower bounds on the cell probe complexity of dynamic problem. In [12], Miltersen noted that Ajtai's proof could be interpreted as communication complexity and combined a technical improvement of it with compression to give lower bounds for the union-split-find problem and a range of other problems.

However, Xiao, in his unpublished PhD-thesis [21], had earlier, and independently, combined compression with a stronger version of Ajtai's proof, giving stronger lower bounds for the union-split-find problem. Beame and Fich [3], upon reading [12] independently gave this stronger bound for all the problems in [12].

Fortunately, the method itself is rather easy to explain. Assume, for convenience of notation, that our dynamic problem is the problem of maintaining a set  $S \subseteq \{1, \dots, n\}$  under insertions, deletions, and some set  $Q$  of query operations. Suppose that we are given an efficient dynamic algorithm that runs in time  $t$  per operation.

- In the compression step we convert the dynamic algorithm to a solution for a *static* data structure problem, namely the problem of storing  $S$  using *small space* ( $O(|S|t)$  cells) so that any query in  $Q$  can be answered in time  $t$ . Basically, this is done by inserting the elements of  $S$  in our dynamic data structure, noting which memory locations have changed value, and storing those in a perfect hash table [8].
- In the next stage we convert the solution to the static problem into an efficient protocol for the following communication game between Alice and Bob:
  - Alice is given a query  $q \in Q$ .
  - Bob is given a subset  $S \subseteq \{1, \dots, n\}$ .

Alice is allowed to send messages that contains  $\log |S|$  bits to Bob, while Bob is allowed to send messages that contain  $\log n$  bits to Alice. The object of the game is for Alice to find out the answer to query  $q$  about  $S$ .

The efficient protocol is as follows: Bob computes the static data structure corresponding to  $S$ , but does not send anything yet. Then Alice simulates the query operation corresponding to  $q$  by sending Bob requests for the cells she wants to read in his data structure. Bob sends the content of the cell in question back. This is repeated until the query operation is completed and Alice knows the answer, i.e.. for at most  $t$  rounds.

We can now use communication complexity techniques to give a lower bound on the communication game and translate this bound back to a lower bound on the dynamic problem.

Examples include an  $\Omega(\log \log n / \log \log \log n)$  lower bound for the union-split-find problem [21, 3]. The largest bound the technique is able to give for any problem is  $\Omega(\log n / \log \log n)$ . This bound is achieved for a language  $L$ , related to polynomial evaluation over finite fields [13].

## 4 Structure

The largest lower bound we can show for an explicit problem by the known techniques is  $\Omega(\log n / \log \log n)$ . It is therefore an open problem if all dynamic language membership problems in  $P$  can be solved in time  $O(\log n / \log \log n)$  per operation.

Thus, we are far from showing that the best known algorithms for e.g. the dynamic connectivity and the spreadsheet problems are optimal. However, from traditional complexity theory, where large lower bounds are also hard to find, we know an alternative: structure. It would be nice to be able to claim that these problems are difficult, because each are *hard* for a large class of natural problems.

Of course, we need to define a notion of reduction that preserves dynamic complexity. One was defined in [14], here's another, a bit more intuitive, and sufficient for our purpose:

For two languages  $L_1$  and  $L_2$ , we say that  $L_1 \leq_d L_2$ , if the dynamic language membership problem for  $L_1$  can be solved in time  $O(\log n)$  if we assume access to black box (oracle) implementations of the dynamic language membership problem for  $L_2$  that runs in time  $O(\log n)$ .

We can now show, by emulating the usual completeness proof for the circuit value problem:

**Theorem 2** *The spreadsheet problem is complete for  $P$  w.r.t.  $\leq_d$ .*

Thus, the spreadsheet problem can not be solved in time  $O(\log n)$  (or  $\log^{O(1)} n$  or  $n^{o(1)}$ ) per operation, unless all of  $P$  can.

Indeed, it seems that almost all natural  $P$ -complete (with respect to e.g. first order projections or whatever your favourite notion of low level reducibility is) problems are  $P$ -complete w.r.t.  $\leq_d$ .

We might now reasonably expect that by a similar argument dynamic graph connectivity is complete for the class  $SL$  since  $USTCON$  is first order complete for  $SL$ .

Unfortunately, this does not seem to be the case. In general, it seems that almost none of the natural first order complete problems for the usual classes (defined in terms of small space or parallel time) smaller than  $P$  are complete w.r.t.  $\leq_d$ .

Examples disobeying these rules of thumb can be constructed: languages, first order complete for  $P$  with efficient dynamic solutions exist, and so do problems which are first order complete, as well as  $\leq_d$ -complete, for  $SL$  [14], but neither are particularly natural when regarded as combinatorial problems.

Though it is hard to find problems  $\leq_d$ -complete for usual complexity classes, we could hope for new structure: The dynamic graph connectivity problem might be complete for a large class of natural problems, different from  $SL$ . Unfortunately, this does not seem to be the case either:  $\leq_d$  seems to be too

weak a reduction for much structure to appear. Therefore, a rich structural complexity theory of dynamic problems seems unlikely.

## References

- [1] M. Ajtai, A lower bound for finding predecessors in Yao's cell probe model, *Combinatorica* **8** (1988) 235-247.
- [2] D. Angluin, L.G. Valiant, Fast probabilistic algorithms for Hamiltonian circuits and matchings, *J. Comput. System Sci.* **18** (1979) 155-193.
- [3] P. Beame, F. Fich, Personal Communication.
- [4] D. Eppstein, Z. Galil, G. F. Italiano, T. H. Spencer, Separator based sparsification for dynamic planar graph algorithms, in: *Proc. 25th ACM Symp. on Theory of Computing* (1993) 208-217.
- [5] G.S. Frandsen, P.B. Miltersen, S. Skyum, Dynamic Word Problems, in: *Proc. 34rd IEEE Symposium on Foundations of Computer Science* (1993) 470-479.
- [6] M.L. Fredman, The complexity of maintaining an array and computing its partial sums, *J. Assoc. Comput. Mach.* **29** (1982) 250-260.
- [7] M.L. Fredman, Lower bounds for dynamic algorithms, in: *Proc. 4th Scandinavian Workshop on Algorithm Theory* (1994) 167-171.
- [8] M.L. Fredman, J. Komlós, E. Szemerédi, Storing a sparse table with  $O(1)$  worst case access time, *J. Assoc. Comput. Mach.* **31** (1984) 538-544.
- [9] M.L. Fredman, M.E. Saks, The cell probe complexity of dynamic data structures, in: *Proc. 21st Ann. ACM Symp. on Theory of Computing* (1989) 345-354.
- [10] K. Mehlhorn, S. Näher, H. Alt, A lower bound on the complexity of the union-split-find problem, *SIAM J. Comput.* **17**(1988) 1093-1102.
- [11] P.B. Miltersen, The bit probe complexity measure revisited, in: *Proc. 10th Symp. on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Vol. 665 (Springer, Berlin, 1993) 662-671.
- [12] P.B. Miltersen, Lower bounds for Union-Split-Find related problems on random access machines, in: *Proc. 26th ACM Symposium on Theory of Computing* (1994) 625-634.
- [13] P.B. Miltersen, On the cell probe complexity of polynomial evaluation, manuscript 1994.

- [14] P.B. Miltersen, S. Subramanian, J.S. Vitter, R. Tamassia, Complexity models for incremental computation, *Theoretical Computer Science* **130** (1994) 203-236.
- [15] R.K. Pardo, R. Landau, Process and apparatus for converting a source program into an object program, US patent # 4,398,249, filed Aug 12, 1970, granted Aug 9, 1983.
- [16] J.A. La Poutré, Lower bounds for the Union-Find and the Split-Find problem on pointer machines, in: *Proc. 20th Ann. ACM Symp. on Theory of Computing* (1990) 34-44.
- [17] M. Rauch, Improved data structures for fully dynamic binconnectivity, in: *Proc. 26th ACM Symposium on Theory of Computing* (1994) 686-695.
- [18] A. Schönhage, On the Power of Random Access Machines, in: *Proc. 6th Annual International Colloquium on Automata, Languages and Programming* (1979) 520-529.
- [19] P. Van Emde Boas, R. Kaas, E. Zijlstra, Design and implementation of an efficient priority queue, *Math. Systems Theory* **10** (1977) 99-127.
- [20] D.E. Willard, Log-logarithmic worst case range queries are possible in space  $\Theta(n)$ , *Inform. Process. Lett.* **17** (1983) 81-84.
- [21] B. Xiao, New bound in cell probe model, Doctoral Dissertation, University of California, San Diego, 1992.

**Lecturer: Michael Ben-or**

**Title: On Algebraic Complexity Theory**

No material, since blackboard was used.

Institute of Computer Science  
Hebrew University  
Jerusalem, Israel  
E-mail: [benor@CS.HUJI.AC.IL](mailto:benor@CS.HUJI.AC.IL)



**Lecturer: Christoph Meinel**

**Title: Modular Communication Complexity of UCON**

Material: Christoph Meinel & Stephan Waack: *The Möbius Function, Variations Ranks, and  $\Theta(n)$ -Bounds on the Modular Communication Complexity of the Undirected Graph Connectivity Problem*

Fachbereich IV - Informatik  
Universität Trier  
D-54286 Trier, Germany  
E-mail: meinel@uni-trier.de



# UNIVERSITÄT TRIER

## Mathematik / Informatik

Forschungsbericht Nr. 94-04

*The Möbius Function, Variations Ranks, and  
 $\Theta(n)$ -Bounds  
on the Modular Communication Complexity  
of the Undirected Graph Connectivity Problem*

Christoph Meinel  
Lehrstuhl Theoretische Informatik  
Fachbereich IV - Informatik  
Universität Trier  
D-54286 Trier  
Stephan Waack  
Institut für Numerische  
und Angewandte Mathematik  
Georg-August-Universität Göttingen  
Lotzestr. 16-18  
D-37083 Göttingen

### Electronic copies of technical reports are available:

- Via FTP: Host [ftp.informatik.uni-trier.de](ftp://ftp.informatik.uni-trier.de), directory /pub/reports
- Via gopher: Host [gopher.informatik.uni-trier.de](gopher://gopher.informatik.uni-trier.de), port 70, type 1,  
paths 1/TechReports/Abstracts and 1/TechReports/FullText
- Via WWW: URL <http://www.informatik.uni-trier.de/Reports/List>
- Via email: Send a mail to [ftpmail@ftp.informatik.uni-trier.de](mailto:ftpmail@ftp.informatik.uni-trier.de), subject  
'MAIL ME CLEAR', body 'TechReports.HowTo' followed  
by an empty line, for detailed instructions

### Printed copies:

Trierer Forschungsberichte  
Fachbereich IV -  
Mathematik / Informatik  
Universität Trier  
D-54286 Trier  
ISSN 0944-0488

# The Möbius Function, Variations Ranks, and $\Theta(n)$ -Bounds on the Modular Communication Complexity of the Undirected Graph Connectivity Problem

Christoph Meinel  
Lehrstuhl Theoretische Informatik  
Fachbereich IV - Informatik  
Universität Trier  
D-54286 Trier

Stephan Waack  
Institut für Numerische  
und Angewandte Mathematik  
Georg-August-Universität Göttingen  
Lotzestr. 16-18  
D-37083 Göttingen

## Abstract

We prove that the modular communication complexity of the undirected graph connectivity problem  $\text{UCONN}$  equals  $\Theta(n)$ , in contrast to the well-known  $\Theta(n \log n)$  bound in the deterministic case (see [9]), and to the  $\Omega(n \log \log n)$  lower bound in the nondeterministic case, recently proved by Raz and Spieker (see [15]).

We obtain our result by combining Möbius function techniques due to Lovasz and Saks (see [12], [13]) with rank and projection reduction arguments.

**Topics:** Computational Complexity, Communication Protocols, Modular Acceptation Modes, Undirected Graph Connectivity.

## Introduction

During the last few years communication complexity theory gained popularity. In several papers many interesting questions of complexity theory were answered by reducing them to several kinds of communication games. Among others, this regards time-area tradeoffs for VLSI-circuits [1], [10], time-space tradeoffs for Turing machines, width-length tradeoffs for oblivious and usual  $\Omega$ -branching programs ([2],[4]), branching programs of bounded alternation [14], and threshold circuits of depth 2 [11] and depth 3 [7].

The *graph connectivity problem for undirected graphs*  $\text{UCONN} = (\text{UCONN}_{n(n-1)})_{n \in \mathbf{N}}$  in distributed form can be formulated as follows. Assume that we are given two not necessarily edge-disjoint undirected graphs  $G_1 = (V, E_1)$  and  $G_2 = (V, E_2)$  on a common  $n$ -set of vertices  $V$ , where both graphs are represented as Boolean vectors of length  $\binom{n}{2}$ . The question is whether or not the graph  $G \stackrel{\text{def}}{=} G_1 \cup G_2 = (V, E_1 \cup E_2)$  is connected, i.e. each pair of vertices

in  $G$  is connected. In [18] the major developments in understanding the complexity of the graph connectivity problem in several computational models are surveyed.

In the following we investigate the modular communication complexity of UCONN. Let two graphs  $G_i = (V, E_i)$ , for  $i = 1, 2$ , be given to two processors  $\mathcal{P}_1$  and  $\mathcal{P}_2$ . In order to solve UCONN both processors have to communicate via a common communication tape. The computation of the whole structure, which is called a *communication protocol* or simply a *protocol*, is going on in *rounds*. Starting with  $\mathcal{P}_1$  the processors write alternately bits on the communication tape. These bits depend on the input available to the processor which is to move and on the bits already written on the communication tape before. We assume without loss of generality, that in each round exactly one bit is written down on the communication tape. If the last bit written on the communication tape is “1” or “0” the computation is called *accepting* or *rejecting*, respectively. So co-operative computations can be thought of as to be Boolean strings. The length of the string is the *communication complexity* of the computation. Since we consider the worst-case-complexity in this paper, we assume without loss of generality, that all computations of a protocol are of equal length, say  $L$ . We shall assume the processors to be nondeterministic. That’s why we have to define the *output* of a protocol via defining *acceptation modes*. As it is common use an acceptation mode is called a *counting mode* if the output of a protocol for a given input depends only on the numbers of accepting and rejecting computations performed by the protocol accessing this input. In this paper we discuss *the modular acceptation modes* in which the protocol accepts an input, if the number of accepting computations is not equal to 0 modulo  $m$ .

How to motivate the modular acceptation modes modulo  $m$ ? In [20] it has been shown that all problems computable by constant depth, polynomial size circuits with  $\text{MOD}_m$ -gates for arbitrary integers  $m$ , are contained in certain counting communication complexity classes. In [5] these modes were formally introduced and studied. Several papers (see e.g. [6]) deal with comparing the power of different counting acceptation modes. Roughly speaking, the computational power of the acceptation modes modulo  $m_i$ ,  $i = 1, 2$ , is uncomparable, provided that  $(m_1, m_2) = 1$  (see [8]).

We conclude this section by reviewing the results and methods which are strongly related to ours and by formulating the result of this paper. We use the notions and notations of Definition 1. Hajnal, Maass, and Turan proved in [9] the following theorem.

**Theorem A**  $\text{Comm}(\text{UCONN}_{n(n-1)}) = \Theta(n \log n)$ . □

Their method involves the use of the Möbius function  $\mu$  for the lattice of partitions of an  $n$ -set. Lovasz and Saks extended in [12] and [13] this ideas to a large class of problems, the so-called *meet problems for finite lattices*, which can be formulated as follows. Let  $S$  be a finite lattice, and let both processor  $\mathcal{P}_1$  and  $\mathcal{P}_2$  be given an element  $x$  and  $y$ , respectively. Then they have to decide whether  $x \wedge y = 0$ .

**Theorem B** *Let  $\text{MEET}_S$  be the meet problem of a finite lattice. Let  $S$  have  $a$  atoms and  $b$  Möbius elements (i.e. elements  $x$  such that  $\mu(0, x) \neq 0$ ). Then*

$$\log b \leq \text{Comm}(\text{MEET}_S) \leq (\log a)(\log b).$$

□

Recently, Raz and Spieker [15] proved

**Theorem C** *If processor  $\mathcal{P}_1$  as well as processor  $\mathcal{P}_2$  have an bipartite perfect matching on  $2n$  vertices with two colors of size  $n$  as an input, and if their goal is to determine whether the union of the two matchings forms a Hamiltonian cycle, the nondeterministic communication complexity of the problem is  $\Omega(n \log \log n)$ .  $\square$*

Since the problem of Theorem C is a subproblem of UCONN (see Lemma 2), it follows

**Corollary D**  $N\text{-Comm}(\text{UCONN}_{n(n-1)}) = \Omega(n \log \log n)$ .  $\square$

It is the aim of this paper to show that modular acceptance modes help for detecting undirected graph connectivity.

**Theorem** *Let  $m$  be arbitrary. Then  $\text{MOD}_m\text{-Comm}(\text{UCONN}_{n(n-1)}) = \Theta(n)$ .*

**Proof.** The claim follows directly from Proposition 2 in Section 3 and from Proposition 4 in Section 4.  $\square$

We use the technique related to the Möbius function to prove the upper bound of Proposition 2. The lower bound of Proposition 4 follows from rank and reduction arguments.

## 1 The computational model

Let  $f : S_1 \times S_2 \rightarrow \{0, 1\}$  be given in distributed form. A *protocol of length  $L$*  consisting of two processors  $\mathcal{P}_1$  and  $\mathcal{P}_2$  that access inputs of  $S_1$  and  $S_2$ , respectively, can be described by two functions

$$\Phi_i : S_i \times \{0, 1\}^{*L} \rightarrow \{0, 1\},$$

$i = 1, 2$ , and  $\{0, 1\}^{*L} = \{w \in \{0, 1\}^* \mid 1 \leq |w| \leq L\}$ . The interpretation is as follows. Let  $\gamma = \gamma_1 \dots \gamma_j$ ,  $\gamma_k \in \{0, 1\}$ . If  $\Phi_i(s_i, \gamma) = 1$ , and if  $|\gamma| - i$  is even, then the corresponding processor  $\mathcal{P}_i$  is able to write  $\gamma_j$  on the communication tape provided that it has read  $\gamma_1 \dots \gamma_{j-1}$  on the communication tape and that it has  $s_i$  as input. If, however,  $\Phi_i(s_i, \gamma) = 0$ , then  $\mathcal{P}_i$  is not able to write  $\gamma_j$ .

Now we define two  $\#S_1 \times \#S_2$ -matrices  $Acc^P$  and  $Rej^P$  associated with the protocol  $P$  of length  $L$  by

$$Acc_{s_1, s_2}^P \stackrel{\text{def}}{=} \sum_{\gamma_1 \dots \gamma_L \in \{0, 1\}^L, \gamma_L = 1} \prod_{j=1}^L \Phi_{1+((j+1) \bmod 2)}(s_{1+((j+1) \bmod 2)}, \gamma_1 \dots \gamma_j) \quad (1)$$

$$Rej_{s_1, s_2}^P \stackrel{\text{def}}{=} \sum_{\gamma_1 \dots \gamma_L \in \{0, 1\}^L, \gamma_L = 0} \prod_{j=1}^L \Phi_{1+((j+1) \bmod 2)}(s_{1+((j+1) \bmod 2)}, \gamma_1 \dots \gamma_j) \quad (2)$$

Clearly,  $Acc_{s_1, s_2}^P$  gives the number of accepting computations of the protocol  $P$  on the input  $(s_1, s_2)$ , whereas  $Rej_{s_1, s_2}^P$  is the number of the rejecting computations. In order to make this

approach unique, we agree that  $\Phi_i(s_i, \gamma) = 1$ , if  $|\gamma| - i$  is odd, for  $i = 1, 2$ . We may give an equivalent definition of the above two matrices as follows. Let  $\gamma \in \{0, 1\}^L$  be a computation. Define

$$\chi_i^P(s_i, \gamma) \stackrel{def}{=} \prod_{\gamma' \in \{0, 1\}^{*L}, \gamma' \leq \gamma} \Phi_i(s_i, \gamma'), \quad (3)$$

for  $i = 1, 2$ . Then we get directly from the equations (1) and (2)

$$Acc_{s_1, s_2}^P = \sum_{\gamma \in \{0, 1\}^L, \gamma_L = 1} \chi_1^P(s_1, \gamma) \cdot \chi_2^P(s_2, \gamma) \quad (4)$$

$$Rej_{s_1, s_2}^P = \sum_{\gamma \in \{0, 1\}^L, \gamma_L = 0} \chi_1^P(s_1, \gamma) \cdot \chi_2^P(s_2, \gamma) \quad (5)$$

**Definition 1** 1. A counting acceptance mode  $\mu$  for a protocol  $P$  is a function  $\mu : \mathbb{N}^2 \rightarrow \{0, 1\}$  such that  $P$  accepts an  $(s_1, s_2)$ , if and only if,  $\mu(Acc_{s_1, s_2}^P, Rej_{s_1, s_2}^P) = 1$ . Otherwise  $P$  rejects the input. A protocol  $P$  equipped with an acceptance mode  $\mu$  is called a  $\mu$ -protocol. The function computed is sometimes denoted by  $\text{Comp}(P, \mu)$ . If we are given a function  $f : S_1 \times S_2 \rightarrow \{0, 1\}$  then  $\mu\text{-Comm}(f) \stackrel{def}{=} \min\{L \mid \text{Comp}(P, \mu) = f, L \text{ is the length of } P\}$ .

2. We define the following acceptance modes.

$$\begin{aligned} \text{Nondeterministic mode:} \quad N(n_1, n_2) &= 1 \stackrel{def}{\iff} n_1 > 0, \\ \text{Modular modes:} \quad \text{MOD}_m(n_1, n_2) &= 1 \stackrel{def}{\iff} n_1 \not\equiv 0 \pmod{m}, \end{aligned}$$

By the way, a deterministic communication protocol is not characterized by a special acceptance mode but by a property of the underlying protocol, namely  $\Phi_i(s_i, \gamma 0) + \Phi_i(s_i, \gamma 1) \leq 1$ , for  $s_i \in S_i$ ,  $i = 1, 2$ , and  $\gamma \in \{0, 1\}^*$ . For such protocols all reasonable counting modes coincide.

**Lemma 1** If  $m_1 \mid m_2$ , then  $\text{MOD}_{m_2}\text{-Comm}(f) \leq \log\left(\frac{m_2}{m_1}\right) \cdot \text{MOD}_{m_1}\text{-Comm}(f)$ , for each function  $f$ .

**Proof.** Clearly,  $m_2 \mid \frac{m_2 \cdot m_2}{m_1}$ , if and only if,  $m_1 \mid m_2$ .

Let  $P$  be the  $\text{MOD}_{m_1}$ -protocol for  $f$ . We describe the following protocol  $P'$ .

First, processor  $\mathcal{P}_1$  chooses nondeterministically an index  $k$ ,  $1 \leq k \leq \frac{m_2}{m_1}$  and sheds  $k$ .

Second,  $\mathcal{P}_2$  and  $\mathcal{P}_1$  proceed in the same way as  $\mathcal{P}_1$  and  $\mathcal{P}_2$  do according to the protocol  $P$ . We get that  $Acc_{ij}^{P'} = \frac{m_2}{m_1} \cdot Acc_{ij}^P$ . Consequently,  $Acc_{ij}^{P'} \equiv 0 \pmod{m_2} \iff Acc_{ij}^{P'} \equiv 0 \pmod{m_1}$ . If  $L$  is the length of protocol  $P$ , then  $\log\left(\frac{m_2}{m_1}\right) \cdot L$  is the length of protocol  $P'$ .  $\square$

Now we have to define what we mean by reductions. Fortunately, this is much easier here than in machine-based complexity theory.

**Definition 2** Let  $F = (f_{2n} : \Sigma^n \times \Sigma^n \rightarrow \{0, 1\})_{n \in \mathbb{N}}$  and  $G = (g_{2q} : \Gamma^n \times \Gamma^n \rightarrow \{0, 1\})_{n \in \mathbb{N}}$  be two decision problems. We say that  $F$  is rectangular reducible to  $G$  with respect to  $q$ , where  $q : \mathbb{N} \rightarrow \mathbb{N}$  is a nondecreasing function, iff there are two transformations  $l_n, r_n : \Sigma^n \rightarrow \Gamma^{q(n)}$  such that for all  $n$  and for all  $\vec{x}, \vec{y} \in \Sigma^n$  we have  $f_{2n}(\vec{x}, \vec{y}) = g_{2q(n)}(l_n(\vec{x}), r_n(\vec{y}))$ . We write  $F \leq_{rec}^q G$ .

We can utilize rectangular reductions for proving lower bounds. Let  $q : \mathbb{N} \rightarrow \mathbb{N}$  be an unbounded nondecreasing function. Then we define  $q^{(-1)}$  by  $q^{(-1)}(i) = \max\{j \mid q(j) \leq i\}$ . For example let  $\rho : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  be an unbounded monotone increasing continuous function and let  $\rho^{-1} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  be a right-inverse to  $\rho$ , i. e.  $\rho \circ \rho^{-1} = 1$ . If we define  $q : \mathbb{N} \rightarrow \mathbb{N}$  to be  $q(i) = \lceil \rho(i) \rceil$ , then  $q^{(-1)}(i) = \lfloor \rho^{-1}(i) \rfloor$ , for almost all natural numbers.

The proof of the following lower bound reduction argument is easy.

**Lemma 2** *Assume that we are given two sequences of functions  $F = (f_{2n} : \Sigma^n \times \Sigma^n \rightarrow \{0, 1\})_{n \in \mathbb{N}}$  and  $G = (g_{2n} : \Gamma^n \times \Gamma^n \rightarrow \{0, 1\})_{n \in \mathbb{N}}$ . If  $\mu\text{-Comm}(F) \geq c(n)$  and if  $F \leq_{rec}^q G$ , then  $\mu\text{-Comm}(G) \geq c \circ q^{(-1)}(n)$ .  $\square$*

One efficient way to get rectangular reductions is to handle with projection reductions. The variables over  $\{0, 1\}^n$  are coordinate functions  $x_i : \{0, 1\}^n \rightarrow \{0, 1\}$  such that  $x_i(\sigma_1, \dots, \sigma_n) = \sigma_i$ . In accordance with Skyum and Valient (see [17]) we define.

**Definition 3** 1. Let  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g_m : \{0, 1\}^m \rightarrow \{0, 1\}$ .  $f_n$  is called reducible to  $g_m$  via a projection  $\pi_n : \{y_1, \dots, y_m\} \rightarrow \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n, 0, 1\}$  and we write  $f_n \leq_{\pi_n} g_m$ , where the  $x_i$  and the  $y_j$  are the Boolean variables of  $F_n$  and  $g_m$ , resp., if

$$f_n(x_1, \dots, x_n) = g_m(\pi(y_1), \dots, \pi(y_m)).$$

2. If  $f_n$  and  $g_m$  are given in distributed form, i. e.  $f_n : \{0, 1\}^{n/2} \times \{0, 1\}^{n/2} \rightarrow \{0, 1\}$  and  $g_m : \{0, 1\}^{m/2} \times \{0, 1\}^{m/2} \rightarrow \{0, 1\}$ , then we say that the reduction  $\pi$  respects the distribution of the variables, if

$$\pi_n^{-1}\{x_1, \dots, x_{n/2}, \neg x_1, \dots, \neg x_{n/2}\} \subseteq \{y_1, \dots, y_{m/2}\}$$

and

$$\pi_n^{-1}\{x_{n/2+1}, \dots, x_n, \neg x_{n/2+1}, \dots, \neg x_n\} \subseteq \{y_{m/2+1}, \dots, y_m\}.$$

3. There is a transpose  $\pi_n^t : \{0, 1\}^m \rightarrow \{0, 1\}^n$  of the projection reduction  $\pi$ . It is defined by

$$\pi_n^t(\vec{u}) = (\pi_n(y_1)(\vec{u}), \dots, \pi_n(y_m)(\vec{u})),$$

where  $\vec{u} = (x_1(\vec{u}), \dots, x_n(\vec{u})) \in \{0, 1\}^m$  is any Boolean vector of length  $n$ .

4. If  $F = (f_n)_{n \in \mathbb{N}}$  and  $G = (g_n)_{n \in \mathbb{N}}$  are sequences of functions, if  $\Pi = (\pi_n)_{n \in \mathbb{N}}$  is a sequence of projection reductions defined in the first item of this definition, i. e.  $f_n \leq_{\pi_n} g_m$ , and if  $m \leq p(n)$ , then we say that  $\Pi$  is a  $p(n)$ -projection reduction and we write  $F \leq_{\Pi}^p G$ . If both  $F$  and  $G$  are given in distributed form, then the definition of the notion "  $\pi$  respects the distribution of the variables" can be done by analogy with the second item of this definition.

If the elements of  $\{0, 1\}^n$  are representations of graphs, then we visualize the graph which is the transpose  $\pi_n^t(\vec{\sigma})$  of a vector  $\vec{\sigma} \in \Sigma^n$  in such a way that the edges which are not constant are labelled by the corresponding literal (see figures 1 and 2). The meaning is that such an edge belongs to the graph, if and only if, the labelling literal is true.

Due to Lemma 2 we get

**Lemma 3** Assume that we are given two sequences of functions  $F = (f_{2n} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\})_{n \in \mathbb{N}}$  and  $G = (G_{2m} : \{0,1\}^m \times \{0,1\}^m \rightarrow \{0,1\})_{m \in \mathbb{N}}$  such that  $F \leq_{\Pi}^p G$ , where  $p : \mathbb{N} \rightarrow \mathbb{N}$  is increasing, and  $\Pi = (\pi_n)_{n \in \mathbb{N}}$  is a sequence of projection reductions which respects the distribution of the variables. If  $\mu\text{-Comm}(F) \geq c(n)$ , then  $\mu\text{-Comm}(G) \geq c \circ q^{(-1)}(n)$ .  $\square$

## 2 Rank arguments for upper and lower bounds

We shall derive rank arguments for proving upper and lower bounds on the length of protocols equipped with the modular acceptance modes from Definition 1. We adopt the concept of variation ranks of communication matrices developed in [11]. Throughout this section let  $f$  denote a function  $f : S_1 \times S_2 \rightarrow \{0,1\}$ ,  $N = \#S_1 = \#S_2$ , and let  $M^f$  denote the communication matrix, where  $M_{i,j}^f = f(i,j)$ , for  $i, j = 1, \dots, N$ .

Let the *sequence equality function* be defined by  $\text{SEQ}_{2n}(x_1, \dots, x_n, y_1, \dots, y_n) = \bigwedge_{i=1}^n (1 - ((x_i + y_i) \bmod 2))$ . Here  $S_1 = S_2 = \{0,1\}^n$ .

**Definition 4** 1. Two  $N \times N$ -matrices  $A$  and  $B$  over the ring of integers are defined to be  $\text{mod}_m$ -equivalent, where  $m$  is a natural number, if and only if, for all indices  $i, j$ ,

$$a_{ij} \equiv 0 \pmod{m} \iff b_{ij} \equiv 0 \pmod{m}.$$

2. Let  $A$  be an integer matrix. We define  $\text{var-rank}_{\mathbb{Z}/m\mathbb{Z}}(A \bmod m)$  to mean the minimum of all numbers  $\text{rank}_{\mathbb{Z}/m\mathbb{Z}}(B \bmod m)$ , where  $B$  is an integer matrix which is  $\text{mod}_m$ -equivalent to  $A$ .

A 0-1 matrix is interpreted as an  $R$ -matrix, where  $R$  is an arbitrary semiring, in the canonical way. As usual, the  $R$ -rank of a  $m \times n$ -matrix  $A$  over  $R$ , which we denote by  $\text{rank}_R A$ , is defined to be the minimal number  $k$  such that  $A = B \cdot C$ , where  $B$  is a  $m \times k$ -matrix and  $C$  is a  $k \times n$ -matrix over  $R$ . A straightforward calculation yields the next lemma.

**Lemma 4** Let  $A$  be an integer matrix.

1.  $\text{rank}_{\mathbb{Z}/m\mathbb{Z}}(A \bmod m) = \max\{\text{rank}_{\mathbb{Z}/m_i\mathbb{Z}}(A \bmod m_i) \mid i = 1, \dots, r\}$ ,  
provided that  $m = m_1 \cdot \dots \cdot m_r$ , where  $(m_i, m_j) = 1$ , for all  $i \neq j$ .

2.  $\text{rank}_{\mathbb{Z}/m\mathbb{Z}}(A \bmod m) = \min\{\text{rank}_{\mathbb{Z}} D \mid D \text{ is } \text{mod}_m\text{-equivalent to } A\}$ .  $\square$

**Lemma 5** Let  $R$  be any semiring. Let  $P$  be a protocol of the length  $L$  on the input set  $S_1 \times S_2$ ,  $\#S_1 = \#S_2 = N$ , and let  $\text{Acc}^P$  be the  $N \times N$ -matrix defined in equation 1. Then  $\text{rank}_R(\text{Acc}^P) \leq 2^{L-1}$ .

**Proof.** The inequality follows directly from equation 4.  $\square$

Now we can fully characterize the modular communication complexity in terms of variation ranks.

**Proposition 1**

$$\log_2 \left( \text{var-rank}_{\mathbb{Z}/m\mathbb{Z}}(M^f) \right) \leq \text{MOD}_m\text{-Comm}(f) \leq \log_2 \left( \text{var-rank}_{\mathbb{Z}/m\mathbb{Z}}(M^f) \right) + 2\log_2 m + 1.$$

**Proof.** The left inequality follows directly from Lemma 5 and from Definition 4. Let us turn to the right one. We choose by Lemma 4 an integer matrix  $B$  which is  $\text{mod}_m$ -equivalent to  $M^f$ , such that  $r = \text{rank}_{\mathbb{Z}/m\mathbb{Z}}(B \bmod m) = \text{var-rank}_{\mathbb{Z}/m\mathbb{Z}}(M^f)$ . Then  $B = B^{(1)} + \dots + B^{(r)}$ , where the  $B^{(k)}$  have  $\mathbb{Z}/m\mathbb{Z}$ -rank 1. This is equivalent to  $B_{ij}^{(k)} \equiv U_i^{(k)} \cdot V_j^{(k)} \pmod{m}$ , for  $U_i^{(k)}, V_j^{(k)} \in \{1, \dots, m\}$ , and for  $i, j = 1, \dots, N$ .

Now we can describe the following protocol  $P$ . Assume that the input is  $(i, j) \in S_1 \times S_2$ .

First, processor  $\mathcal{P}_1$  chooses nondeterministically some indices  $k, 1 \leq k \leq r$ , and  $l_1, 1 \leq l_1 \leq U_i^{(k)}$ , and sheds  $(k, l_1)$ .

Second, processor  $\mathcal{P}_2$  chooses nondeterministically some index  $l_2, 1 \leq l_2 \leq V_j^{(k)}$ , and sheds  $(l_2, 1)$ .

Clearly, there are  $\sum_{k=1}^r U_i^{(k)} \cdot V_j^{(k)} \equiv B_{ij} \pmod{m}$  many accepting computations assigned to the input  $(i, j)$ . It follows that  $\text{Comp}(P, \text{MOD}_m) = f$ . Obviously, the length of the protocol is bounded above by  $\log_2 r + 2\log_2 m + 1$ .  $\square$

In the case of  $m$  being a prime number, we can even do better.

**Corollary 1** *If  $m = p$  is a prime number, we have*

$$\frac{1}{p-1} \cdot \log_2 \left( \text{rank}_{\mathbb{Z}/p\mathbb{Z}}(M^f) \right) \leq \text{MOD}_p\text{-Comm}(f) \leq \frac{1}{p-1} \cdot \left( \log_2 \left( \text{rank}_{\mathbb{Z}/p\mathbb{Z}}(M^f) \right) + 2\log_2 p + 1 \right).$$

**Proof.** By means of Fermat's Little Theorem each protocol of length  $L$  can be transformed into a protocol  $P'$  of length  $(p-1)L$  such that for all inputs  $(i, j)$

$$\text{Acc}_{ij}^{P'} = \left( \text{Acc}_{ij}^P \right)^{p-1} \equiv \begin{cases} 0 \pmod{p} & \text{if } \text{Acc}_{ij}^P \equiv 0 \pmod{p}; \\ 1 \pmod{p} & \text{if } \text{Acc}_{ij}^P \not\equiv 0 \pmod{p}. \end{cases}$$

$\square$

### 3 The Möbius function and upper bounds on the length of $\text{MOD}_m$ -protocols for undirected graph connectivity

In this section we transform a method due to Lovasz and Saks (see [12], [13]) for proving lower bounds on the length of deterministic protocols to the case of  $\text{MOD}_m$ -protocols in order to prove upper bounds. We can only give a very brief treatment on Möbius functions. For more see [16].

Let  $S$  be a finite partially ordered set,  $R$  be a commutative ring with 1. The  $R$ -valued incidence algebra  $\mathcal{A}(S, R)$  is defined as follows. Consider the set of functions of two variables



$f(x, y)$ , for  $x$  and  $y$  ranging over  $S$  having values in  $R$ , and with the property that  $f(x, y) = 0$  whenever  $x \not\leq y$ . The sum and the multiplication by scalars are defined pointwise. The product of  $f$  and  $g$  is defined as follows.

$$(fg)(x, y) \stackrel{\text{def}}{=} \sum_z f(x, z)g(z, y)$$

Clearly, Kronecker's  $\delta$ -function is the 1 of  $\mathcal{A}(S, R)$ . The  $R$ -valued zeta function  $\zeta(x, y) \in \mathcal{A}(S, R)$  is defined by  $\zeta(x, y) = 1$  if  $x \leq y$  and  $\zeta(x, y) = 0$  otherwise. The function  $\iota(x, y) \stackrel{\text{def}}{=} \zeta(x, y) - \delta(x, y)$  is called the *incidence function*.

The following formula is the key to prove Lemma 6.

$$g(x, y) = -\frac{1}{f(x, x)} \sum_z f(x, z)g(z, y)\iota(z, y) \quad (6)$$

It allows a recursive definition of the inverse of  $f$ , provided that the  $f(x, x)$  are units in  $R$ .

**Lemma 6** *An element of  $\mathcal{A}(S, R)$  is a unit, if and only if,  $\prod_x f(x, x)$  is a unit in  $R$ .  $\square$*

Consequently, we can define *the  $R$ -valued Möbius function* to be the inverse of the zeta function. Let us denote this function for a moment by  $\mu^{(R)}$ .

Analogously to the standard real-valued case, we have the *Möbius inversion formula*. Let  $f(x)$  be an  $R$ -valued function, for  $x$  ranging over the finite poset  $S$ , and let  $g(x) = \sum_y f(y)\zeta(y, x)$ . Then  $f(x) = \sum_y g(y)\mu^{(R)}(y, x)$ .

If  $\mu$  denotes the real-valued Möbius function, then because of formula 6  $\mu$  takes values only in  $\mathbb{Z}$ . Consequently, if  $R_0 \subseteq R$  is the prime ring of  $R$ , which equals either  $\mathbb{Z}$  or  $\mathbb{Z}/m\mathbb{Z}$ , for some  $m \in \mathbb{Z}$ , then

$$\mu^{(R)}(x, y) = \begin{cases} \mu(x, y) & \text{if } R_0 = \mathbb{Z}; \\ \mu(x, y) \bmod m & \text{if } R_0 = \mathbb{Z}/m\mathbb{Z}. \end{cases}$$

Now, of course, we can drop the notation  $\mu^{(R)}$ .

Again from formula 6 it follows that  $\mu(x, y)$  only depends on the structure of the interval. Moreover, we know, that if  $\mu^*$  is the Möbius function of the dual poset  $S^*$ , then  $\mu^*(x, y) = \mu(y, x)$ .

Let us assume from now on that the poset  $S$  is a lattice. In line with [12] we shall consider the meet problem  $\text{MEET}_S : S \times S \rightarrow \{0, 1\}$  of the finite lattice  $S$ , defined by  $\text{MEET}_S(x, y) = \delta(0, x \wedge y)$ . We proceed as follows. Let  $M$  be a 0-1 matrix. Check whether there are two equal rows or columns in  $M$  and if this is the case, then delete one of them. Do that as long as possible. The resulting matrix  $\tilde{M}$  is called the *core* of  $M$ . Clearly, the communication complexity of the underlying problems is the same. Now it is not difficult to see that the core of  $M^{\text{UCONN}_{n(n-1)}}$  equals the core of  $M^{\text{MEET}_{\mathcal{P}(n)^*}}$ , where  $\mathcal{P}(n)^*$  is the lattice dual to the lattice of partitions of an  $n$ -set.

**Lemma 7** *Let  $M$  be the communication matrix of the meet problem assigned to the finite lattice  $S$ , and let  $p$  be a prime number. Then  $\text{rank}_{\mathbb{Z}/p\mathbb{Z}}(M) = \#\{x \in S \mid \mu(0, x) \not\equiv 0 \pmod{p}\}$ .*

**Proof.** Let  $\tilde{M}$  be the diagonal matrix  $\text{diag}(\mu(0, x))_{x \in S}$ , and let  $\zeta = (\zeta(x, y))_{x, y \in S}$  be the matrix associated with the zeta function. Wilf observed in [19], that  $\zeta^T \cdot \tilde{M} \cdot \zeta = M$ . The claim follows from the Möbius Inversion Formula.  $\square$

Now let us compute  $\#\{x \in S \mid \mu(0, x) \not\equiv 0 \pmod{p}\}$  in a special case.

**Lemma 8** *Let  $\mathcal{P}(n)^*$  be the lattice dual to the lattice  $\mathcal{P}(n)$  of partitions, let  $p < n$  be a prime number, and let  $\mu^*$  be the Möbius function of  $\mathcal{P}(n)^*$ . Then  $\#\{x \in \mathcal{P}(n)^* \mid \mu^*(0, x) \not\equiv 0 \pmod{p}\} \leq p^n$ .*

**Proof:** The following three facts are well-known.

**Fact 1.** If  $x \in \mathcal{P}(n)$ , and if  $b(x)$  is the number of blocks of the partition  $x$ , then  $[x, 1] \cong \mathcal{P}(b(x))$ .

**Fact 2.** If  $\mu$  is the Möbius function of  $\mathcal{P}(n)$ , then  $\mu^*(0, 1) = \mu(0, 1) = (-1)^{n-1}(n-1)!$ .

**Fact 3.** Let  $S(n, k)$  denote the number of partitions of an  $n$ -set into exactly  $k$  blocks (Stirling numbers of the second kind), then

$$\sum_{k=0}^n S(n, k)[X]_k = X^n,$$

where  $X$  is an indeterminant and  $[X]_k = X \cdot (X-1) \cdot \dots \cdot (X-k+1)$  is the falling factorial.

The next equality follows from Fact 1 and from Fact 2. The next but one from Fact 3.

$$\begin{aligned} \sum_{k=0}^p S(n, k) &= \#\{x \in \mathcal{P}(n)^* \mid \mu^*(0, x) \not\equiv 0 \pmod{p}\} \\ \sum_{k=0}^p S(n, k) &\leq \sum_{k=0}^p S(n, k)[p]_k = p^n \end{aligned}$$

$\square$

**Proposition 2** *Let  $m$  be arbitrary. Then  $\text{MOD}_m\text{-Comm}(\text{UCONN}_{n(n-1)}) = O(n)$ .*

**Proof.** Let  $p$  be a prime number such that  $p \mid m$ . By Lemma 1 we have

$$\text{MOD}_m\text{-Comm}(\text{UCONN}) \leq \frac{m}{p} \cdot \text{MOD}_p\text{-Comm}(\text{UCONN}).$$

The claim follows from Corollary 1, Lemma 7, and Lemma 8.  $\square$

## 4 Variation ranks and lower bounds on the length of $\text{MOD}_m$ -protocols for undirected graph connectivity

The following lemma improves the corresponding one from [11].

**Lemma 9** *Let  $I_N$  denote the identity  $N \times N$ -matrix. Let  $m = p_1^{l_1} \cdot \dots \cdot p_r^{l_r}$  be a natural number which is given by its primary decomposition. Then  $\text{var-rank}_{\mathbb{Z}/m\mathbb{Z}}(I_N) = \lceil N/r \rceil$ .*

**Proof.** First we prove that  $\lceil N/r \rceil$  is a lower bound. Let  $A$  be an integer matrix such that  $A$  is  $\text{mod}_m$ -equivalent to  $I_N$  and  $\text{var-rank}_{\mathbb{Z}/m\mathbb{Z}}(I_N) = \text{rank}_{\mathbb{Z}} A$ , which exists by Lemma 4. By definition we have, for all  $i$ ,  $a_{ii} \not\equiv 0 \pmod{m}$ , and  $a_{ij} \equiv 0 \pmod{m}$ , for all  $j \neq i$ . For all  $i \in \{1, \dots, N\}$  there is a  $k \in \{1, \dots, r\}$  such that  $a_{ii} \not\equiv 0 \pmod{p^k}$ . We conclude that there is a primary component  $p_k^{l_k}$  of  $m$ , which we denote for simplicity by  $p^l$ , a set of indices

$$\mathcal{I} \subseteq \{1, 2, \dots, N\}, \#\mathcal{I} \geq N' := \lceil N/r \rceil,$$

and, for all  $i \in \mathcal{I}$ , natural numbers  $\nu_i \in \{1, \dots, l_i\}$ , such that

$$\begin{aligned} a_{ii} &\equiv 0 \pmod{p^{l-\nu_i}}, \\ a_{ii} &\not\equiv 0 \pmod{p^{l-\nu_i+1}}, \\ a_{ij} &\equiv 0 \pmod{p^l}, \end{aligned}$$

for all  $j \in \mathcal{I}$ ,  $j \neq i$ . After deleting all rows and columns of  $A$  whose indices do not belong to  $\mathcal{I}$ , we get an integer  $N' \times N'$ -matrix  $B$ . It is sufficient to show that  $\det B \neq 0$ . It is easy to see that

$$\begin{aligned} b_{1,1} \cdot \dots \cdot b_{N',N'} &\not\equiv 0 \pmod{p^{N' \cdot l + 1 - \sum_{i=1}^{N'} \nu_i}}, \text{ but} \\ b_{1,\sigma(1)} \cdot \dots \cdot b_{N',\sigma(N')} &\equiv 0 \pmod{p^{N' \cdot l + 1 - \sum_{i=1}^{N'} \nu_i}}, \end{aligned}$$

for all permutations  $\sigma$  of the set  $\{1, \dots, N'\}$  different from the identity permutation. Consequently,

$$\det B \equiv b_{1,1} \cdot \dots \cdot b_{N',N'} \not\equiv 0 \pmod{p^{N' \cdot l + 1 - \sum_{i=1}^{N'} \nu_i}}.$$

Second let us prove that  $\lceil N/r \rceil$  is an upper bound. Let  $f_i = p_i^{-l_i} \prod_{\mu=1}^r p_\mu^{l_\mu}$ ,  $F_j = (f_1, \dots, f_j)$ , and  $A_j = F_j^T \cdot F_j$  for  $i, j = 1 \in \{1, \dots, r\}$ .  $A_0$  is defined to be the unique  $0 \times 0$ -matrix, which, of course, has rank 0. Clearly,  $A_j \text{ mod } m$  is a  $j \times j$ -diagonal matrix of  $\mathbb{Z}/m\mathbb{Z}$ -rank 1, for  $j \in \{1, \dots, r\}$ . Define the matrix  $A$  to be the following direct sum of matrices.

$$A \stackrel{\text{def}}{=} A_r \oplus \dots \oplus A_{\lceil N/r \rceil} \oplus A_r \oplus A_{r'},$$

where  $r' \equiv N \pmod{r}$ , and  $r' \in \{0, \dots, r-1\}$ . It follows that  $A \text{ mod } m$  is a diagonal  $N \times N$ -matrix, and that  $\text{rank}_{\mathbb{Z}/m\mathbb{Z}}(A \text{ mod } m) \leq \lceil N/r \rceil$ .  $\square$

**Proposition 3** *For  $m$  arbitrary, we have that  $\text{MOD}_m\text{-Comm}(\text{SEQ}_{2n}) = \Theta(n)$ .*

**Proof.** The claim follows from Proposition 1 and from Lemma 9.  $\square$

**Lemma 10**  $SEQ = (SEQ_{2n})_{n \in \mathbf{N}}$  is reducible to  $UConn = (UConn_{n(n-1)})_{n \in \mathbf{N}}$  given in distributed form via a  $O(n^2)$ -projection reduction with respect to the partition of the variables.

**Proof.** Consider an input  $(t_1, \dots, t_n, u_1, \dots, u_n)$  of  $SEQ_{2n}$ . The projection reduction

$$\pi_{n(n-1)} : \{x_{ij}, y_{ij} \mid i, j = 1, \dots, n, i < j\} \rightarrow \{0, 1, t_\nu, u_\nu, \neg t_\nu, \neg u_\nu \mid \nu = 1, \dots, n\},$$

where the values of the Boolean variables  $x_{ij}$  and  $y_{ij}$  define the graphs  $G_1$  and  $G_2$  accessible to the processors  $\mathcal{P}_1$  and  $\mathcal{P}_2$ , is defined by the help of Figure 1 and Figure 2, in which the transpose

$$\pi_{n(n-1)}^t(t_1, \dots, t_n, u_1, \dots, u_n)$$

is shown. Clearly, this graph is connected, if and only if,

$$SEQ_{2n}(t_1, \dots, t_n, u_1, \dots, u_n) = 1.$$

□

Now it is easy to prove the lower bound.

**Proposition 4** Let  $m$  be arbitrary. Then  $MOD_m\text{-Comm}(UConn_{n(n-1)}) = \Omega(n)$ .

**Proof.** The claim follows from Lemma 10, Lemma 3 and Proposition 3. □

## References

- [1] A. V. Aho, J. D. Ullman, M. Yannakakis, *On notions of information transfer in VLSI circuits*, in: Proc. 15th ACM STOC 1983, pp. 133–183.
- [2] N. Alon, W. Maass, *Meanders, Ramsey theory and lower bounds for branching programs*, Journal of Computer and System Sciences **37**(1988), pp. 118–129.
- [3] L. Babai, P. Frankl, J. Simon, *Complexity classes in communication complexity theory*, in: Proc. 27th IEEE FOCS, pp. 337–347, 1986.
- [4] L. Babai, N. Nisan, M. Szegedy, *Multiparty protocols and logspace-hard pseudorandom sequences*, Journal of Computer and System Sciences **45**(1992), pp. 204–232.
- [5] B. Halstenberg, R. Reischuk, *Relations between Communication Complexity Classes*, Journal of Computer and System Sciences **41**(1990), pp. 402–429.
- [6] B. Halstenberg, *The Polynomial Communication Hierarchy and Protocols with Moderately Bounded Error*, Technical Report TI 1/90 of TH Darmstadt, 1990.
- [7] J. Hastad, M. Goldmann, *On the power of small-depth threshold circuits*, in: Proc. 31st IEEE FOCS 1990, pp. 610–618.

- [8] C. Damm, M. Krause, Ch. Meinel, St. Waack, *Separating counting communication complexity classes*, in: Proc. 9th STACS, Lecture Notes in Computer Science 577, Springer Verlag 1992, pp. 281–293.
- [9] A. Hajnal, W. Maass, G. Turan, *On the communication complexity of graph problems* in: Proc. 20th ACM STOC 1988, pp. 186–191.
- [10] J. Hromkovic, M. Krause, Ch. Meinel, St. Waack, *Branching programs provide lower bounds on the area of multilevel deterministic and nondeterministic VLSI circuits.*, Information and Computation 94(2)(1992) pp. 168–178.
- [11] M. Krause, St. Waack, *Variation ranks of communication matrices and lower bounds for depth two circuits having symmetric gates with unbounded fan-in*, in: Proc. 32th IEEE FOCS 1991, pp. 777–782.
- [12] L. Lovasz, *Communication complexity: A survey*, in: *Paths, flows and VLSI-layouts*, Springer-Verlag 1990, pp. 235–266.
- [13] L. Lovasz, M. Saks, *Communication complexity and combinatorial lattice theory*, Journal of Computer and System Sciences 47(1993), pp. 322–349.
- [14] Ch. Meinel, St. Waack, *Upper and lower bounds for certain graph-accessibility problems on bounded alternating  $\omega$ -branching programs*, in: *Complexity Theory – current research*, K. Ambros–Spies, St. Homer, U. Schöning (editors), Cambridge University Press 1993, 273–290.
- [15] R. Raz, B. Spieker, *On the “log rank”-conjecture in communication complexity*, in: Proc. 34th IEEE FOCS 1993, pp. 168–176.
- [16] G.-C. Rota, *On the foundation of combinatorial theory: I. Theory of Möbius functions*, Z. Wahrscheinlichkeitstheorie 2(1964), pp. 340–368.
- [17] L. Skyum, L. V. Valiant, *A complexity theory based on Boolean algebra*, in: Proc. 22th IEEE FOCS, pp. 244–253.
- [18] A. Wigderson, *The complexity of graph connectivity*, TR 92-19, Leibniz Center for Research in Computer Science, Institute of Computer Science, Hebrew University, Jerusalem.
- [19] S. Wilf, *Hadamard determinants, Möbius functions and the chromatic number of graphs*, Bull./ Amer. Math. Soc. 74(1968), pp. 960–964.
- [20] A. C.-C. Yao, *On ACC and threshold circuits*, in: Proc. 31st IEEE FOCS 1990, pp. 619–627.

Figure 1: The graph  $\pi_{n(n-1)}^t(t_1, \dots, t_n, u_1, \dots, u_n)$   
 ( $K_{2,2}$  denotes full bipartite graph having  $2 \times 2$  nodes,  $\mathcal{G}(t_\mu, u_\mu)$  is defined in Figure 2.)

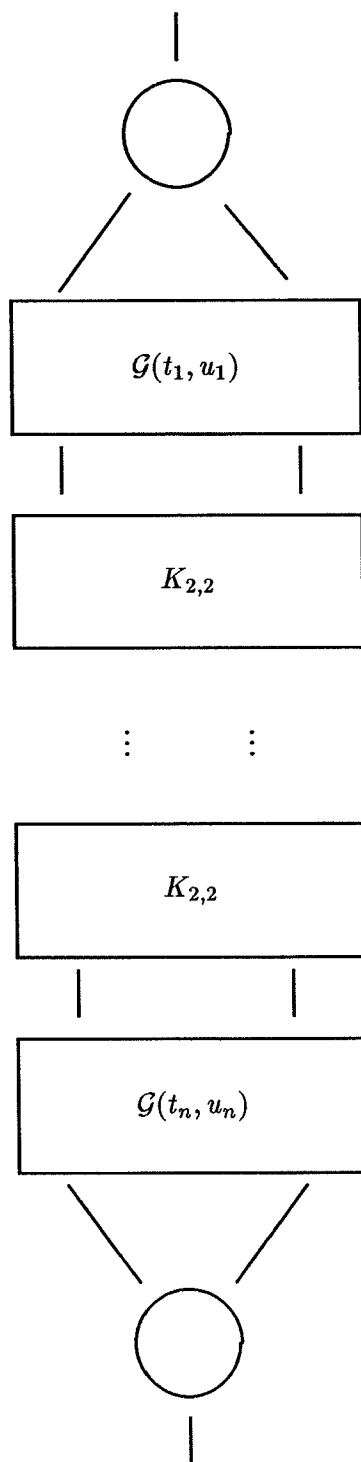
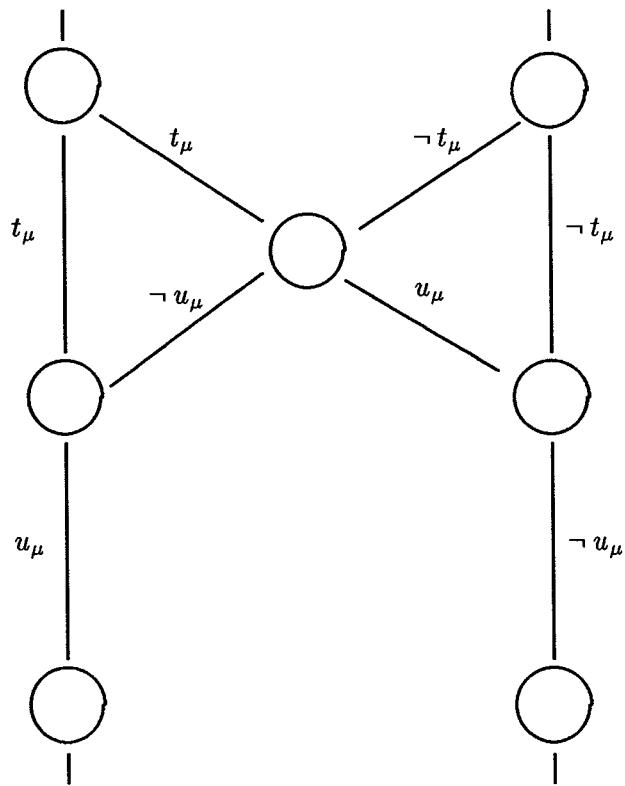


Figure 2: The graphs  $\mathcal{G}(t_\mu, u_\mu)$  of Figure 1

**Lecturer: Søren Riis**

**Title: Complexity of Counting Principles**

**Material: Søren Riis: *Count( $q$ ) versus the Pigeon-Hole Principle***

BRICS  
Computer Science Department  
University of Aarhus  
DK-8000 Aarhus C, Denmark  
E-mail: smriis@daimi.aau.dk



# Count( $q$ ) versus the Pigeon-Hole Principle

Søren Riis  
BRICS\*

June 1994

## Abstract

For each  $q \leq 2$  there exist a model  $\mathbb{M}$  of  $I\Delta_0(\alpha)$  which satisfies the Count( $q$ ) principle. Furthermore there exist  $n, r \in \mathbb{M}$  and a bijective map  $f \in \text{Set}(\mathbb{M})$  mapping  $\{1, 2, \dots, n\}$  onto  $\{1, 2, \dots, n + q^r\}$ .

A corollary is a complete classification of the Count( $q$ ) versus Count( $p$ ) problem. Another corollary solves an open question ([3]).

In this note I state and prove a Theorem which actually can be viewed as the main result of [10].

**Theorem:** *Let  $q \geq 2$ . Suppose that  $r(n)$  is an function with*

- (a)  $\lim_{n \rightarrow \infty} r(n) = \infty$ .
- (b) For all  $\epsilon > 0$   $\lim_{n \rightarrow \infty} \frac{q^{r(n)}}{n^\epsilon} = 0$

*Suppose that  $\mathcal{F}$  is any system of Bounded Arithmetic over some countable language  $L$ . Suppose that  $L$  in addition to the language of arithmetic also contains at least one undefined relation symbol. Suppose that all terms  $t$  in  $L$  have polynomial growth rate. Then there exists a model  $\mathbb{M}$  of  $\mathcal{F}$  such that:*

- (i)  $\mathbb{M} \models \text{Count}(q)$ .
- (ii) The  $\text{PHP}_{*+q^{r(*)}}^*(\text{bij})$ -principle fails in  $\mathbb{M}$ .

Here  $\text{PHP}_{*+s}^*(\text{bij})$  is the elementary principle stating that *there does not exist  $n$  and a bijective map from  $\{1, 2, \dots, n\}$  onto  $\{1, 2, \dots, n + s\}$* . And

---

\*Basic Research in Computer Science, Centre of the Danish National Research Foundation.

$\text{Count}(p)$  is the elementary matching principle stating that *if*  $\{1, 2, \dots, n\}$  *is divided into disjoint*  $p$ -*element subsets, then*  $p$  *divides*  $n$ . The principle is expressed as a  $\Delta_0$ -axiom scheme.

**Proof:** As in [10] let  $\mathbb{M}$  be a countable non-standard model of first order Arithmetic. Then by a similar forcing construction (which actually avoids certain technical problems) we expand  $\mathbb{M}$  by a generic bijection  $f$  mapping  $\{1, 2, \dots, n\}$  onto  $\{1, 2, \dots, n + q^{r(n)}\}$ . Assumption (a) allows us to assume that  $q^{r(n)}$  is a non-standard number. Furthermore condition (b) ensures that the circuit collapsing argument goes through. Now it follows by the analysis in [10] that the  $\text{Count}(p)$  principle can never be forced false. If it was false, there would exist an impossible  $\mathbb{M}$ -definable object. In this case a forest of  $(D, R)$ -labelled trees where  $|R| - |D| = q^{r(n)}$ , but where all trees would have height dominated by some standard number. This violates the main lemma (lemma 6.1.5) in [10]. Finally  $\mathbb{M}^*$  is got a the initial segment  $\{m \in \mathbb{M} : n^k > m, k \in \mathbb{N}\}$ .  $\square$

**Corollary 1: (Settling conjecture by Ajtai [3], [5], [10])**

*For different primes*  $q, p$   $\text{Count}(q) \not\vdash \text{Count}(p)$

**Corollary 2: (Obtaining the complete classification [4], [10])**

*For fixed*  $q, p \geq 2$  *the following is equivalent*

- (a)  $p$  *divides a power of*  $q$
- (b)  $\text{Count}(q) \vdash \text{Count}(p)$ .

**Proof:** The implication (a)  $\Rightarrow$  (b) was shown in [4] or [10]. The implication (b)  $\Rightarrow$  (a) follows from the Theorem. According to the Theorem  $\text{Count}(p) \not\vdash \text{PHP}_{*+q^{r(*)}}^*(\text{bij})$  if  $\text{Count}(q) \vdash \text{Count}(p)$ . But then by the easy ‘only if’ in corollary 1,  $p$  must divide a power of  $q$ .  $\square$

**Corollary 3: (Solving the Count versus PHP problem)** *Let*  $r(n)$  *be as above. For each*  $q, p \geq 2$

$\text{Count}(p) \not\vdash \text{PHP}_{*+q^{r(*)}}^*(\text{bij})$  *if and only if*  $p$  *divides a power of*  $q$ .

Let  $\text{PHP}_{*+p}^{*+p}(\text{inj})$  be the the statement that *there is no*  $n$  *and no injective map from*  $\{1, 2, \dots, n + p\}$  *into*  $\{1, 2, \dots, n\}$  and let  $\text{PHP}_{*+p}^*(\text{sur})$  be the statement that *there is no*  $n$  *and no surjective map from*  $\{1, 2, \dots, n\}$  *onto*  $\{1, 2, \dots, n + p\}$ .

**Corollary 4: (Answering an open question in [3])**

- (a)  $\text{PHP}_{*+1}^*(\text{bij}) \not\vdash \text{PHP}_{*+1}^{*+1}(\text{inj})$ .
- (b)  $\text{PHP}_{*+1}^{*+1}(\text{inj}) \not\vdash \text{PHP}_{*+1}^*(\text{sur})$ .

(c)  $\text{Count}(q) \not\vdash \text{PHP}_*^{*+1}(\text{inj})$ .

**Proof:** (b) is a simple exercise, and (a) clearly follows from (c). To show (c) notice that  $\text{PHP}_*^{*+1}(\text{inj}) \vdash \text{PHP}_{*+q^r(*)}^*(\text{bij})$  for any  $r$ .  $\square$

This shows that the Pigeon-hole Principle for injective maps are efficiently stronger than the Pigeon-hole Principle for bijective maps. Actually it shows that:

**Corollary 5:** *There exists a model  $\mathbb{M}^*$  of  $I\Delta_0(\alpha)$  in which  $\text{Count}(p)$  holds for each  $p \in \mathbb{N} \setminus \{1\}$ . Yet, there exists  $n \in \mathbb{M}^*$  and an injective map  $f \in \text{Set}(\mathbb{M}^*)$  mapping  $\{1, 2, \dots, n+1\}$  into  $\{1, 2, \dots, n\}$ .*

**Proof:** By the completeness theorem it suffice to show that for each finite set  $p_1, p_2, \dots, p_l$  of integers, the conjunction  $\text{Count}(p_1) \wedge \dots \wedge \text{Count}(p_l)$  does not imply  $\text{PHP}_*^{*+1}(\text{inj})$ . This follows by an argument similar to the one given for (c) in corollary 4.  $\square$

## References

- [1] M.Ajtai; On the complexity of the pigeonhole principle. 29<sup>th</sup> Annual symp. on Found. Comp.Sci.(1988),pp 340-355.
- [2] M.Ajtai; Parity and the pigeon-hole principle, in Feasible Mathematics Birkhauser, (1990), pp 1-24.
- [3] M.Ajtai; The independence of the modulo  $p$  counting principles, Proceedings 9<sup>th</sup>-annual IEEE symposium on computer science (1994).
- [4] P.Beame, R. Impagliazzo, J. Krajicek, T. Pitassi, P. Pudlak; Lower bounds on Hilbert's Nullstellensatz and propositional proofs, preliminary version.
- [5] P.Clote, J.Krajicek; Open problems, in: Arithmetic, Proof theory and computorial complexity, Oxford university press (1993) pp 1-19.
- [6] J.Krajicek, P.Pudlak, and A.Wood, Exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle, submitted (1991).

- [7] T.Pitassi, P.Beame, and R.Impagliazzo; Exponential lower bounds for the pigeonhole principle, preprint (1991).
- [8] T.Pitassi, P.Beame; An Exponential separation between the Matching Principle and the pigeonhole principle. Proceedings 8<sup>th</sup>-annual IEEE symposium on computer science (1993), pp 308-319
- [9] S.M.Riis; Independence in Bounded Arithmetic; DPhil dissertation, Oxford University (1993)
- [10] S.M.Riis; Count( $q$ ) does not imply Count( $p$ ); Submitted. Report Series, BRICS RS-94-21.



**Lecturer: Mark Jerrum**

**Title: Approximation via Semidefinite Programming Relaxations**

**Material:** Alan Frieze and Mark Jerrum: *Improved approximation algorithms for MAX k-CUT and MAX BISECTION*

Department of Computer Science  
University of Edinburgh  
JCMB, The King's Buildings  
Edinburgh EH9 3JZ, Scotland  
E-mail: mrj@dcs.edinburgh.ac.uk

# Improved approximation algorithms for MAX $k$ -CUT and MAX BISECTION

Alan Frieze\*  
Carnegie Mellon University

Mark Jerrum†  
University of Edinburgh

6th June, 1994

## Abstract

Polynomial-time approximation algorithms with non-trivial performance guarantees are presented for the problems of (a) partitioning the vertices of a weighted graph into  $k$  blocks so as to maximise the weight of crossing edges, and (b) partitioning the vertices of a weighted graph into two blocks of equal cardinality, again so as to maximise the weight of crossing edges. The approach, pioneered by Goemans and Williamson, is via a semidefinite relaxation.

## 1 Introduction

Goemans and Williamson [5] have significantly advanced the theory of approximation algorithms. Previous work on approximation algorithms was largely dependent on comparing heuristic solution values to that of a Linear Program (LP) relaxation, either implicitly or explicitly. This was recognised some time ago by Wolsey [11]. (One significant exception to this general rule has been the case of Bin Packing.)

The main novelty of [5] is that it uses a Semi-Definite Program (SDP) as a relaxation. To be more precise let us consider the problem MAX-CUT studied (among

---

\*Department of Mathematics, Carnegie Mellon University, Pittsburgh PA15213, U.S.A., Supported by NSF grant CCR-9225008.

†Department of Computer Science, University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ, UK. The second author is a Nuffield Science Research Fellow, and is supported in part by grant GR/F 90363 of the UK Science and Engineering Research Council, and Esprit Working Group "RAND;" this work was done while visiting Carnegie Mellon University.

others) in [5]: we are given a vertex set  $V = \{1, \dots, n\}$  and non-negative weights  $w_{i,j}$ ,  $1 \leq i, j \leq n$ , where  $w_{i,j} = w_{j,i}$  and  $w_{i,i} = 0$  for all  $i, j$ . If  $S \subseteq V$  and  $\bar{S} = V \setminus S$  then the *weight* of the cut  $(S : \bar{S})$  is

$$w(S : \bar{S}) = \sum_{i \in S, j \in \bar{S}} w_{i,j}.$$

The aim is to find a cut of maximum weight.

Introducing integer variables  $y_j \in \{-1, 1\}$  for  $j \in V$  we can formulate the MAX CUT problem as

$$\begin{aligned} \text{IP: maximise } & \frac{1}{2} \sum_{i < j} w_{i,j} (1 - y_i y_j) \\ \text{subject to } & y_j \in \{-1, 1\}, \quad \forall j \in V \end{aligned} \quad (1)$$

The key insight of Goemans and Williamson is that instead of converting this to an integer linear program and then considering the LP relaxation, it is possible to relax IP directly to the following

$$\begin{aligned} \text{SDP: maximise } & \frac{1}{2} \sum_{i < j} w_{i,j} (1 - v_i \cdot v_j) \\ \text{subject to } & v_j \in S_n, \quad \forall j \in V \end{aligned}$$

Here  $S_n = \{x \in \mathbf{R}^n : \|x\| = 1\}$  is the unit sphere in  $n$  dimensions. SDP's are a special class of convex program (see Alizadeh [1] for a detailed exposition). In particular the above problem can be replaced by

$$\begin{aligned} \text{CP: maximise } & \frac{1}{2} \sum_{i < j} w_{i,j} (1 - Y_{i,j}) \\ \text{subject to } & Y_{j,j} = 1, \quad \forall j \in V \\ & Y = [Y_{i,j}] \succ 0 \end{aligned} \quad (2)$$

Here  $Y_{i,j}$  replaces  $v_i \cdot v_j$ , and the notation  $Y \succ 0$  indicates that the matrix  $Y$  is constrained to be positive semi-definite; this constraint defines a convex subset of  $\mathbf{R}^{n^2}$ . The idea of Goemans and Williamson is to solve SDP and then use the following simple (randomised rounding) heuristic to obtain a remarkably good solution to MAX-CUT: choose a random hyperplane through the origin, and partition the vectors  $v_i$  (and hence the vertex set  $V$ ) according to which side of the hyperplane they fall.

This is an exciting new idea and it is important to see in what directions it can be generalised. In this paper we do so in two ways. First we consider MAX  $k$ -CUT where the aim is to partition  $V$  into  $k$  subsets: for a partition  $\mathcal{P} = P_1, P_2, \dots, P_\ell$  of  $V$  we let  $|\mathcal{P}| = \ell$  and

$$w(\mathcal{P}) = \sum_{1 \leq r < s \leq \ell} \sum_{i \in P_r, j \in P_s} w_{i,j}.$$

The problem is then

$$\begin{aligned} \text{MAX } k\text{-CUT: maximise } & w(\mathcal{P}) \\ \text{subject to } & |\mathcal{P}| = k. \end{aligned}$$



Note that MAX  $k$ -CUT has an important interpretation as the search for a ground state in the anti-ferromagnetic  $k$ -state Potts model: see Welsh [10]. To attack this problem we need to be able to handle variables which can take on one of  $k$  values as opposed to just two, a similar problem to that faced in trying to colour 3-colourable graphs [6]. Our solution is a natural extension of the existing solution for the case  $k = 2$ , but the performance analysis presents greater technical difficulties.

The simplest heuristic for MAX  $k$ -CUT is just to randomly partition  $V$  into  $k$  sets. If  $\hat{\mathcal{P}}$  denotes the (random) partition produced and  $\mathcal{P}^*$  denotes the optimum partition then it is easy to see that

$$\mathbf{E}(w(\hat{\mathcal{P}})) \geq \left(1 - \frac{1}{k}\right) w(\mathcal{P}^*),$$

since each edge  $(i, j)$  has probability  $(1 - k^{-1})$  of joining vertices in different sets of the partition.

We describe a (randomised) heuristic  $k$ -CUT which produces a partition  $\mathcal{P}_k$ . We prove the existence of a sequence of constants  $\alpha_k, k \geq 2$  such that if  $\mathcal{P}_k^*$  denotes the optimal partition in MAX  $k$ -CUT then:

**Theorem 1**

$$\mathbf{E}(w(\mathcal{P}_k)) \geq \alpha_k w(\mathcal{P}_k^*),$$

where the  $\alpha_k$  satisfy

- (i)  $\alpha_k > 1 - k^{-1}$ ;
- (ii)  $\alpha_k - (1 - k^{-1}) \sim 2k^{-2} \ln k$ ;
- (iii)  $\alpha_2 \geq 0.878567$ ,  $\alpha_3 \geq 0.800217$ ,  $\alpha_4 \geq 0.850304$ ,  $\alpha_5 \geq 0.874243$ ,  $\alpha_{10} \geq 0.926642$ , and  $\alpha_{100} \geq 0.990625$ .

The performance ratio for  $k = 2$  is the same as that achieved by Goemans and Williamson, as our heuristic is a generalisation of theirs.

Our next result concerns the problem MAX BISECTION. Here we have to partition  $V$  into two subsets of equal size (assuming that  $n$  is even) so as to maximise  $w$ .

$$\begin{aligned} \text{MAX BISECTION: } & \text{maximise } w(\mathcal{P}) \\ & \text{subject to } \mathcal{P} = S, V \setminus S \\ & |S| = n/2. \end{aligned}$$

A random bisection produces an expected guarantee of  $\frac{1}{2}$ . We describe a heuristic BISECT which produces a partition  $\mathcal{P}_B$  such that if  $\mathcal{P}_B^*$  denotes the optimal bisection,

**Theorem 2** *Let  $\epsilon$  be a small positive constant. Then  $\mathbf{E}(w(\mathcal{P}_B)) \geq \beta w(\mathcal{P}_B^*)$  where  $\beta = 2(\sqrt{2(1-\epsilon)\alpha_2} - 1)$ , which is greater than 0.65 for  $\epsilon$  sufficiently small.*

Note that  $\alpha_2 = 0.878567\dots$ , as in Theorem 1. The difficulty with generalising Goemans and Williamson's heuristic to MAX BISECTION is that their heuristic does not generally give a bisection of  $V$ . We prove that a simple modification of their basic algorithm beats the trivial  $\frac{1}{2}$  lower bound.

Note that there is a natural generalisation of this problem MAX  $k$ -SECTION where we seek to partition  $V$  into  $k$  equal pieces. Unfortunately we cannot prove that the natural generalisation of our bisection heuristic beats the  $1 - k^{-1}$  lower bound of the simple random selection heuristic when  $k \geq 3$ .

## 2 MAX $k$ -CUT

In this section we describe our heuristic  $k$ -CUT. We first describe a suitable way of modelling variables which can take one of  $k$  values. Just allowing  $y_j = 1, 2, \dots, k$  does not easily yield a useful integer program. Instead we allow  $y_j$  to be one of  $k$  vectors  $a_1, a_2, \dots, a_k$  defined as follows: take an equilateral simplex  $\Sigma_k$  in  $\mathbf{R}^{k-1}$  with vertices  $b_1, b_2, \dots, b_k$ . Let  $c_k = (b_1 + b_2 + \dots + b_k)/k$  be the centroid of  $\Sigma_k$  and let  $a_i = b_i - c_k$ , for  $1 \leq i \leq k$ . Finally assume that  $\Sigma_k$  is scaled so that  $|a_i| = 1$  for  $1 \leq i \leq k$ .

**Lemma 1**

$$a_i \cdot a_j = -1/(k-1), \quad \text{for } i \neq j. \quad (3)$$

**Proof** Since  $a_1, a_2, \dots, a_k$  are of unit length we have to show that the angle between  $a_i$  and  $a_j$  is  $\arccos(-1/(k-1))$  for  $i \neq j$ . Let  $b_1, b_2, \dots, b_{k-1}$  lie in the plane  $x_{k-1} = 0$  and form an equilateral simplex of dimension  $k-2$ . Let  $b_i = (b'_i, 0)$  for  $1 \leq i \leq k-1$ , where  $b'_i$  has dimension  $k-2$ , and assume  $b'_1 + b'_2 + \dots + b'_{k-1} = 0$ . Then  $c_k = (0, 0, \dots, 0, x)$  and  $b_k = (0, 0, \dots, 0, kx)$  for some  $x > 0$ . But  $|b_k - c_k| = 1$  and so  $x = 1/(k-1)$ . But then  $(b_k - c_k) \cdot (b_1 - c_k) = -(k-1)x^2 = -1/(k-1)$ .  $\square$

Note that  $-1/(k-1)$  is the best angle separation we can obtain for  $k$  vectors as we see from:

**Lemma 2** *If  $u_1, u_2, \dots, u_k$  satisfy  $|u_i| = 1$  for  $1 \leq i \leq k$ , and  $u_i \cdot u_j \leq \gamma$  for  $i \neq j$ , then  $\gamma \geq -1/(k-1)$ .*

**Proof**

$$\begin{aligned} 0 &\leq (u_1 + u_2 + \dots + u_k)^2 \\ &\leq k + k(k-1)\gamma. \end{aligned}$$

□

Given Lemma 1 we can formulate MAX  $k$ -CUT as follows:

$$\begin{aligned} \text{IP}_k: \quad & \text{maximise} \quad \frac{k-1}{k} \sum_{i < j} w_{i,j} (1 - y_i \cdot y_j) \\ & \text{subject to} \quad y_j \in \{a_1, a_2, \dots, a_k\}. \end{aligned}$$

Here we use the fact that

$$1 - y_i \cdot y_j = \begin{cases} 0, & \text{if } y_i = y_j \\ \frac{k}{k-1}, & \text{if } y_i \neq y_j \end{cases}$$

To obtain our SDP relaxation we replace  $y_i$  by  $v_i$ , where  $v_i$  can now be any vector in  $S_n$ . There is a problem in that we can have  $v_i \cdot v_j = -1$  whereas  $y_i \cdot y_j \geq -1/(k-1)$ . We need therefore to add the constraint  $v_i \cdot v_j \geq -1/(k-1)$ . We obtain

$$\begin{aligned} \text{SDP}_k: \quad & \text{maximise} \quad \frac{k-1}{k} \sum_{i < j} w_{i,j} (1 - v_i \cdot v_j) \\ & \text{subject to} \quad v_j \in S_n, \quad \forall j \\ & \quad \quad \quad v_i \cdot v_j \geq -1/(k-1), \quad \forall i \neq j \end{aligned} \quad (4)$$

Note that (4) reduces to the linear constraint  $Y_{i,j} \geq -1/(k-1)$  if we go to the convex programming form CP. We can now describe our heuristic

**$k$ -CUT:**

**Step 1** solve the problem  $\text{SDP}_k$  to obtain vectors  $v_1, v_2, \dots, v_n \in S_n$ .

**Step 2** choose  $k$  random vectors  $z_1, z_2, \dots, z_k$ .

**Step 3** partition  $V$  according to which of  $z_1, z_2, \dots, z_k$  is closest to each  $v_j$ , i.e., let  $\mathcal{P} = P_1, P_2, \dots, P_k$  be defined by

$$P_i = \{j : v_j \cdot z_i \geq v_j \cdot z_{i'} \text{ for } i' \neq i\}, \quad \text{for } 1 \leq i \leq k.$$

(Break ties for the minimum arbitrarily: they occur with probability zero!)

The most natural way of choosing  $z_1, z_2, \dots, z_k$  is to choose them independently at random from  $S_n$ . Forcing  $|z_i| = 1$  complicates the analysis marginally and so we let  $z_j = (z_{1,j}, z_{2,j}, \dots, z_{n,j})$ ,  $1 \leq j \leq k$  where the  $z_{i,j}$  are  $kn$  independent samples from a (standard) normal distribution with mean 0 and variance 1. When  $k = 2$  we have the heuristic of Goemans and Williamson, although they define it in terms of cutting  $S_n$  by a random hyperplane through the origin.

Let  $W_k$  denote the weight of the partition produced by the heuristic, let  $W_k^*$  be the weight of the optimal partition and let  $\widetilde{W}_k$  denote the maximum value of  $\text{SDP}_k$ . Putting  $y_j = a_i$  for  $j \in P_i$ ,  $1 \leq i \leq k$  we see that

$$\mathbf{E}(W_k) = \sum_{i < j} w_{i,j} \Pr(y_i \neq y_j). \quad (5)$$

Now by symmetry  $\Pr(y_i \neq y_j)$  depends only on the angle  $\theta$  between  $v_i$  and  $v_j$ , and hence on  $\rho = \cos \theta = v_i \cdot v_j$ . Let this separation probability be denoted by  $\Phi_k(\rho)$ . It then follows from (5) that

$$\begin{aligned} \frac{\mathbf{E}(W_k)}{W_k^*} &\geq \frac{\mathbf{E}(W_k)}{\widetilde{W}_k} \\ &= \frac{\sum_{i<j} w_{i,j} \Phi_k(v_i \cdot v_j)}{\frac{k-1}{k} \sum_{i<j} w_{i,j} (1 - v_i \cdot v_j)} \\ &\geq \alpha_k, \end{aligned}$$

where

$$\alpha_k = \min_{-1/(k-1) \leq \rho \leq 1} \frac{k \Phi_k(\rho)}{(k-1)(1-\rho)}.$$

We leave the estimation of the  $\alpha_k$  to an appendix (see Corollaries 1, 2, and 3). Suffice it to say that they satisfy the claims of Theorem 1.

### 3 MAX BISECTION

We now describe how to ensure that the partition we obtain divides  $V$  into equal parts. As an integer program we can express MAX BISECTION as

$$\begin{aligned} \text{IP}_B: \quad &\text{maximise} \quad \frac{1}{2} \sum_{i<j} w_{i,j} (1 - y_i y_j) \\ &\text{subject to} \quad \sum_{i<j} y_i y_j \leq -n/2 \\ &\quad \quad \quad y_j \in \{-1, 1\} \quad \forall j \in V \end{aligned} \quad (6)$$

Constraint (6) expresses the fact that we force  $|S| = n/2$  by maximising the number of pairs  $i, j$  where  $i \in S, j \notin S$ . It has the advantage of being easily relaxed to give an SDP problem:

$$\begin{aligned} \text{SDP}_B: \quad &\text{maximise} \quad \frac{1}{2} \sum_{i<j} w_{i,j} (1 - v_i \cdot v_j) \\ &\text{subject to} \quad \sum_{i<j} v_i \cdot v_j \leq -n/2 \\ &\quad \quad \quad v_j \in S_n, \quad \forall j \in V \end{aligned} \quad (7)$$

We can now describe our heuristic:  $\epsilon$  is a small positive constant,  $\epsilon = 1/100$  is small enough.

#### BISECT

**Step 1** solve the problem  $\text{SDP}_B$  to obtain vectors  $v_1, v_2, \dots, v_n \in S_n$ .

Repeat Steps 2-4 below for  $t = 1, 2, \dots, K = K(\epsilon) = \lceil \epsilon^{-1} \ln \epsilon^{-1} \rceil$  and output the best partition  $\tilde{S}_t, V \setminus \tilde{S}_t$  found in Step 4.

**Step 2** choose 2 *random* vectors  $z_1, z_2$ .

**Step 3** let  $S_t = \{j : v_j \cdot z_1 \leq v_j \cdot z_2\}$ .

**Step 4** suppose (w.l.o.g.) that  $|S_t| \geq n/2$ . For each  $i \in S_t$  let  $\zeta(i) = \sum_{j \notin S_t} w_{i,j}$  and let  $S_t = \{x_1, x_2, \dots, x_\ell\}$  where  $\zeta(x_1) \geq \zeta(x_2) \geq \dots \geq \zeta(x_\ell)$ . Let  $\tilde{S}_t = \{x_1, \dots, x_{n/2}\}$ .

Clearly the construction in Step 4 satisfies

$$w(\tilde{S}_t : V \setminus \tilde{S}_t) \geq \frac{n w(S_t : V \setminus S_t)}{2\ell}. \quad (8)$$

In order to analyse the quality of the final partition we define two sets of random variables.

$$\begin{aligned} X_t &= w(S_t : V \setminus S_t), & 1 \leq t \leq K. \\ Y_t &= |S_t|(n - |S_t|), & 1 \leq t \leq K. \end{aligned}$$

Recall that  $\mathcal{P}_B^*$  denotes the optimum bisection, and let  $W^* \geq w(\mathcal{P}_B^*)$  denote the maximum of  $\text{SDP}_B$ . Then, by the analysis of Theorem 1 (or [5]),

$$\mathbf{E}(X_t) \geq \alpha_2 W^*. \quad (9)$$

Also

$$\begin{aligned} \mathbf{E}(Y_t) &= \sum_{i < j} \Phi_2(v_i \cdot v_j) \\ &\geq \frac{\alpha_2}{2} \sum_{i < j} (1 - v_i \cdot v_j) \\ &\geq \alpha_2 N, \end{aligned}$$

where  $N = n^2/4$  (note the use of (7) here.)

Thus if

$$Z_t = \frac{X_t}{W^*} + \frac{Y_t}{N}$$

then

$$\mathbf{E}(Z_t) \geq 2\alpha_2. \quad (10)$$

On the other hand

$$Z_t \leq 2, \quad (11)$$

since  $X_t \leq W^*$  and  $Y_t \leq N$ .

Define  $Z_\tau = \max_{1 \leq t \leq K} \{Z_t\}$ . Now (10) and (11) imply that for any  $\epsilon > 0$

$$\Pr(Z_1 \leq 2(1 - \epsilon)\alpha_2) \leq \frac{1 - \alpha_2}{1 - (1 - \epsilon)\alpha_2}$$

and so

$$\Pr(Z_\tau \leq 2(1 - \epsilon)\alpha_2) \leq \left( \frac{1 - \alpha_2}{1 - (1 - \epsilon)\alpha_2} \right)^K \leq \epsilon,$$

for the given choice of  $K(\epsilon)$ . Assume that

$$Z_\tau \geq 2(1 - \epsilon)\alpha_2 \quad (12)$$

and suppose

$$X_\tau = \lambda W^*.$$

which from (10) and (12) implies

$$Y_\tau \geq (2(1 - \epsilon)\alpha_2 - \lambda)N. \quad (13)$$

Suppose  $|S_\tau| = \delta n$ ; then (13) implies

$$\delta(1 - \delta) \geq (2(1 - \epsilon)\alpha_2 - \lambda)/4. \quad (14)$$

Applying (8) and (14) we see that

$$\begin{aligned} w(\tilde{S}_\tau : V \setminus \tilde{S}_\tau) &\geq w(S_\tau : V \setminus S_\tau)/(2\delta) \\ &\geq \lambda W^*/(2\delta) \\ &\geq (2(1 - \epsilon)\alpha_2 - 4\delta(1 - \delta))W^*/(2\delta) \\ &\geq 2(\sqrt{2(1 - \epsilon)\alpha_2} - 1)W^*. \end{aligned}$$

The last inequality follows from simple calculus.

Thus

$$\begin{aligned} \mathbf{E}(w(\tilde{S}_\tau)) &\geq 2(\sqrt{2(1 - \epsilon)\alpha_2} - 1) \left( 1 - \left( \frac{1 - \alpha_2}{1 - (1 - \epsilon)\alpha_2} \right)^K \right) W^* \\ &\geq 2(\sqrt{2(1 - 3\epsilon)\alpha_2} - 1)W^*. \end{aligned}$$

Finally note that the partition output by BISECT is at least as good as  $\tilde{S}_\tau$ . We divide  $\epsilon$  above by 3 to get the precise result.

## 4 Appendix

Let  $u, v$  be vectors, and  $r_1, \dots, r_k$  be a sequence of vectors, all in  $\mathbf{R}^n$ . We say that  $u$  and  $v$  are *separated by*  $r_1, \dots, r_k$  if the vector  $r_i$  maximising  $u \cdot r_i$  is distinct from the vector  $r_j$  maximising  $v \cdot r_j$ . When we speak of a random vector, we mean a vector  $r = (\xi_1, \dots, \xi_n)$  whose coordinates  $\xi_i$  are independent, normally distributed random variables with mean 0 and variance 1. Note that the probability density

function of  $r$  is  $(2\pi)^{-n/2} \exp(-(\xi_1^2 + \dots + \xi_n^2)/2)$ , and in particular is spherically symmetric.

Denote by  $g(x) = (2\pi)^{-1/2} \exp(-x^2/2)$  the probability density function of the univariate normal distribution, and by  $G(x) = \int_{-\infty}^x g(\xi) d\xi$  the corresponding cumulative distribution function. For  $i = 1, 2, \dots$ , the normalised *Hermite polynomials*  $\phi_i(\cdot)$  are defined by

$$(-1)^i \sqrt{i!} \phi_i(x) g(x) = \frac{d^i g(x)}{dx^i}. \tag{15}$$

Let  $h_i = h_i(k)$  denote the expectation of  $\phi_i(x_{\max})$ , where  $x_{\max}$  is distributed as the maximum of a sequence of  $k$  independent normally distributed random variables.

**Lemma 3** *Suppose  $u, v \in \mathbf{R}^n$  are unit vectors at angle  $\theta$ , and  $r_1, \dots, r_k$  is a sequence of random vectors. Let  $\rho = \cos \theta = u \cdot v$ , and denote by  $N_k(\rho) = 1 - \Phi_k(\rho)$  the probability that  $u$  and  $v$  are not separated by  $r_1, \dots, r_k$ . Then the Taylor series expansion*

$$N_k(\rho) = a_0 + a_1 \rho + a_2 \rho^2 + a_3 \rho^3 + \dots$$

*of  $N_k(\rho)$  about the point  $\rho = 0$  converges for all  $\rho$  in the range  $|\rho| \leq 1$ . The coefficients  $a_i$  of the expansion are all non-negative, and their sum converges to  $N_k(1) = 1$ . The first three coefficients are  $a_0 = 1/k$ ,  $a_1 = h_1^2/(k-1)$  and  $a_2 = kh_2^2/(k-1)(k-2)$ .*

**Proof** We begin by computing the joint distribution of  $x = u \cdot r$  and  $y = v \cdot r$ , where  $r = (\xi_1, \dots, \xi_n)$  is a random vector. Since the density function of  $r$  is spherically symmetric, this joint distribution is dependent on  $\theta$  only, and not on the particular choice of  $u$  and  $v$ ; for convenience let  $u = (1, 0, \dots, 0)$  and  $v = (\cos \theta, \sin \theta, 0, \dots, 0)$ . Then

$$\begin{aligned} \Pr(u \cdot r \leq x \text{ and } v \cdot r \leq y) &= \Pr(\xi_1 \leq x \text{ and } \xi_1 \cos \theta + \xi_2 \sin \theta \leq y) \\ &= \frac{1}{2\pi} \int_{\xi_1=-\infty}^x \int_{\xi_2=-\infty}^{(y-\xi_1 \cos \theta)/\sin \theta} \exp\left(-\frac{\xi_1^2 + \xi_2^2}{2}\right) d\xi_2 d\xi_1 \\ &= \frac{1}{2\pi \sin \theta} \int_{\zeta_1=-\infty}^x \int_{\zeta_2=-\infty}^y \exp\left(-\frac{\zeta_1^2 - 2 \cos(\theta) \zeta_1 \zeta_2 + \zeta_2^2}{2(\sin \theta)^2}\right) d\zeta_2 d\zeta_1, \end{aligned}$$

where we have applied the change of coordinates  $\zeta_1 = \xi_1$  and  $\zeta_2 = \xi_1 \cos \theta + \xi_2 \sin \theta$ . The joint probability density function of  $x = u \cdot r$  and  $y = v \cdot r$  is thus

$$f(x, y; \rho) = \frac{1}{2\pi \sqrt{1-\rho^2}} \exp\left(-\frac{x^2 - 2\rho xy + y^2}{2(1-\rho^2)}\right),$$

where  $\rho = \cos \theta$ ; this is the probability density function of the bivariate normal distribution in standard form, with correlation  $\rho = \cos \theta$ . Denote by

$$F(x, y; \rho) = \int_{\xi=-\infty}^x \int_{\eta=-\infty}^y f(\xi, \eta; \rho) d\eta d\xi$$

the corresponding cumulative distribution function.

Let  $r_1, \dots, r_k$  be independent random vectors; then

$$\begin{aligned} & \Pr(u \text{ and } v \text{ are not separated by } r_1, \dots, r_k) \\ &= k \times \Pr(u \cdot r_1 = \max_i u \cdot r_i \text{ and } v \cdot r_1 = \max_j v \cdot r_j) \\ &= k I(\rho), \end{aligned}$$

where

$$I(\rho) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y; \rho) F(x, y; \rho)^{k-1} dx dy.$$

There is no expression for the integral  $I(\rho)$  in closed form, so we compute instead a Taylor series expansion for  $I(\rho)$  about  $\rho = 0$  using ideas (and notation) from Bofinger and Bofinger [2]. The *Mehler expansion* [9] of the bivariate normal probability density function

$$f(x, y; \rho) = g(x)g(y) \left( 1 + \rho \phi_1(x)\phi_1(y) + \rho^2 \phi_2(x)\phi_2(y) + \dots \right), \quad (16)$$

converges uniformly for  $|\rho| < 1$ . Three facts that follow easily from the Mehler expansion and definition (15) of the Hermite polynomials are:

$$\frac{d}{dx} g(x)\phi_{i-1}(x) = -\sqrt{i} g(x)\phi_i(x), \quad (17)$$

$$\frac{\partial F(x, y; \rho)}{\partial \rho} = f(x, y; \rho) \quad (18)$$

and

$$\left. \frac{\partial^i f}{\partial \rho^i} \right|_{\rho=0} = i! g(x)g(y)\phi_i(x)\phi_i(y). \quad (19)$$

We now evaluate  $I(\rho)$  and its successive derivatives with respect to  $\rho$  at the point  $\rho = 0$  by noting that  $F(x, y; 0)$  and  $f(x, y; 0)$  factorise into  $G(x)G(y)$  and  $g(x)g(y)$ , respectively. In this way we obtain a Taylor series expansion for  $I(\rho)$  about the point  $\rho = 0$ . We defer an examination of the radius of convergence of this Taylor expansion to the end of the proof.

Starting with  $I$  itself, we have

$$I(0) = \left( \int g(x)G(x)^{k-1} dx \right)^2 = \frac{1}{k^2}, \quad (20)$$

where the second equality can be seen by interpreting the integral as the probability that the maximum of a sequence of  $k$  independent, normally distributed random variables is achieved by the first variable.<sup>1</sup>

<sup>1</sup>Integration will be assumed to be over the infinite line when the limits of integration are omitted.



By identities (18) and (19),

$$\left. \frac{\partial I}{\partial \rho} \right|_{\rho=0} = \left( \int g(x) \phi_1(x) G(x)^{k-1} dx \right)^2 + (k-1) \left( \int g(x)^2 G(x)^{k-2} dx \right)^2.$$

(Passing the derivative through the integral is justified by Section 1.88 of Titchmarsh's text on analysis of functions [8].) The first integral is simply  $h_1/k$ ; the second may be simplified using integration by parts, and identity (17):

$$\begin{aligned} \int g(x)(g(x)G(x)^{k-2}) dx &= \left[ \frac{g(x)G(x)^{k-1}}{k-1} \right]_{-\infty}^{\infty} - \frac{1}{k-1} \int g'(x)G(x)^{k-1} dx \\ &= \frac{1}{k-1} \int g(x)\phi_1(x)G(x)^{k-1} dx \\ &= \frac{h_1}{k(k-1)}. \end{aligned}$$

Substituting these expressions for the two integrals yields

$$\left. \frac{\partial I}{\partial \rho} \right|_{\rho=0} = \frac{h_1^2}{k(k-1)}. \quad (21)$$

Differentiating with respect to  $\rho$  a second time, we obtain

$$\begin{aligned} \left. \frac{\partial^2 I}{\partial \rho^2} \right|_{\rho=0} &= 2 \left( \int g(x)\phi_2(x)G(x)^{k-1} dx \right)^2 \\ &\quad + 3(k-1) \left( \int g(x)^2\phi_1(x)G(x)^{k-2} dx \right)^2 \\ &\quad + (k-1)(k-2) \left( \int g(x)^3G(x)^{k-3} dx \right)^2. \end{aligned}$$

The first integral is just  $h_2/k$ . The second, using integration by parts and identity (17), is

$$\begin{aligned} \int (g(x)\phi_1(x))(g(x)G(x)^{k-2}) dx &= -\frac{1}{k-1} \int (-\sqrt{2}g(x)\phi_2(x))G(x)^{k-1} dx \\ &= \frac{\sqrt{2}h_2}{k(k-1)}. \end{aligned}$$

A further application of integration by parts reduces the third integral to the second, from which

$$\int g(x)^3G(x)^{k-3} = \frac{2\sqrt{2}h_2}{k(k-1)(k-2)}.$$

Substituting these expressions for the three integrals yields

$$\left. \frac{\partial^2 I}{\partial \rho^2} \right|_{\rho=0} = \left( \frac{2}{k^2} + \frac{6}{k^2(k-1)} + \frac{8}{k^2(k-1)(k-2)} \right) h_2^2 = \frac{2h_2^2}{(k-1)(k-2)}. \quad (22)$$

In principle the process of repeated differentiation by  $\rho$  could be continued indefinitely; for any  $i$ , the  $i$ th derivative of  $I(\rho)$  evaluated at  $\rho = 0$  is a positive linear combination of squares of one-dimensional integrals. This observation, combined with (20), (21), and (22) establishes the claims concerning the Taylor expansion of  $I(\rho)$ .

It remains to show that the Taylor expansion of  $I(\rho)$  is valid for  $|\rho| < 1$  and hence — by continuity of  $N_k(\rho)$  at  $\rho = 1$  and the fact that all terms in the expansion are positive — for  $|\rho| \leq 1$ . Observe that  $I(\rho)$  is defined by an integral of the form

$$I(\rho) = \iint \sum_{i=0}^{\infty} \rho^i s_i(x, y) dx dy, \quad (23)$$

where  $s_i(x, y) = \sum_{j=0}^{n_i-1} t_{ij}(x, y)$  is a sum of terms  $t_{ij}(x, y)$ , and each term  $t_{ij}(x, y)$  is a product of factors of the form  $g(x)g(y)\phi_l(x)\phi_l(y)$ . Now  $\iint |t_{ij}(x, y)| dx dy < 2.6$ , since  $\int |g(x)\phi_l(x)| dx < 1.6$  and  $\max_x |g(x)\phi_l(x)| < 1$  for all  $l$ . (These facts follow from the key inequality on page 324 of Sansone's treatise on orthogonal functions [7], which bounds  $|\phi_l(x)|$  by  $c \exp(-x^2/4)$  for an absolute constant  $c$ ; note, however, that the bound given by Sansone is for *un-normalised* Hermite polynomials, and must be scaled accordingly.) Noting that  $n_i = O(i^{k-1})$ , we see that the sum

$$\sum_{i=0}^{\infty} \rho^i \sum_{j=0}^{n_i-1} \iint |t_{ij}(x, y)| dx dy$$

converges, provided  $|\rho| < 1$ . Thus — by uniform convergence of the Mehler expansion, and the theorems contained in Sections 1.71 and 1.77 of Titchmarsh [8] — it is permissible to integrate (23) term by term, yielding

$$I(\rho) = \sum_{i=0}^{\infty} \rho^i \iint s_i(x, y) dx dy.$$

The above expression is a power series expansion of  $I(\rho)$  valid for  $|\rho| < 1$ , which must be identical to the Taylor expansion, by uniqueness.  $\square$

Denote by  $A_k(\rho)$  the function

$$A_k(\rho) = \frac{k(1 - N_k(\rho))}{(k-1)(1-\rho)},$$

and recall that the performance ratio of the  $k$ -CUT heuristic is given by

$$\alpha_k = \min_{-1/(k-1) \leq \rho < 1} A_k(\rho).$$

**Corollary 1**  $\alpha_k > 1 - k^{-1}$ , for all  $k \geq 2$ .

**Proof** At  $\rho = 0$ , the numerator and denominator of  $A_k(\rho)$  are both  $k - 1$ ; at  $\rho = 1$  they are both 0. Since the power series expansion of  $N_k(\rho)$  has only positive terms, the numerator is a concave function in the range  $0 \leq \rho \leq 1$ , and hence  $A_k(\rho) \geq 1$  in that range.

Turning to the case  $\rho < 0$ , note that  $N_k(1) = 1$  and  $N_k(-1) = 0$  implies  $\sum_{i \text{ even}} a_i = \frac{1}{2}$ ; furthermore, since  $h_1(k)$  increases with  $k$  and  $h_1(3) = 3/2\sqrt{\pi}$  (using calculations described by David in [3, Section 3.2]), we have  $a_1 \geq 9/4\pi(k - 1)$ . Therefore,

$$N_k(\rho) \leq \frac{1}{k} - \frac{9(-\rho)}{4\pi(k-1)} + \frac{\rho^2}{2} \leq \frac{1}{k} - \frac{(-\rho)}{5(k-1)},$$

where the second inequality is valid over the range  $-1/(k - 1) \leq \rho \leq 0$ , since  $9/4\pi - 1/2 \geq 1/5$ ; hence

$$A_k(\rho) \geq \frac{1}{1 - \rho} \left( 1 + \frac{k(-\rho)}{5(k-1)^2} \right).$$

It is easily verified that the above expression is strictly greater than  $1 - k^{-1}$  over the closed interval  $-1/(k - 1) \leq \rho \leq 0$ .  $\square$

**Corollary 2**  $\alpha_k - (1 - k^{-1}) \sim 2k^{-2} \ln k$ .

**Proof** Galambos [4, Section 2.3.2], gives the asymptotic distribution of the maximum of  $k$  independent, normally distributed random variables. In particular the quantity  $h_1(k)$ , which is just the expectation of the maximum, satisfies  $h_1(k) \sim \sqrt{2 \ln k}$ . Thus we have the asymptotic estimate

$$N_k(\rho) = \frac{1}{k} + (1 + \epsilon(k)) \frac{2 \ln k}{k} \rho + O(\rho^2),$$

where  $\epsilon(k)$  is a function tending to 0, as  $k \rightarrow \infty$ . The result follows by arguments used in the proof of the previous corollary.  $\square$

**Corollary 3**  $\alpha_2 \geq 0.878567$ ,  $\alpha_3 \geq 0.800217$ ,  $\alpha_4 \geq 0.850304$ ,  $\alpha_5 \geq 0.874243$ ,  $\alpha_{10} \geq 0.926642$ , and  $\alpha_{100} \geq 0.990625$ .

**Proof** The value of  $\alpha_2$  was obtained by Goemans and Williamson. For  $k \geq 3$ , we use the bound  $N_k(\rho) \leq 1/k + a_1\rho + a_2\rho^2 + \rho^4/2$ , valid for  $-1 < \rho < 0$ , and evaluate  $a_1$  and  $a_2$  numerically. (Observe that the coefficient of  $\rho^3$  is positive, and hence the term itself makes a negative contribution.) Note that by computing further terms in the Taylor expansion of  $N_k(\rho)$  it is possible to give better bounds on  $\alpha_k$ ; e.g., by expanding to the term in  $\rho^4$ , we obtain  $\alpha_3 \geq 0.832718$ .  $\square$

## References

- [1] F. Alizadeh, *Interior point methods in Semi-Definite Programming with applications to Combinatorial Optimisation*. (To appear.)
- [2] E. Bofinger and V. J. Bofinger, The correlation of maxima in samples drawn from a bivariate normal distribution, *The Australian Journal of Statistics* **7** (1965), pp. 57–61.
- [3] H. A. David, *Order Statistics*, Wiley, New York, 1980.
- [4] J. Galambos, *The Asymptotic Theory of Extreme Order Statistics*, Wiley, New York, 1978.
- [5] M. X. Goemans and D. P. Williamson, .878-Approximation algorithms for MAX-CUT and MAX 2SAT, *Proceedings of the 26th Annual ACM Symposium on Theory of Computing* (1994) pp. 422–431.
- [6] D. Karger, R. Motwani, and M. Sudan, *Improved graph coloring by semidefinite programming*. (In preparation.)
- [7] G. Sansone, *Orthogonal Functions*, (translated from the Italian by A. H. Diamond), Interscience Publishers, New York, 1959.
- [8] E. C. Titchmarsh, *The Theory of Functions* (second edition), Oxford University Press, 1939.
- [9] G. N. Watson, Notes on generating functions of polynomials: Hermite polynomials, *Journal of the London Mathematical Society* **8** (1933), pp. 194–199.
- [10] D. J. A. Welsh, Complexity: Knots, Colourings and Counting, *London Mathematical Society Lecture Notes* **186**, Cambridge University Press, 1993.
- [11] L. A. Wolsey, Heuristic analysis, linear programming and branch and bound, *Mathematical Programming Study* **13: Combinatorial Optimization II**, North-Holland, 1980, pp. 121–134.



**Lecturer: Avi Wigderson**

**Title: On Rank and Communication Complexity**

Material: Noam Nisan and Avi Wigderson:  
*On Rank vs. Communication Complexity*

Institute of Computer Science  
Hebrew University  
Jerusalem, Israel  
E-mail: avi@CS.HUJI.AC.IL

# On Rank vs. Communication Complexity

Noam Nisan \*

Avi Wigderson †

## Abstract

This paper concerns the open problem of Lovász and Saks regarding the relationship between the communication complexity of a boolean function and the rank of the associated matrix. We first give an example exhibiting the largest gap known. We then prove two related theorems.

## 1 Introduction

For a 0,1 matrix  $M$ , denote by  $c(M)$  the deterministic communication complexity of the associated function [Y79], and by  $rk(M)$  its rank over the reals. It is well known [MS82] that  $\log rk(M) \leq c(M) \leq rk(M)$ . It is a fundamental question of communication complexity to narrow this exponential gap. As rank arguments are the main source of deterministic communication complexity lower bounds, and the rank function has many useful properties, it would make life nicer if the lower bound was rather tight. A tempting

---

\*Institute of Computer Science, Hebrew University of Jerusalem, Israel. This work was supported by USA-Israel BSF grant 92-00043 and by a Wolfson research award administered by the Israeli Academy of Sciences.

†Institute of Computer Science, Hebrew University of Jerusalem, Israel. This work was supported by USA-Israel BSF grant 92-00106 and by a Wolfson research award administered by the Israeli Academy of Sciences.

conjecture (see [LS88]) is

**Conjecture 1** *For every matrix  $M$ ,  $c(M) = (\log rk(M))^{O(1)}$*

Lovász and Saks [LS89] also show that this conjecture is strongly related to a conjecture of van Nuffelen [Nu76] and Fajtlowicz [Fa87] regarding the connection between the chromatic number of a graph and the rank of its adjacency matrix.

Several authors have obtained separation results between  $c(M)$  and  $\log rk(M)$  [AS89, Raz92]. The best separation known so far gives an infinite family of matrices for which  $c(M) \geq \log rk(M) \log \log \log rk(M)$  [RS93]. Our first result is an example with a much larger gap.

**Theorem 1** *There exist (explicitly given) 0-1 matrices  $M$  of size  $2^n \times 2^n$  such that  $c(M) = \Omega(n)$ , and  $\log rk(M) = O(n^\alpha)$ , where  $\alpha = \log_3 2 = 0.63\dots$*

The same  $\Omega(n)$  lower bound applies also to the randomized and to the nondeterministic communication complexities. The construction is based on boolean functions with high “sensitivity” and low degree. Such a function was constructed in [NS92]. The lower bound for the communication complexity relies on the known lower bounds for randomized communication complexity of “disjointness” [KS87, Raz90]. Recently Kushilevitz

[Ku94] has somewhat improved the construction of [NS92] and has thus reduced the value of  $\alpha$  to  $\log_6 3 = 0.61\dots$ . The main lemma of [NS92] shows however that this technique cannot reduce the value of  $\alpha$  to below  $1/2$ .

We then return our attention to conjecture 1, and consider weaker related conjectures. To explain them, we need some notation. If  $S$  is a subset of the entries of  $M$ , let  $S_0$  and  $S_1$  denote respectively the subsets of  $S$  whose value is 0 and 1 respectively. Call  $S$  *monochromatic* if either  $S = S_0$  or  $S = S_1$ . Let  $\text{mono}(M)$  denote the maximum fraction  $|A|/|M|$  over all monochromatic submatrices  $A$  of  $M$ . When  $S$  is not monochromatic, we will be interested in the advantage one color has over the other. The (*absolute*) *discrepancy* of  $S$  is  $\delta(S) = (|S_0| - |S_1|)/|M|$ . Define  $\text{disc}(M)$  to be the maximum of  $\delta(A)$  over all submatrices  $A$  of  $M$ .

Since an optimal protocol for  $M$  partitions it into at most  $2^{c(M)}$  monochromatic rectangles, we have the basic relation:

$$\text{disc}(M) \geq \text{mono}(M) \geq 2^{-c(M)}$$

or, equivalently,

$$-\log \text{disc}(M) \leq -\log \text{mono}(M) \leq c(M).$$

Thus two conjectures weaker than Conjecture 1 suggest themselves. They respectively assert that low rank matrices have large monochromatic rectangles, or weaker still, large discrepancy.

**Conjecture 2** For every  $M$ ,  
 $-\log \text{mono}(M) = (\log \text{rk}(M))^{O(1)}$

**Conjecture 3** For every  $M$ ,  
 $-\log \text{disc}(M) = (\log \text{rk}(M))^{O(1)}$

As mentioned, Conjecture 1  $\rightarrow$  Conjecture 2  $\rightarrow$  Conjecture 3. We first prove, in theorem 2, that conjectures 1 and 2 are equivalent. We then prove, in theorem 3, (a strong form of) conjecture 3.

**Theorem 2** Conjecture 1 iff Conjecture 2.

Thus in order to prove conjecture 1 it suffices to show that every low rank boolean matrix has a “large” monochromatic submatrix. In fact, the proof of the theorem implies that it suffices to show that every rank  $r$  boolean matrix has a “large” submatrix of rank at most, say,  $0.99r$ .

**Theorem 3** For every  $M$ ,  $1/\text{disc}(M) = O(\text{rk}(M)^{3/2})$ .

Note that Theorem 3 implies Conjecture 3. The bound in this theorem is nearly tight: for every  $r$  there are infinitely many matrices  $M$  of rank  $r$  and  $1/\text{disc}(M) \geq r$ . This can be easily seen by taking any square array of  $r \times r$  Hadamard matrices.

This theorem supplies the first clue that low rank has something to do with low communication complexity, though in a very weak sense. The communication model we have in mind is distributional communication complexity, where the inputs are chosen at random [Y83]. For this model, low rank guarantees a cheap protocol with a nontrivial advantage over guessing the function value. In the protocol each player sends one bit specifying whether or not his input is in the biased rectangle. Precisely:

**Corollary 1** If  $\text{rk}(M) = r$ , then there is a 2 bit protocol  $P$ , which satisfies  $\Pr[P(x, y) = M(x, y)] \geq 1/2 + \Omega(1/r^{3/2})$ , where the input  $(x, y)$  is chosen uniformly at random.

## 2 Proof of Theorem 1

We will require the following definition.

**Definition:** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a boolean function. We say that  $f$  is fully sensitive at  $\vec{0}$  if  $f(\vec{0}) = 0$  and yet for any



vector  $x$  of hamming weight 1 (i.e. for any unit vector),  $f(x) = 1$ .

The degree of  $f$ ,  $\deg(f)$  is defined to be the degree of the unique multivariate multi-linear polynomial over the reals which agrees with  $f$  on  $\{0, 1\}^n$ .

In [NS92] it is shown that any boolean function which is fully sensitive at  $\vec{0}$  must have degree of at least  $\sqrt{n}/2$ . They also give an example of a fully sensitive function with degree significantly less than  $n$ .

**Lemma 1** [NS92] *There exists an (explicitly given) boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  which is fully sensitive at  $\vec{0}$  and  $\deg(f) = n^\alpha$ , for  $\alpha = \log_3 2 = 0.63\dots$ . Furthermore,  $f$  has at most  $2^{O(n^\alpha)}$  monomials.*

For completeness we repeat the construction of [NS92].

**Proof:** Let  $E(z_1, z_2, z_3)$  be the symmetric boolean function giving 1 iff exactly 1 or 2 of its inputs are 1. It is easy to check that  $E$  is fully sensitive at  $\vec{0}$ . One may also readily verify that  $\deg(E) = 2$  as  $E(z_1, z_2, z_3) = z_1 + z_2 + z_3 - z_1z_2 - z_1z_3 - z_2z_3$ . We now recursively define a function  $E_k$  on  $3^k$  input bits by:  $E^0(z) = z$ , and  $E^k(\cdot) = E(E^{k-1}(\cdot), E^{k-1}(\cdot), E^{k-1}(\cdot))$ , where each instance of  $E^{k-1}$  is on a different set of  $3^{k-1}$  input bits. It is easy to prove by induction that (1)  $E^k$  is fully sensitive at  $\vec{0}$ , (2)  $\deg(E^k) = 2^k$ , and (3)  $E^k$  has at most  $6^{2^k-1}$  monomials. Our desired  $f$  is the function  $E^k$  on  $n = 3^k$  variables<sup>1</sup>.  $\square$

We now transform  $f$  into a matrix as follows.

<sup>1</sup>Recently, [Ku94] has improved upon this construction by exhibiting a function  $E'$  on 6 variables which is fully sensitive at  $\vec{0}$  and with degree only 3. Using the same recursion, this reduces  $\alpha$  to  $\log_6 3 = 0.61\dots$

**Definition:** With every boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  we associate a  $2^n \times 2^n$  matrix  $M_f$  as follows:

$$M_f(x_1 \dots x_n; y_1 \dots y_n) = f(x_1 \cdot y_1, x_2 \cdot y_2, \dots, x_n \cdot y_n)$$

The properties of  $M_f$  are ensured by the following lemmas.

**Lemma 2** *If  $f$  is fully sensitive at  $\vec{0}$  then  $c(M_f) = \Omega(n)$ . The same lower bound holds for the randomized and for the nondeterministic complexity of  $M_f$ .*

**Lemma 3** *Let  $f$  be a polynomial with  $m$  monomials, then  $\text{rk}(M_f) \leq m$ . In particular, if  $d = \deg(f)$  then  $\text{rk}(M_f) \leq \sum_{i=0}^d \binom{n}{i} = 2^{O(d \log n)}$ .*

**Proof** (of lemma 2): This proof is a direct reduction from the known lower bounds for the randomized communication complexity of disjointness. These bounds actually show that it is even hard to distinguish between the case where the sets are disjoint and the case where the intersection size is 1.

Let the *UDISJ* problem be the following: the two players are each given a subset of  $\{1 \dots n\}$ . If the sets are disjoint they must accept. If the sets intersect at exactly 1 point then they must reject. If the size of the intersection is greater than 1 then the players are allowed to either accept or reject.

**Theorem** ([KS87], see also [Raz90]): Any communication complexity protocol for *UDISJ* requires  $\Omega(n)$  bits of communication. The same is true for non-deterministic and for randomized protocols.

Now notice that if  $f$  is fully sensitive at  $\vec{0}$  then any protocol for  $M_f$  directly solves *UDISJ*. This is done by transforming each set to its characteristic vector. If the sets are disjoint then for each  $i$ ,  $x_i y_i = 0$ , and thus

$M_f(\vec{x}, \vec{y}) = f(\vec{0}) = 0$ . If the intersection size is exactly 1 then in exactly 1 position  $x_i y_i = 1$ , and thus  $M_f(\vec{x}, \vec{y}) = 1$ .  $\square$

**Proof** (of lemma 3): Let  $f(z_1 \dots z_n) = \sum_S \alpha_S \prod_{i \in S} z_i$  be the representation of  $f$  as a real polynomial. By the definition of  $M_f$  it follows that  $M_f = \sum_S \alpha_S M_S$ , where the matrix  $M_S$  is defined by  $M_S(\vec{x}, \vec{y}) = \prod_{i \in S} x_i \cdot y_i$ . But clearly for each  $S$ ,  $rk(M_S) = 1$ . It follows that the rank of  $M_f$  is bounded from above by the number of non-zero monomials of  $f$ . The bound in terms of the degree follows directly.  $\square$

The combination of lemmas 2 and 3 with the function  $E^k$  constructed in lemma 1 gives the statement of the theorem.  $\square$

### 3 Proof of Theorem 2

Assume conjecture 2, i.e. assume that every 0, 1 matrix  $M$  has a monochromatic submatrix of size  $|M|/\exp(\log^k rk(M))$ . Given a 0, 1 matrix  $M$  we will design a communication protocol for  $M$ .

Let  $A$  be the largest monochromatic submatrix of  $M$ . Then  $A$  induces in a natural way a partition of  $M$  into 4 submatrices  $A, B, C, D$ , with  $B$  sharing the rows of  $A$  and  $C$  sharing the columns of  $A$ . Clearly  $rk(B) + rk(C) \leq rk(M) + 1$ . Assume w.l.o.g. that  $rk(B) \leq rk(C)$ , then the submatrix  $(A|B)$  has rank at most  $2 + rk(M)/2$ .

In our protocol the row player sends a bit saying if his input belongs to the rows of  $A$  or not. The players then continue recursively with a protocol for the submatrix  $(A|B)$ , or for the submatrix  $(C|D)$ , according to the bit communicated.

Denote by  $L(m, r)$  the number of leaves of this protocol, starting with a matrix of area at most  $m$  and rank at most  $r$ . By the protocol presented we get a recurrence  $L(m, r) \leq L(m, 2 + r/2) + L(m(1 - \alpha), r)$ , where  $\alpha$  is

the fraction of rows in  $A$ . By the assumption,  $\alpha \geq (\exp(\log^k r))^{-1}$ . Note that (assuming the players ignore identical rows and columns) that  $m \leq 2^r$ , and that  $L(m, 1) = 1$ . It is standard to see that the solution to the recurrence satisfies  $L(m, r) \leq \exp(\log^{k+1} r)$ .

We have so far obtained a protocol for  $M$  with  $\exp(\log^{k+1} rk(M))$  leaves; it is well known that this implies also  $c(M) \leq O(\log^{k+1} rk(M))$ .  $\square$

**Remark:** Note that the same proof, yielding essentially the same bound, would go through even if instead of a large monochromatic (rank 1) submatrix we were promised a large submatrix of rank  $r/4$ , say. The idea is that for the decomposition  $A, B, C, D$  in the proof we have in general  $rk(B) + rk(C) \leq rk(M) + rk(A)$ . We used it above for a monochromatic  $A$ , so  $rk(A) \leq 1$ . Now we have  $rk(A) \leq r/4$ , and using  $rk(B) \leq rk(C)$  we get  $rk(B) \leq (rk(M) + rk(A))/2 \leq 5r/8$ . Thus  $rk(A|B) \leq rk(A) + rk(B) \leq 7r/8$ . The recurrence relation changes to  $L(m, r) \leq L(m, 7r/8) + L(m(1 - \alpha), r)$ , which has the same asymptotic behavior.

The expression  $r/4$  may be replaced by  $\alpha r$  for any  $\alpha < 1$  by repeatedly taking a large submatrix of low rank of the current submatrix. After constant number of times the rank is reduced to  $r/4$ . Again, this does not change the asymptotics of the recurrence.

### 4 Proof of Theorem 3

Let us consider  $-1, +1$  matrices rather than 0, 1 matrices; this obviously changes the rank by at most 1, and does not change the discrepancy. The advantage is that the discrepancy of a submatrix  $N$  of  $M$  has a simple form:  $\delta(N)$  is the sum of entries of  $N$ , divided by the area of  $M$ .

We will use the following notation. Let  $x = (x_i) \in R^n$  and  $A = (a_{ij})$  be an  $n \times n$  real

matrix. Then:

- $\|x\| = (\sum_{i=1}^n x_i^2)^{1/2}$ , the  $L_2$  norm of  $x$ .
- $\|x\|_\infty = \max_{i=1}^n |x_i|$ , the  $L_\infty$  norm of  $x$ .
- $\|A\| = \max_{\|x\|=1} \|Ax\|$ , the spectral norm of  $A$ . It is well known that also  $\|A\| = \max_{\|x\|=1, \|y\|=1} |x^T Ay|$ ; and  $\|A\| = \max\{\sqrt{\lambda} : \lambda \text{ is an eigenvalue of } A^T A\}$ .
- $W(A) = (\sum_{i,j=1}^n a_{ij}^2)^{1/2}$ , the Euclidean norm of  $A$ .
- $tr(A) = \sum_{i=1}^n a_{ii}$ , the trace of  $A$ .

**Overview of Proof:** It is best to summarize the proof backwards. We are given a  $\pm 1$  matrix  $A$  of low rank and wish to find in it a submatrix of high discrepancy. This is done in lemma 6 and is clearly equivalent to finding 0,1 vectors  $x$  and  $y$  such that  $x^T Ay$  is large. As an intermediate step we shall, in lemma 5, find real vectors  $u$  and  $v$ , having low  $L_\infty$ -norm, with  $u^T Av$  large. Towards this we shall need real vectors  $w$  and  $z$  having low  $L_2$ -norm, with  $w^T Az$  large. This is equivalent to proving lower bounds on  $\|A\|$ , which we do in lemma 4.

**Lemma 4** For every real matrix  $A$ ,

$$\frac{W(A)}{\sqrt{rk(A)}} \leq \|A\| \leq W(A)$$

**Proof:** Let  $r = rk(A)$ . Let us compute the trace of  $A^T A$ . On one hand, direct calculation by definition shows that  $tr(A^T A) = W(A)^2$ . On the other hand  $tr(A^T A) = \sum_i \lambda_i$ , where the sum is over all eigenvalues  $\lambda_i$  of  $A^T A$ . Since  $A^T A$  has only  $r$  non-zero eigenvalues, and since all eigenvalues of  $A^T A$  are positive, the largest eigenvalue,  $\lambda_1$ , is bounded by  $\frac{W(A)^2}{r} \leq \lambda_1 \leq W(A)^2$ . The lemma follows since  $\|A\| = \sqrt{\lambda_1}$ .  $\square$

**Lemma 5** Let  $A$  be an  $n \times n$   $\pm 1$  matrix of rank  $r$ . Then there exist vectors  $u, v$ ,  $\|u\|_\infty \leq 1$ ,  $\|v\|_\infty \leq 1$ , such that  $u^T Av \geq \frac{n^2}{16r^{3/2}}$ .

**Proof:** Denote  $r = rk(A)$ . Let  $x$  and  $y$  be vectors such that  $\|x\| = 1$ ,  $\|y\| = 1$ , and  $x^T Ay = \|A\|$ . Let  $I = \{i : |x_i| > \sqrt{8r/n}\}$  and  $J = \{j : |y_j| > \sqrt{8r/n}\}$ . Notice that  $|I| \leq n/(8r)$ , and  $|J| \leq n/(8r)$ .

Let  $\hat{u}$  be the vector that agrees with  $x$  outside of  $I$  and is 0 for indices in  $I$ , and let  $\hat{v}$  be the vector that agrees with  $y$  outside of  $J$  and is 0 for indices in  $J$ .

We shall compute a lower bound on  $\hat{u}^T A \hat{v}$ . Consider the matrix  $B$  defined to agree with  $A$  on all entries  $i, j$  such that  $i \in I$  or  $j \in J$ , and to be 0 elsewhere. Using this notation it is clear that

$$\hat{u}^T A \hat{v} = x^T Ay - x^T B y.$$

A lower bound for  $x^T Ay = \|A\|$  is obtained using the lower bound in lemma 4, and as  $W(A) = n$ ,  $x^T Ay \geq n/\sqrt{r}$ . An upper bound for  $x^T B y$  is given by the upper bound in the last lemma  $x^T B y \leq \|B\| \leq W(B)$ . Since  $B$  has at most  $n/(8r)$  non-zero rows and  $n/(8r)$  non-zero columns,  $W(B) \leq n/(2\sqrt{r})$ . It follows that  $\hat{u}^T A \hat{v} \geq n/(2\sqrt{r})$ .

Now define  $u = \sqrt{n/(8r)} \hat{u}$  and  $v = \sqrt{n/(8r)} \hat{v}$ . By definition  $\|v\|_\infty \leq 1$  and  $\|u\|_\infty \leq 1$ . The lemma follows since  $u^T Av = n/(8r) \hat{u}^T A \hat{v}$ .  $\square$

**Lemma 6** Let  $A$  be an  $n \times n$  matrix, and  $u, v$  vectors such that  $\|u\|_\infty \leq 1$ ,  $\|v\|_\infty \leq 1$ . Then there exists a submatrix  $B$  of  $A$  with  $\delta(B) \geq u^T Av / (4n^2)$ .

**Proof:** Let  $z = Av$ . Clearly,  $\sum_{i \in K} u_i z_i \geq u^T Av / 2$ , where  $K$  is either the coordinates where both  $u_i$  and  $z_i$  are positive or the coordinates in which both are negative. Assume the first case (otherwise replace below  $v \leftarrow$

$-v$ ). Then setting  $x = \chi_K$  (the characteristic vector of  $K$ ), we have (using  $\|u\|_\infty \leq 1$ ),  $x^T A v \geq u^T A v / 2$ . Repeating this argument with  $z = x^T A$ , we can replace  $v$  with a 0, 1 vector  $y$  obtaining  $x^T A y \geq u^T A v / 4$ . Now take  $B$  to be the submatrix defined by the 1's in  $x$  and  $y$ . Since  $B$  is a  $\pm 1$  matrix, the bilinear form divided by  $n^2$  gives its discrepancy.  $\square$

Combining lemmas 5 and 6, every  $\pm 1$  matrix  $A$  of rank  $r$ , contains a submatrix  $B$  with  $\delta(B) \geq \frac{1}{64r^{3/2}}$ . Thus  $\text{disc}(M) \geq \frac{1}{64r^{3/2}}$ , and theorem 3 follows.  $\blacksquare$

## Acknowledgements

We thank Oded Goldreich, Russell Impagliazzo, Mauricio Karchmer, Eyal Kushilevitz, and Roman Smolensky for conversations on this subject.

## References

- [AS89] N. Alon, P. Seymour, "A counter example to the rank-covering conjecture", *J. Graph Theory* 13, pp. 523–525, 1989.
- [Fa87] S. Fajtlowicz, "On conjectures of Graffiti" II, *Congressus Numeratum* 60, pp. 189–198, (1987).
- [KS87] B. Kalyanasundaram and G. Schnitger, "The probabilistic communication complexity of set intersection", *2nd Structure in Complexity Theory Conference*, pages 41–49, 1987.
- [Ku94] E. Kushilevitz, manuscript, 1994.
- [LS88] L. Lovász and M. Saks, "Lattices, Möbius functions, and communication complexity", *Proc. of the 29th FOCS*, pp. 81–90, 1988.
- [LS89] L. Lovász and M. Saks, Private communication.
- [MS82] K. Mehlhorn, E.M. Schmidt, "Las Vegas is better than determinism in VLSI and distributive computing", *Proceedings of 14th STOC*, pp. 330–337, 1982.
- [NS92] N. Nisan and M. Szegedy, "On the degree of boolean functions as real polynomials", *Proceedings of 24th STOC*, pp. 462–467, 1992.
- [Nu76] C. van Nuffelen, "A bound for the chromatic number of a graph", *American Mathematical Monthly* 83, pp. 265–266, 1976.
- [Raz90] A. Razborov, "On the distributional complexity of disjointness", *Proceedings of the ICALP*, pp. 249–253, 1990. (to appear in *Theoretical Computer Science*).
- [Raz92] A. Razborov, "The gap between the chromatic number of a graph and the rank of its adjacency matrix is super-linear", *Discrete Math.* 108, pp 393–396, 1992.
- [RS93] R. Raz and B. Spiker, "On the Log-Rank conjecture in communication complexity", *Proc. of the 34th FOCS*, pp. 168–176, 1993
- [Y79] A. C.-C. Yao, "Some complexity questions related to distributive computing", *Proceedings of 11th STOC*, pp. 209–213 (1979).
- [Y83] A. C.-C. Yao, "Lower Bounds by Probabilistic Arguments", *Proc. 24th FOCS*, pp. 420–428, (1983).

## Recent Publications in the BRICS Notes Series

- NS-94-4** Peter D. Mosses, editor. *Abstracts of the 6th Nordic Workshop on PROGRAMMING THEORY* (Aarhus, Denmark, 17–19 October, 1994), October 1994. v+52 pp.
- NS-94-3** Sven Skyum, editor. *Complexity Theory: Present and Future* (Aarhus, Denmark, 15–18 August, 1994), September 1994. v+213 pp.
- NS-94-2** David Basin. *Induction Based on Rippling and Proof Planning. Mini-Course.* August 1994. 62 pp.
- NS-94-1** Peter D. Mosses, editor. *Proc. 1st International Workshop on Action Semantics* (Edinburgh, 14 April, 1994), May 1994. 145 pp.