## Separations in Query Complexity Based on Pointer Functions (with a slight hint of quantum complexity)

Alexander Belov CWI

Joint work with: Andris Ambainis, Kaspars Balodis, Troy Lee, Miklos Santha, and Juris Smotrovs
(presented at QIP'16, to appear in STOC'16)

# Introduction 



## Computational Models: Deterministic



## Computational Models: Deterministic



## Computational Models: Deterministic



## Computational Models: Randomised

## Introduction

Deterministic
Randomised
Quantum
Separations
Overview of Results
$R_{1}$ versus $R_{0}$
$R_{0}$ versus $D$

## Conclusion

$D$ : Deterministic (Decision Tree)
$R$ : Randomised (Probability distribution on decision trees)

$a, b, c$ : uniform random permutation of $1,2,3$.
Complexity

- on input: Expected number of queries 2 or $\frac{8}{3}$
- in total:

Worst input
$\frac{8}{3}$

## Computational Model: Randomised

## Introduction

Deterministic
Randomised
Quantum
Separations
$D$ : Deterministic (Decision Tree)
$R$ : Randomised (Probability distribution on decision trees)
$R_{0}$ : Zero-error (Las Vegas)

- always outputs the correct output


## Computational Models: Randomised

## Introduction

$\underline{R_{1} \text { versus } R_{0}}$
$R_{0}$ versus $D$

## Conclusion

$D$ : Deterministic (Decision Tree)
$R$ : Randomised (Probability distribution on decision trees)

$$
R_{0}: \text { Zero-error (Las Vegas) }
$$

- always outputs the correct output


## $R_{2}$ : Bounded-error (Monte Carlo)

- rejects a negative input with probability $\geq \frac{2}{3}$
- accepts a positive input with probability $\geq \frac{2}{3}$


## Computational Models: Randomised

$D: \quad$ Deterministic (Decision Tree)
$R$ : Randomised (Probability distribution on decision trees)
$R_{0}$ : Zero-error (Las Vegas)

- always outputs the correct output
$R_{1}$ : One-sided error
- always rejects a negative input
- accepts a positive input with probability $\geq \frac{1}{2}$ (or vice versa)
$R_{2}$ : Bounded-error (Monte Carlo)
- rejects a negative input with probability $\geq \frac{2}{3}$
- accepts a positive input with probability $\geq \frac{2}{3}$


## Computational Models: Quantum

## Introduction

Deterministic
Randomised
Quantum
Separations

D: Deterministic (Decision Tree)
$R$ : Randomised (Probability distribution on decision trees)
$R_{0}$ : Zero-error (Las Vegas)

- always outputs the correct output
$R_{1}$ : One-sided error
- always rejects a negative input
- accepts a positive input with probability $\geq \frac{1}{2}$ (or vice versa)
$R_{2}$ : Bounded-error (Monte Carlo)
- rejects a negative input with probability $\geq \frac{2}{3}$
- accepts a positive input with probability $\geq \frac{2}{3}$

Q: Quantum
$Q_{E}$ : Exact
$Q_{2}$ : Bounded-error

## Separations

## Easy for partial functions

## Separations

## Introduction

Deterministic
Randomised
Quantum
Separations
$R_{1}$ versus $R_{0}$
$\underline{R_{0} \text { versus } D}$

## Conclusion

Easy for partial functions

## Example: Deutsch-Jozsa problem (almost)

- Reject iff all input variables are zeroes

- Accept iff exactly half of the variables are ones



## Separations

## Introduction

Deterministic
Randomised
Quantum
Separations

Easy for partial functions

## Example: Deutsch-Jozsa problem (almost)

- Reject iff all input variables are zeroes

- Accept iff exactly half of the variables are ones


$$
R_{1}=1
$$

## Separations

## Introduction

Deterministic
Randomised
Quantum
Separations

Easy for partial functions

## Example: Deutsch-Jozsa problem (almost)

- Reject iff all input variables are zeroes

- Accept iff exactly half of the variables are ones

$$
\begin{array}{|l|l|l|l|l|l|l|}
\hline 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
\hline
\end{array}
$$

$$
R_{1}=1, \quad Q_{E}=1
$$

## Separations

## Introduction

Deterministic
Randomised
Quantum
Separations

Easy for partial functions

## Example: Deutsch-Jozsa problem (almost)

- Reject iff all input variables are zeroes

- Accept iff exactly half of the variables are ones

$$
\begin{aligned}
& \begin{array}{|l|l|l|l|l|l|l|}
\hline \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\
\hline
\end{array} \\
& R_{1}=1, \quad Q_{E}=1, \\
& R_{0}=n / 2+1
\end{aligned}
$$



## Separations

Easy for partial functions
Example: Deutsch-Jozsa problem (almost)

- Reject iff all input variables are zeroes

- Acrant iff nuantly half of the wariablac arn anac


## Total Functions — ???



# Overview of Results 

$\underline{R_{1} \text { versus } R_{0}}$
$R_{0}$ versus $D$
Conclusion

## Iterated Functions

We have just seen $D\left(M A J_{3}\right)=3$ and $R_{0}\left(M A J_{3}\right)=8 / 3$.


## Iterated Functions

We have just seen $D\left(M A J_{3}\right)=3$ and $R_{0}\left(M A J_{3}\right)=8 / 3$. Iterate it:

## Iterated Functions

## Introduction

## Overview of Results

Iterated Functions
Record-Holder
Our Main Results

## Göös-Pitassi-Watson

Our Modifications
$R_{1}$ versus $R_{0}$
$R_{0}$ versus $D$
Conclusion

We have just seen $D\left(M A J_{3}\right)=3$ and $R_{0}\left(M A J_{3}\right)=8 / 3$.

## Iterate it:



We get

$$
D\left(M A J_{3}^{d}\right)=3^{d} \quad \text { and } \quad R_{0}\left(M A J_{3}^{d}\right) \leq(8 / 3)^{d} .
$$

(Actually, it is less...)

## Previous Record-Holder

## Introduction

Overview of Results

Iterated Functions
Record-Holder
Our Main Results

## Göös-Pitassi-Watson

Our Modifications
$R_{1}$ versus $R_{0}$
$R_{0}$ versus $D$
Conclusion

Iterated NAND: record-holder for $R_{0}, R_{1}, R_{2}$ versus $D$


We have [Snir'85, Saks \& Wigderson'86]:

$$
R_{0}=R_{1}=R_{2}=O\left(n^{0.7537 \ldots}\right), \quad D=n
$$

## State of the Art

## We have [Snir'85, Saks \& Wigderson'86]:

$$
R_{0}=R_{1}=R_{2}=O\left(n^{0.7537 \ldots}\right), \quad D=n
$$

It is known [Nisan'89]

$$
D=O\left(R_{1}^{2}\right)
$$

## Our Main Results

It is known [Nisan'89]

$$
D=O\left(R_{1}^{2}\right)
$$

We get functions with:

$$
D=\widetilde{\Theta}\left(R_{0}^{2}\right)
$$

$$
R_{0}=\widetilde{\Theta}\left(R_{1}^{2}\right)
$$



## Our Main Results

It is known [Nisan'89]

$$
D=O\left(R_{1}^{2}\right)
$$

We get functions with:

$$
D=\widetilde{\Theta}\left(R_{0}^{2}\right)
$$

$$
R_{0}=\widetilde{\Theta}\left(R_{1}^{2}\right)
$$



The last one also saturates [Kulkarni \& Tal'13, Midrijānis'05]

$$
R_{0}=\widetilde{O}\left(R_{2}^{2}\right)
$$

# Göös-Pitassi-Watson 

Adversary Method
$D$ Lower Bound
Features of Pointers
Our Modifications
$\underline{R_{1} \text { versus } R_{0}}$
$R_{0}$ versus $D$
Conclusion
$\underline{\text { Introduction }}$

Overview of Results Paper

Goal
$D$ versus 1-certificates
Pointers
Adversary Method
$D$ Lower Bound
Features of Pointers
Our Modifications
$R_{1}$ versus $R_{0}$
$R_{0}$ versus $D$

## Deterministic Communication vs. Partition Number

Mika Göös Toniann Pitassi Thomas Watson

Department of Computer Science, University of Toronto

$$
\text { April 1, } 2015
$$

## Abstract

We show that deterministic communication complexity can be superlogarithmic in the partition number of the associated communication matrix. We also obtain near-optimal deterministic lower bounds for the Clique vs. Independent Set problem, which in particular yields new lower bounds for the log-rank conjecture. All these results follow from a simple adaptation of a communication-to-query simulation theorem of Raz and McKenzie (Combinatorica 1999) together with lower bounds for the analogous query complexity questions.

## Goal

- Clique vs. Independent Set in communication complexity

There exists a number of 1-certificates such that each positive input satisfies exactly one of them.

## $D$ versus 1-certificates

## Function on $n m$ Boolean variables

- Accept iff there exists a unique all-1 column

- $D=n m$
- short 1-certificates $(n+m-1)$, BUT not unambiguous.


## $D$ versus 1-certificates

## Function on $n m$ Boolean variables

- Accept iff there exists a unique all-1 column

- $D=n m$
- short 1-certificates $(n+m-1)$, BUT not unambiguous.

Should specify which zero to take in each column!

## Pointers

| Introduction |
| :--- |
| Overview of Results |
| Göös-Pitassi-Watson |
| Paper |
| Goal |
| $D$ versus 1-certificates |

Pointers
Adversary Method
$D$ Lower Bound
Features of Pointers

Our Modifications
$R_{1}$ versus $R_{0}$
$\underline{R_{0} \text { versus } D}$
Conclusion


- Alphabet: $\{0,1\} \times([n] \times[m] \cup\{\perp\})$

Not Boolean, but we can encode using $O(\log (n+m))$ bits.

- Accept iff
$\square$ There is a (unique) all- 1 column $b$;
$\square$ in $b$, there is a unique element $r$ with non-zero pointer;
$\square$ following the pointers from $r$, we traverse through exactly one zero in each column but $b$.


## Pointers

$\underline{\text { Introduction }}$
Overview of Results


- short unambiguous 1-certificates $(n+m-1)$


## Pointers



- short unambiguous 1-certificates $(n+m-1)$
- Still have $D=n m$ (Adversary argument, next slide)


## Adversary Method

Adversary finds a bad input for each deterministic decision tree, by playing along with the decision tree.

## Adversary Method

Adversary finds a bad input for each deterministic decision tree, by playing along with the decision tree.


## Adversary Method

## Introduction

Overview of Results

Göös-Pitassi-Watson
Paper
Goal
$D$ versus 1-certificates
Pointers
Adversary Method
D Lower Bound
Features of Pointers
Our Modifications
$R_{1}$ versus $R_{0}$
$R_{0}$ versus $D$
Conclusion

Adversary finds a bad input for each deterministic decision tree, by playing along with the decision tree.
irrelevant


## Adversary Method

## Introduction

Overview of Results

Göös-Pitassi-Watson
Paper
Goal
$D$ versus 1-certificates
Pointers
Adversary Method
D Lower Bound
Features of Pointers
$R_{1}$ versus $R_{0}$
$R_{0}$ versus $D$
Conclusion

Adversary finds a bad input for each deterministic decision tree, by playing along with the decision tree.
irrelevant


## Adversary Method

Adversary finds a bad input for each deterministic decision tree, by playing along with the decision tree.


For each queried variable, the adversary provides the value, so that the value of the function is unknown as long as possible.

## Deterministic Lower Bound

■ While there are non-queried elements in a column:
$\square$ Return 1 .

- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .



## Deterministic Lower Bound

- While there are non-queried elements in a column:
$\square$ Return 1.
- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .

|  |  |  |  |  | 1 |  | $!$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  | 1 | 1 |  |
| 1 |  |  |  | 1 |  |  |  |
|  |  |  |  |  | 1 |  |  |
|  |  |  |  | 1 | 1 |  |  |
|  |  |  |  |  | 1 |  |  |
|  |  |  |  |  | 1 | 1 |  |
| 1 |  |  | 1 |  | 1 |  | 1 |

## Deterministic Lower Bound

- While there are non-queried elements in a column:
$\square$ Return 1.
- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .

|  |  |  |  |  | 1 |  | $!$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  | 1 | 1 |  |
| 1 |  |  |  | 1 | 0 |  |  |
|  |  |  |  |  | 1 |  |  |
|  |  |  |  | 1 | 1 |  |  |
|  |  |  |  |  | 1 |  |  |
|  |  |  |  |  | 1 | 1 |  |
| 1 |  |  | 1 |  | 1 |  | 1 |

## Deterministic Lower Bound

- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .

|  | 1 |  | 1 |  | 1 |  | $!$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | 1 |  | 1 | 1 |  |
| 1 |  |  | 1 | 1 | 0 |  |  |
|  |  |  | 1 |  | 1 | 1 |  |
|  |  |  | 1 | 1 | 1 |  |  |
|  |  |  | 0 |  | 1 |  |  |
| 1 | 1 | 1 | 1 |  | 1 | 1 |  |
| 1 |  |  | 1 |  | 1 |  | 1 |

## Deterministic Lower Bound

- While there are non-queried elements in a column:
$\square$ Return 1.
- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .

\left.| 1 | 1 | 1 | 1 |  | 1 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | 0 | 1 |  | 1 |  |  |
| 1 | 1 | 1 | 1 | 1 | 1 |  |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |$\right)$

## Deterministic Lower Bound

- While there are non-queried elements in a column:
$\square$ Return 1.
- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | 0 | -1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |  | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

## Deterministic Lower Bound

- While there are non-queried elements in a column:
$\square$ Return 1.
- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | 0 | -1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

## Deterministic Lower Bound

- While there are non-queried elements in a column:
$\square$ Return 1.
- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 1 | 1 | 1 | 1 | 1 |  |  |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

## Features of Pointers

Highly elusive (flexible)


Still traversable (if know where to start).

# Our Modifications 

## Binary Tree

Definition (base)

$R_{1}$ versus $R_{0}$
$\underline{R_{0} \text { versus } D}$

## Conclusion

Instead of a list

we use a balanced binary tree


- More elusive
- Random access


## Definition (base)



## Accept iff

- There is a (unique) all-1 column $b$;
- in $b$, there is a unique element $r$ with non-zero pointers;
- for each $j \neq b$, following a path $T(j)$ from $r$ gives a zero in the $j$ th column.


## Definition (base)



## Accept iff

- There is a (unique) all-1 column $b$;
- in $b$, there is a unique element $r$ with non-zero pointers;
- for each $j \neq b$, following a path $T(j)$ from $r$ gives a zero in the $j$ th column.
- Some additional information is contained in the leaves (to be defined).


## $R_{1}$ versus $R_{0}$

## State of the Art

# ■ NO separation was known even between $R_{2}$ and $R_{0}$. 

## Reminder 1: Partial Separation

Recall the separation for a partial function

- Reject iff all input variables are zeroes

- Accept iff exactly half of the variables are ones



## Reminder 2: Definition (base)

$R_{1}$ versus $R_{0}$

State of the Art
Reminder 1
Reminder 2
Definition
Totalisation
Check Column
$R_{1}$ Upper Bound
$R_{0}$ Lower Bound
Summary
$R_{0}$ versus $D$
Conclusion


## Accept iff

- There is a (unique) all-1 column $b$;
- in $b$, there is a unique element $r$ with non-zero pointers;
- for each $j \neq b$, following a path $T(j)$ from $r$ gives a zero in the $j$ th column.
- Some additional information is contained in the leaves (to be defined).


## Definition



## Accept iff

- There is a (unique) all-1 column $b$;
- in $b$, there is a unique element $r$ with non-zero pointers;
- for each $j \neq b$, following a path $T(j)$ from $r$ gives a zero in the $j$ th column.
■ exactly $m / 2$ of the leaves back point to the root $r$.


## Totalisation



A column is good if it contains a leaf back pointing to the root of a legitimate tree.

- A positive input contains exactly $m / 2$ good columns.
- A negative input contains no good columns.


## Totalisation



A column is good if it contains a leaf back pointing to the root of a legitimate tree.

- A positive input contains exactly $m / 2$ good columns.
- A negative input contains no good columns.

A total function looks like a partial function!

## Check Column: Informal

Reminder 1
Reminder 2
Definition
Totalisation
Check Column
$R_{1}$ Upper Bound
$R_{0}$ Lower Bound
Summary
$R_{0}$ versus $D$
Conclusion


## Check Column: Informal

## Deterministic subroutine

Given a column $c \in[m]$, accept iff it is good.


Go through column $c$, find the back pointer to $r$, and check the tree.

## Check Column: Informal

## Deterministic subroutine

Given a column $c \in[m]$, accept iff it is good.


Go through column $c$, find the back pointer to $r$, and check the tree. Wait, column $c$ may contain many bogus pointers — ???

## Check Column: Informal

## Deterministic subroutine

Given a column $c \in[m]$, accept iff it is good.


Go through column $c$, find the back pointer to $r$, and check the tree.
Wait, column $c$ may contain many bogus pointers — ???
On each step, either

- eliminate a column: it is not the all-1 column; or
- eliminate an element in column $c$ : it is not a leaf of the tree.


## Check Column: Formal

## Deterministic subroutine

Given a column $c \in[m]$, accept iff it is good.


- While there is $\geq 2$ non-eliminated columns:
$\square$ Let $a$ be a non-eliminated element in $c$. If none, reject.
$\square$ Let $r$ be the back pointer of $a$, and $b$ be the column of $r$.
$\square \quad$ Let $j$ be a non-eliminated column $\neq b$.
$\square$ If the path $T(j)$ from $r$ ends in a zero in column $j$, eliminate column $j$. Otherwise, eliminate element $a$.
- Verify the only non-eliminated column.


## $R_{1}$ Upper Bound



- On each iteration of the loop, either an element or a column gets eliminated. At most $n+m$ iterations.
Complexity: $\widetilde{O}(n+m)$.
Sticking into Deutsch-Jozsa, get $R_{1}$ and $Q_{E}$ upper bound of

$$
\widetilde{O}(n+m)
$$

## $R_{0}$ Lower Bound

Introduction $\quad$ :
(Negative) input with exactly one zero in each column.

## $R_{0}$ Lower Bound


(Negative) input with exactly one zero in each column.

- An $R_{0}$ algorithm can reject only if it has found $m / 2$ zeroes.


## $R_{0}$ Lower Bound

|  | 1 | 1 | 1 | 1 | 1 |  |  | 1 | 1 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Overview of Results | 1 | 1 | 1 | 0 | 1 |  |  | 0 | 1 |
| Gubs.Pliass:-Walson | 1 | 1 | 1 | 1 | . 1 |  |  | 1 | 1 |
| Our Modificitions | 1 | 1 | 1 | 1 | 0 |  |  | 1 | 1 |
| $R_{\text {vesus }} R_{0}$ | 0 | 1 | 1 | 1 | 1 |  |  | 1 | 1. |
| State of the At | 1 | 1 | 0 | 1 | 1 |  |  | 1 | 0 |
| Reminder 1 | 1 | 1 | 1 | 1 | 1 |  |  | 1 | 1 |
|  | 1 | 0 |  | 1 | 1 |  |  | 1 | 1 |

(Negative) input with exactly one zero in each column.

- An $R_{0}$ algorithm can reject only if it has found $m / 2$ zeroes.

Requires $\Omega(n m)$ queries.

## Summary

- Upper bound for $R_{1}$ and $Q_{E}$ is $\widetilde{O}(n+m)$.
- Lower bound for a $R_{0}$ algorithm is $\Omega(n m)$.

Reminder 2
Definition
Taking $n=m$, we get a quadratic separation between $R_{1}$ and $R_{0}$, as well as between $Q_{E}$ and $R_{0}$

NB. The previous separation was [Ambainis'12]:

$$
Q_{E}=O\left(R_{0}^{0.8675 \ldots}\right)
$$

$\qquad$
Overview of Results
Göös-Pitassi-Watson
Our Modifications
$\xrightarrow{R_{1} \text { versus } R_{0}}$
$R_{0}$ versus $D$
Reminder
Definition
Reminder 2
$D$ Lower Bound
$R_{0}$ Upper Bound

$R_{0}$ versus $D$

## Reminder: Definition (base)

| Introduction |
| :--- |
| Overview of Results |
| Göös-Pitassi-Watson |
| Our Modifications |
| $R_{1}$ versus $R_{0}$ |
| $R_{0}$ versus $D$ |
| Reminder |
| Definition |
| Reminder 2 |
| $D$ Lower Bound |
| $R_{0}$ Upper Bound |
| Summary |



## Accept iff

- There is a (unique) all- 1 column $b$;
- in $b$, there is a unique element $r$ with non-zero pointers;
- for each $j \neq b$, following a path $T(j)$ from $r$ gives a zero in the $j$ th column.
- Some additional information is contained in the leaves (to be defined).


## Definition

## Accept iff



- There is a (unique) all- 1 column $b$;
$\square$ in $b$, there is a unique element $r$ with non-zero pointers;
- for each $j \neq b$, following a path $T(j)$ from $r$ gives a zero in the $j$ th column.
- all the leaves back point to the all-1 column $b$.


## Reminder 2: Adversary Argument

- While there are non-queried elements in a column:
$\square$ Return 1 .
- When the last element in a column is queried:
$\square$ Return 0 , linking it to the last returned 0 .

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | 0 | -1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |  | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

## Deterministic Lower Bound

$R_{1}$ versus $R_{0}$
$\underline{R_{0} \text { versus } D}$
Reminder
Definition
Reminder 2
$D$ Lower Bound
$R_{0}$ Upper Bound
Summary

## Conclusion

Adversary Method.
Let $n=2 m$.
If the $k$ th element is queried in a column:

- If $k \leq m$, return 1 .
- Otherwise, return 0 with back pointer to column $k-m$.


At the end, the column contains $m!1.0$ and $m$ with back pointers to all columns $1,2, \ldots, m$.

## Deterministic Lower Bound

$\qquad$
Overview of Results
$R_{1}$ versus $R_{0}$
$R_{0}$ versus $D$
Reminder
Definition
Reminder 2

Adversary Method.
Let $n=2 m$.
If the $k$ th element is queried in a column:

- If $k \leq m$, return (1).
- Otherwise, return 0 with back pointer to column $k-m$.


At the end, the column contains $m!1.0$ and $m$ with back pointers to all columns $1,2, \ldots, m$.

■ The algorithm does not know the value of the function until it has queried $>m$ elements in each of $m$ columns.

## Deterministic Lower Bound

$\qquad$
Overview of Results
$\underline{R_{1} \text { versus } R_{0}}$
$R_{0}$ versus $D$
Reminder
Definition
Reminder 2

Adversary Method.
Let $n=2 m$.
If the $k$ th element is queried in a column:

- If $k \leq m$, return (1).
- Otherwise, return 0 with back pointer to column $k-m$.


At the end, the column contains $m, 1$ and $m$ with back pointers to all columns $1,2, \ldots, m$.

- The algorithm does not know the value of the function until it has queried $>m$ elements in each of $m$ columns.

Lower bound: $\Omega\left(m^{2}\right)$.

## $R_{0}$ Upper Bound: Informal

$\qquad$
$\underline{R_{1} \text { versus } R_{0}}$
$R_{0}$ versus $D$


- Each column contains a back pointer to the all-1 column. BUT which one is the right one-?


## $R_{0}$ Upper Bound: Informal

| Introduction |
| :--- |
| Overview of Results |
| Göös-Pitassi-Watson |
| Our Modifications |
| $R_{1}$ versus $R_{0}$ |
| $R_{0}$ versus $D$ |
| Reminder |
| Definition |
| Reminder 2 |
| $D$ Lower Bound |
| $R_{0}$ Upper Bound |
| Summary |



- Each column contains a back pointer to the all-1 column. BUT which one is the right one-?

We try each back pointer by quering few elements in the column, and proceed to a one where no zeroes were found.

- Even if this is not the all-1 column, we can arrange that it contains fewer zeroes whp.


## $R_{0}$ Upper Bound: Formal

## Algorithm

- Let $c$ be the first column, and $k \leftarrow n$.
- While $k>1$,
$\square \quad$ Let $c \leftarrow \operatorname{ProcessColumn}(c, k)$, and $k \leftarrow k / 2$.
ProcessColumn(column $c$, integer $k$ )
- Query all elements in column $c$.
- If there are no zeroes, verify column $c$.
- If there are $>k$ zeroes, query all $n m$ variables, and output the value of the function.
- For each zero $a$ :
$\square \quad$ Let $j$ be the back pointer of $a$.
$\square$ Query $\widetilde{O}(n / k)$ elements in column $j$. (Probability $<\frac{1}{(n m)^{2}}$ that no zero found if there are $>k / 2$ of them).
$\square$ If no zero was found, return $j$.
- Reject


## Summary

## Introduction

Overview of Results
Göös-Pitassi-Watson
Our Modifications
$R_{1}$ versus $R_{0}$
$\underline{R_{0} \text { versus } D}$
Reminder
Definition
Reminder 2
$D$ Lower Bound
$R_{0}$ Upper Bound
Summary
Conclusion

Take $n=2 m$.

- Lower bound for a $D$ algorithm is $\Omega\left(m^{2}\right)$.
- Upper bound for a $R_{0}$ algorithm is $O(n+m)$.

We get a quadratic separation between $R_{0}$ and $D$.

## Summary

## Introduction

Overview of Results
Göös-Pitassi-Watson
Our Modifications
$R_{1}$ versus $R_{0}$
$R_{0}$ versus $D$
Reminder
Definition
Reminder 2
$D$ Lower Bound
$R_{0}$ Upper Bound
Summary
Conclusion

Take $n=2 m$.

- Lower bound for a $D$ algorithm is $\Omega\left(m^{2}\right)$.
- Upper bound for a $R_{0}$ algorithm is $O(n+m)$.

We get a quadratic separation between $R_{0}$ and $D$.

- Also, upper bound for a $Q_{2}$ algorithm is $\widetilde{O}(\sqrt{n+m})$.

We get a quartic separation between $Q_{2}$ and $D$.
NB. Previous separation was quadratic: Grover's search.

## Conclusion

Open Problems

$$
\begin{aligned}
R_{1} & =\widetilde{O}\left(R_{0}^{1 / 2}\right) \\
Q_{E} & =\widetilde{O}\left(R_{0}^{1 / 2}\right) \\
R_{0} & =\widetilde{O}\left(D^{1 / 2}\right) \\
Q_{2} & =\widetilde{O}\left(D^{1 / 4}\right) \\
Q_{2} & =\widetilde{O}\left(R_{0}^{1 / 3}\right) \\
Q_{E} & =\widetilde{O}\left(R_{2}^{2 / 3}\right) \\
\widetilde{\operatorname{deg}} & =\widetilde{O}\left(R_{2}^{1 / 4}\right)
\end{aligned}
$$

## Open Problems



We have resolved $R_{2} \leftrightarrow R_{0}$ and $R_{1} \leftrightarrow D$. Can we resolve $R_{2} \leftrightarrow D$ too? Known: $R_{2}=\Omega\left(D^{1 / 3}\right)$ and $R_{2}=\widetilde{O}\left(D^{1 / 2}\right)$.

■ Can we overcome the "certificate complexity barrier"? Obtain a function with $R_{2}=o(C)$ ?

- The same about $Q_{2} \leftrightarrow D$

Known: $Q_{2}=\Omega\left(D^{1 / 6}\right)$ and $Q_{2}=\widetilde{O}\left(D^{1 / 4}\right)$.
$\square$ and $Q_{E} \leftrightarrow D$ ?
Known: $Q_{E}=\Omega\left(D^{1 / 3}\right)$ and $Q_{E}=\widetilde{O}\left(D^{1 / 2}\right)$.

## Cheat Sheets

Aaronson, Ben-David, and Kothari came up with the Cheat-Sheet technique.

## Cheat Sheets

## Introduction

Overview of Results
Göös-Pitassi-Watson
Our Modifications
$\underline{R_{1} \text { versus } R_{0}}$
$\underline{R_{0} \text { versus } D}$
Conclusion
Results
Open Problems
Cheat Sheets

## Aaronson, Ben-David, and Kothari came up with the Cheat-Sheet technique.

- also uses pointers
- is incomparable to our results
- prove a number of interesting results, e.g., a total Boolean function $f$ with

$$
R_{2}(f)=\widetilde{\Omega}\left(Q_{2}(f)^{2.5}\right) .
$$

## Cheat Sheets

Aaronson, Ben-David, and Kothari came up with the Cheat-Sheet technique.
also uses pointers

- is incomparable to our results
- prove a number of interesting results, e.g., a total Boolean function $f$ with

$$
R_{2}(f)=\widetilde{\Omega}\left(Q_{2}(f)^{2.5}\right) .
$$

- Actually, $R_{2}(f)=\widetilde{\Omega}\left(Q_{2}(f)^{3}\right)$, if there exists a partial function $g$ on $n$ variables with

$$
Q_{2}(g)=O(\log n) \quad \text { and } \quad R_{2}(g)=\widetilde{\Omega}(n)
$$

## Any questions?



