Natural Proofs

Yiannis Kokkinis

Structural Complexity $(\mu\Pi\lambda\forall)$

May 17, 2012

Overview

1 Introduction

Overview

Overview

Overview

Overview

Overview

Why bother?

- Why are proofs for lower bounds so difficult? (e.g. $P \neq PSPACE$, $P \neq NP$)

Why bother?

- Why are proofs for lower bounds so difficult? (e.g. $P \neq PSPACE$, $P \neq NP$)
- Answer 1: because great mathematicians cannot prove them...

Why bother?

- Why are proofs for lower bounds so difficult? (e.g. $P \neq PSPACE$, $P \neq NP$)
- Answer 1: because great mathematicians cannot prove them...
- Answer 2: because known techniques are not good enough:

Why bother?

- Why are proofs for lower bounds so difficult? (e.g. $P \neq PSPACE$, $P \neq NP$)
- Answer 1: because great mathematicians cannot prove them...
- Answer 2: because known techniques are not good enough:
    - relativization: oracles, diagonalization (Baker, Gill, Solovay 1975)

Why bother?

- Why are proofs for lower bounds so difficult? (e.g. $P \neq PSPACE$, $P \neq NP$)
- Answer 1: because great mathematicians cannot prove them...
- Answer 2: because known techniques are not good enough:
  - relativization: oracles, diagonalization (Baker, Gill, Solovay 1975)
  - natural proofs: circuit lower bounds (Razborov, Rudich 1994)

Why bother?

- Why are proofs for lower bounds so difficult? (e.g. $P \neq PSPACE$, $P \neq NP$)
- Answer 1: because great mathematicians cannot prove them...
- Answer 2: because known techniques are not good enough:
  - relativization: oracles, diagonalization (Baker, Gill, Solovay 1975)
  - natural proofs: circuit lower bounds (Razborov, Rudich 1994)
  - algebrization: IP = PSPACE, PCP theorems (Aaronson, Wigderson 2008)

Circuit Complexity

- A boolean circuit is a directed acyclic graph. It's nodes are AND, OR and NOT gates

Circuit Complexity

- A boolean circuit is a directed acyclic graph. It's nodes are AND, OR and NOT gates
- The size of the circuit is the number of gates in it

Circuit Complexity

- A boolean circuit is a directed acyclic graph. It's nodes are AND, OR and NOT gates
- The size of the circuit is the number of gates in it
- Different circuit for every different input size

Circuit Complexity

- A boolean circuit is a directed acyclic graph. It's nodes are AND, OR and NOT gates
- The size of the circuit is the number of gates in it
- Different circuit for every different input size
- Let Γ be a complexity class. A family of circuits $C_0, C_1, \ldots$ is said to be Γ-uniform if there is a Γ-bounded TM that on input 1^n outputs C_n

Circuit Complexity

- A boolean circuit is a directed acyclic graph. It's nodes are AND, OR and NOT gates
- The size of the circuit is the number of gates in it
- Different circuit for every different input size
- Let Γ be a complexity class. A family of circuits $C_0, C_1, \ldots$ is said to be Γ-uniform if there is a Γ-bounded TM that on input 1^n outputs C_n

Theorem

A language L is in P iff L has logspace-uniform polynomial circuits

Overview

What is a natural proof?

Definition (n^c-usefulness)

Let $f : \{0,1\}^n \rightarrow \{0,1\}$, $c \in \mathbb{N}$. Any proof that f does not have n^c-sized circuits can be viewed as defining a predicate $\mathcal{P}$ s.t. $\mathcal{P}(f) = 1$ and $\forall g \in SIZE(n^c)\ \mathcal{P}(g) = 0$

What is a natural proof?

Definition (n^c-usefulness)

Let $f : \{0,1\}^n \to \{0,1\}$, $c \in \mathbb{N}$. Any proof that f does not have n^c-sized circuits can be viewed as defining a predicate $\mathcal{P}$ s.t. $\mathcal{P}(f) = 1$ and $\forall g \in SIZE(n^c)\ \mathcal{P}(g) = 0$

Definition (Natural predicate)

We say that a predicate $\mathcal{P}$ is natural if it satisfies the following two conditions ($g : \{0,1\}^n \to \{0,1\}$):

What is a natural proof?

Definition (n^c-usefulness)

Let $f : \{0,1\}^n \to \{0,1\}$, $c \in \mathbb{N}$. Any proof that f does not have n^c-sized circuits can be viewed as defining a predicate $\mathcal{P}$ s.t. $\mathcal{P}(f) = 1$ and $\forall g \in SIZE(n^c)\ \mathcal{P}(g) = 0$

Definition (Natural predicate)

We say that a predicate $\mathcal{P}$ is natural if it satisfies the following two conditions ($g : \{0,1\}^n \to \{0,1\}$):

Constructiveness: We can compute $\mathcal{P}(g)$ in time polynomial to the size of the truth table of g (that is in time $2^{\mathcal{O}(n)}$)

What is a natural proof?

Definition (n^c-usefulness)

Let $f : \{0,1\}^n \to \{0,1\}$, $c \in \mathbb{N}$. Any proof that f does not have n^c-sized circuits can be viewed as defining a predicate $\mathcal{P}$ s.t. $\mathcal{P}(f) = 1$ and $\forall g \in SIZE(n^c)\ \mathcal{P}(g) = 0$

Definition (Natural predicate)

We say that a predicate $\mathcal{P}$ is natural if it satisfies the following two conditions ($g : \{0,1\}^n \to \{0,1\}$):

Constructiveness: We can compute $\mathcal{P}(g)$ in time polynomial to the size of the truth table of g (that is in time $2^{\mathcal{O}(n)}$)

Largeness: $Pr[\mathcal{P}(g) = 1] \geqslant 1/n$

What is a natural proof?

Theorem (Existence of hard functions (Shannon, 1949))

The vast majority of all boolean functions with n inputs requires $\Omega(2^n/n)$ gates

What is a natural proof?

Theorem (Existence of hard functions (Shannon, 1949))

The vast majority of all boolean functions with n inputs requires $\Omega(2^n/n)$ gates

- The above theorem implies that only a small fraction of boolean functions have polynomial size circuits. (but it's proof is not constructive, so we cannot use it to prove that there is a languge in $NP \setminus SIZE(n^c)$)

What is a natural proof?

Theorem (Existence of hard functions (Shannon, 1949))

The vast majority of all boolean functions with n inputs requires $\Omega(2^n/n)$ gates

- The above theorem implies that only a small fraction of boolean functions have polynomial size circuits. (but it's proof is not constructive, so we cannot use it to prove that there is a languge in $NP \setminus SIZE(n^c)$)
- So the largeness condition does not contradict n^c-usefulness.

What is a natural proof?

Theorem (Existence of hard functions (Shannon, 1949))

The vast majority of all boolean functions with n inputs requires $\Omega(2^n/n)$ gates

- The above theorem implies that only a small fraction of boolean functions have polynomial size circuits. (but it's proof is not constructive, so we cannot use it to prove that there is a languge in $NP \setminus SIZE(n^c)$)
- So the largeness condition does not contradict n^c-usefulness.

Definition (Natural Proof)

A proof that a function does not have polynomial size circuits is called natural if it defines a natural predicate that is n^c-useful.

Examples of predicates

1. $\mathcal{P}(g) = 1 \Leftrightarrow g \notin SIZE(n^{\log n})$

Examples of predicates

1. $\mathcal{P}(g) = 1 \Leftrightarrow g \notin SIZE(n^{\log n})$
   Usefulness: $n^c = \mathcal{O}(n^{\log n})$ ✓

Examples of predicates

1. $\mathcal{P}(g) = 1 \Leftrightarrow g \notin SIZE(n^{\log n})$
   Usefulness: $n^c = \mathcal{O}(n^{\log n})$ ✓
   Largeness: existence of hard functions theorem ✓

Examples of predicates

1. $\mathcal{P}(g) = 1 \Leftrightarrow g \notin SIZE(n^{\log n})$

   Usefulness: $n^c = \mathcal{O}(n^{\log n})$ ✓

   Largeness: existence of hard functions theorem ✓

   Constructiveness: it's an open problem (we can check whether $\mathcal{P}(g) = 1$ in time $\mathcal{O}(2^{n^{\log n}})$ by enumerating all circuits of size $n^{\log n}$) ?

Examples of predicates

1. $\mathcal{P}(g) = 1 \Leftrightarrow g \notin SIZE(n^{\log n})$
   Usefulness: $n^c = \mathcal{O}(n^{\log n})$ ✓
   Largeness: existence of hard functions theorem ✓
   Constructiveness: it's an open problem (we can check whether $\mathcal{P}(g) = 1$ in time $\mathcal{O}(2^{n^{\log n}})$ by enumerating all circuits of size $n^{\log n}$) ?
2. $\mathcal{P}(g) = 1 \Leftrightarrow g$ correctly solves the decision problem 3SAT for inputs of size n

Examples of predicates

1. $\mathcal{P}(g) = 1 \Leftrightarrow g \notin SIZE(n^{\log n})$

   Usefulness: $n^c = \mathcal{O}(n^{\log n})$ ✓

   Largeness: existence of hard functions theorem ✓

   Constructiveness: it's an open problem (we can check whether $\mathcal{P}(g) = 1$ in time $\mathcal{O}(2^{n^{\log n}})$ by enumerating all circuits of size $n^{\log n}$) ?

2. $\mathcal{P}(g) = 1 \Leftrightarrow g$ correctly solves the decision problem 3SAT for inputs of size n

   Usefulness: if $3SAT \notin SIZE(n^c)$... ?

Examples of predicates

1. $\mathcal{P}(g) = 1 \Leftrightarrow g \notin SIZE(n^{\log n})$
   Usefulness: $n^c = \mathcal{O}(n^{\log n})$ ✓
   Largeness: existence of hard functions theorem ✓
   Constructiveness: it's an open problem (we can check whether $\mathcal{P}(g) = 1$ in time $\mathcal{O}(2^{n^{\log n}})$ by enumerating all circuits of size $n^{\log n}$) ?
2. $\mathcal{P}(g) = 1 \Leftrightarrow g$ correctly solves the decision problem 3SAT for inputs of size n
   Usefulness: if $3SAT \notin SIZE(n^c)$... ?
   Largeness: it is true only for one function ✗

Examples of predicates

1. $\mathcal{P}(g) = 1 \Leftrightarrow g \notin SIZE(n^{\log n})$

   Usefulness: $n^c = \mathcal{O}(n^{logn})$ ✓

   Largeness: existence of hard functions theorem ✓

   Constructiveness: it's an open problem (we can check whether $\mathcal{P}(g) = 1$ in time $\mathcal{O}(2^{n^{\log n}})$ by enumerating all circuits of size $n^{\log n}$) ?

2. $\mathcal{P}(g) = 1 \Leftrightarrow g$ correctly solves the decision problem 3SAT for inputs of size n

   Usefulness: if $3SAT \notin SIZE(n^c)$... ?

   Largeness: it is true only for one function ✗

   Constructiveness: we can check whether $\mathcal{P}(g) = 1$ by checking g's truth table in time $2^{\mathcal{O}(n)}$ ✓

A natural proof

AC^0: constant depth, polynomial size, unlimited fan-in

A natural proof

AC^0: constant depth, polynomial size, unlimited fan-in

Parity($\{$x : x has an odd number of 1s$\}$) $\notin AC^0$. In this proof the following predicate is defined:
$\mathcal{P}(g) = 1 \Leftrightarrow$ g cannot be made constant by restricting $n - n^{\epsilon}$ input bits

A natural proof

AC^0: constant depth, polynomial size, unlimited fan-in

Parity($\{$x : x has an odd number of 1s$\}$) $\notin AC^0$. In this proof the following predicate is defined:
$\mathcal{P}(g) = 1 \Leftrightarrow$ g cannot be made constant by restricting $n - n^{\epsilon}$ input bits

Usefulness: $\mathcal{P}(g) = 0$ for every AC^0 circuit and $\mathcal{P}(g) = 1$ for the parity function ✓

A natural proof

AC^0: constant depth, polynomial size, unlimited fan-in

Parity($\{$x : x has an odd number of 1s$\}$) $\notin AC^0$. In this proof the following predicate is defined:
$\mathcal{P}(g) = 1 \Leftrightarrow$ g cannot be made constant by restricting $n - n^\epsilon$ input bits

Usefulness: $\mathcal{P}(g) = 0$ for every AC^0 circuit and $\mathcal{P}(g) = 1$ for the parity function ✓

Largeness: if g is a random function then $\mathcal{P}(g) = 1$ with high probability ✓

A natural proof

AC^0: constant depth, polynomial size, unlimited fan-in

Parity($\{$x : x has an odd number of 1s$\}$) $\notin AC^0$. In this proof the following predicate is defined:
$\mathcal{P}(g) = 1 \Leftrightarrow$ g cannot be made constant by restricting $n - n^\epsilon$ input bits

Usefulness: $\mathcal{P}(g) = 0$ for every AC^0 circuit and $\mathcal{P}(g) = 1$ for the parity function ✓

Largeness: if g is a random function then $\mathcal{P}(g) = 1$ with high probability ✓

Constructiveness: we can check in time $2^{\mathcal{O}(n)}$ if $\mathcal{P}(g) = 1$ from g's truth table ✓

Overview

Why did we define natural predicates that way?

1. Why constructiveness?

Why did we define natural predicates that way?

1. Why constructiveness?

Why did we define natural predicates that way?

1. Why constructiveness?
   We are interested in checking our conditions efficiently...

Why did we define natural predicates that way?

1. Why constructiveness?
   We are interested in checking our conditions efficiently...
2. Why largeness?

Why did we define natural predicates that way?

1. Why constructiveness?
   We are interested in checking our conditions efficiently...
2. Why largeness?

Why did we define natural predicates that way?

1. Why constructiveness?
   We are interested in checking our conditions efficiently...
2. Why largeness?

Lemma

If a function f does not have circuits of size $< S$ then at least half of the functions (with the same number of input variables as f) do not have circuits of size $\leqslant S/2 - 3$

Why did we define natural predicates that way?

1. Why constructiveness?
   We are interested in checking our conditions efficiently...
2. Why largeness?

Lemma

If a function f does not have circuits of size < S then at least half of the functions (with the same number of input variables as f) do not have circuits of size $\leqslant$ S/2 -3

Proof.

Let g be a random function then f = $(f \oplus g) \oplus g$. If both g and $f \oplus g$ have circuits of size < S/2 - 3 then f has a circuit of size <S (we need only 5 gates to compute $\oplus$) □

The Natural Proofs Theorem

- So, we can easily check whether a function satisfies a natural predicate or not

The Natural Proofs Theorem

- So, we can easily check whether a function satisfies a natural predicate or not
- That means that we can easily tell whether a function is "random" or not

The Natural Proofs Theorem

- So, we can easily check whether a function satisfies a natural predicate or not
- That means that we can easily tell whether a function is "random" or not
- That is a key fact to prove the

The Natural Proofs Theorem

- So, we can easily check whether a function satisfies a natural predicate or not
- That means that we can easily tell whether a function is "random" or not
- That is a key fact to prove the

Theorem (Natural Proofs, Razborov-Rudich 1994)

Suppose that subexponentially strong one-way functions exist. Then there exists a constant $c \in \mathbb{N}$ such that there is no n^c-useful natural predicate $\mathcal{P}$

The Natural Proofs Theorem

- So, we can easily check whether a function satisfies a natural predicate or not
- That means that we can easily tell whether a function is "random" or not
- That is a key fact to prove the

Theorem (Natural Proofs, Razborov-Rudich 1994)

Suppose that subexponentially strong one-way functions exist. Then there exists a constant $c \in \mathbb{N}$ such that there is no n^c-useful natural predicate $\mathcal{P}$

The Natural Proofs Theorem

- So, we can easily check whether a function satisfies a natural predicate or not
- That means that we can easily tell whether a function is "random" or not
- That is a key fact to prove the

Theorem (Natural Proofs, Razborov-Rudich 1994)

Suppose that subexponentially strong one-way functions exist. Then there exists a constant $c \in \mathbb{N}$ such that there is no n^c-useful natural predicate $\mathcal{P}$

*subexponentially strong one-way function = one that resists inverting even by a $2^{n^{\epsilon}}$ -time adversary for some fixed $\epsilon > 0$.

Pseudorandom functions

- We will show the contrapositive: suppose that for every c there exists a natural predicate, then one way functions do not exist

Pseudorandom functions

- We will show the contrapositive: suppose that for every c there exists a natural predicate, then one way functions do not exist
- J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby in 1999 showed that we can built a pseudorandom function family from every one way function.

Pseudorandom functions

- We will show the contrapositive: suppose that for every c there exists a natural predicate, then one way functions do not exist
- J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby in 1999 showed that we can built a pseudorandom function family from every one way function.

Definition (Pseudorandom function family)

A family of functions $\{f_s\}_{s \in \{0,1\}^*}$, where for $s \in \{0,1\}^m$, f_s is a function from $\{0,1\}^m$ to $\{0,1\}$, s.t.:

Pseudorandom functions

- We will show the contrapositive: suppose that for every c there exists a natural predicate, then one way functions do not exist
- J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby in 1999 showed that we can built a pseudorandom function family from every one way function.

Definition (Pseudorandom function family)

A family of functions $\{f_s\}_{s \in \{0,1\}^*}$, where for $s \in \{0,1\}^m$, f_s is a function from $\{0,1\}^m$ to $\{0,1\}$, s.t.:

1. We can built $f_s(x)$ in time polynomial in s and x

Pseudorandom functions

- We will show the contrapositive: suppose that for every c there exists a natural predicate, then one way functions do not exist
- J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby in 1999 showed that we can built a pseudorandom function family from every one way function.

Definition (Pseudorandom function family)

A family of functions $\{f_s\}_{s \in \{0,1\}^*}$, where for $s \in \{0,1\}^m$, f_s is a function from $\{0,1\}^m$ to $\{0,1\}$, s.t.:

1. We can built $f_s(x)$ in time polynomial in s and x
2. For $s \in \{0,1\}^m$ no polynomial time algorithm can distinguish f_s from a random function from $\{0,1\}^m$ to $\{0,1\}$

Proof of the theorem

- Let $\mathcal{P}$ be a natural property on n-bit functions that is n^c -useful. That means that we have a $2^{\mathcal{O}(n)}$-time algorithm which:

Proof of the theorem

- Let $\mathcal{P}$ be a natural property on n-bit functions that is n^c -useful. That means that we have a $2^{\mathcal{O}(n)}$-time algorithm which:
  1. outputs 0 on functions with circuit complexity lower than n^c

Proof of the theorem

- Let $\mathcal{P}$ be a natural property on n-bit functions that is n^c -useful. That means that we have a $2^{\mathcal{O}(n)}$-time algorithm which:
  1. outputs 0 on functions with circuit complexity lower than n^c
  2. outputs 1 on a nonnegligible fraction of functions

Proof of the theorem

- Let $\mathcal{P}$ be a natural property on n-bit functions that is n^c -useful. That means that we have a $2^{\mathcal{O}(n)}$-time algorithm which:
  1. outputs 0 on functions with circuit complexity lower than n^c
  2. outputs 1 on a nonnegligible fraction of functions
- Let $\{f_s\}$ be a 2^{m^ϵ}-secure pseudorandom function family

Proof of the theorem

- Let $\mathcal{P}$ be a natural property on n-bit functions that is n^c -useful. That means that we have a $2^{\mathcal{O}(n)}$-time algorithm which:
  1. outputs 0 on functions with circuit complexity lower than n^c
  2. outputs 1 on a nonnegligible fraction of functions
- Let $\{f_s\}$ be a 2^{m^ϵ}-secure pseudorandom function family
- Let $h : \{0,1\}^m \rightarrow \{0,1\}$ be an unknown function (it could be either f_s for some s or a random function)

Proof of the theorem

- Let $\mathcal{P}$ be a natural property on n-bit functions that is n^c -useful. That means that we have a $2^{\mathcal{O}(n)}$-time algorithm which:
    1. outputs 0 on functions with circuit complexity lower than n^c
    2. outputs 1 on a nonnegligible fraction of functions
- Let $\{f_s\}$ be a 2^{m^ϵ}-secure pseudorandom function family
- Let $h : \{0,1\}^m \to \{0,1\}$ be an unknown function (it could be either f_s for some s or a random function)
- We will use the natural property $\mathcal{P}$ to tell whether h is a (truly) random function

Proof of the theorem

- Let $n = m^{\epsilon/2}, x \in \{0,1\}^n$ and $g(x) = h(x0^{m-n})$.

Proof of the theorem

- Let $n = m^{\epsilon/2}, x \in \{0,1\}^n$ and $g(x) = h(x0^{m-n})$.
- We construct g's truth table (it costs $2^{\mathcal{O}(n)}$ time)

Proof of the theorem

- Let $n = m^{\epsilon/2}, x \in \{0,1\}^n$ and $g(x) = h(x0^{m-n})$.
- We construct g's truth table (it costs $2^{\mathcal{O}(n)}$ time)
- We calculate $\mathcal{P}(g)$. There are two cases:

Proof of the theorem

- Let $n = m^{\epsilon/2}, x \in \{0,1\}^n$ and $g(x) = h(x0^{m-n})$.
- We construct g's truth table (it costs $2^{\mathcal{O}(n)}$ time)
- We calculate $\mathcal{P}(g)$. There are two cases:
  1. h is a random function, so g is also a random function from $\{0,1\}^n$ to $\{0,1\}$. That is $Pr[\mathcal{P}(g) = 1] \geqslant 1/n$.

Proof of the theorem

- Let $n = m^{\epsilon/2}, x \in \{0,1\}^n$ and $g(x) = h(x0^{m-n})$.
- We construct g's truth table (it costs $2^{\mathcal{O}(n)}$ time)
- We calculate $\mathcal{P}(g)$. There are two cases:
  1. h is a random function, so g is also a random function from $\{0,1\}^n$ to $\{0,1\}$. That is $Pr[\mathcal{P}(g) = 1] \geqslant 1/n$.
  2. h is f_s for some s. The map $s, x \mapsto f_s(x)$ can be computed in poly(m) time and hence the map $x \mapsto g(x)$ is computable by a circuit of size $poly(m) = n^c$ (for some c) that has s hard-wired into it. (To be sure, the distinguisher does not know s or this circuit; we are only asserting that the circuit exists). Hence $\mathcal{P}(g) = 0$.

Proof of the theorem

- Let $n = m^{\epsilon/2}, x \in \{0,1\}^n$ and $g(x) = h(x0^{m-n})$.
- We construct g's truth table (it costs $2^{\mathcal{O}(n)}$ time)
- We calculate $\mathcal{P}(g)$. There are two cases:
  1. h is a random function, so g is also a random function from $\{0,1\}^n$ to $\{0,1\}$. That is $Pr[\mathcal{P}(g) = 1] \geqslant 1/n$.
  2. h is f_s for some s. The map $s, x \mapsto f_s(x)$ can be computed in poly(m) time and hence the map $x \mapsto g(x)$ is computable by a circuit of size $poly(m) = n^c$ (for some c) that has s hard-wired into it. (To be sure, the distinguisher does not know s or this circuit; we are only asserting that the circuit exists). Hence $\mathcal{P}(g) = 0$.
- That means that we can distinguish between f_s and a random function with nonnegligible probability in polynomial time.

Overview

Trying to find a circuit lower bound for 3SAT...

- If a circuit is complicated, some part of it should be complicated too

Trying to find a circuit lower bound for 3SAT...

- If a circuit is complicated, some part of it should be complicated too
- So it is tempting to try to prove circuit lower bounds by induction on a measure defined on the circuit size

Trying to find a circuit lower bound for 3SAT...

- If a circuit is complicated, some part of it should be complicated too
- So it is tempting to try to prove circuit lower bounds by induction on a measure defined on the circuit size

Definition (Formal complexity measure)

A function $\mu : \{\{0,1\}^n \to \{0,1\}\} \to \mathbb{N}^+$ s.t.:

Trying to find a circuit lower bound for 3SAT...

- If a circuit is complicated, some part of it should be complicated too
- So it is tempting to try to prove circuit lower bounds by induction on a measure defined on the circuit size

Definition (Formal complexity measure)

A function $\mu : \{\{0,1\}^n \rightarrow \{0,1\}\} \rightarrow \mathbb{N}^+$ s.t.:

1. $\mu(x) \leqslant 1, \mu(\bar{x}) \leqslant 1$

Trying to find a circuit lower bound for 3SAT...

- If a circuit is complicated, some part of it should be complicated too
- So it is tempting to try to prove circuit lower bounds by induction on a measure defined on the circuit size

Definition (Formal complexity measure)

A function $\mu : \{\{0,1\}^n \to \{0,1\}\} \to \mathbb{N}^+$ s.t.:

1. $\mu(x) \leqslant 1, \mu(\bar{x}) \leqslant 1$
2. $\mu(f \wedge g) \leqslant \mu(f) + \mu(g)$

Trying to find a circuit lower bound for 3SAT...

- If a circuit is complicated, some part of it should be complicated too
- So it is tempting to try to prove circuit lower bounds by induction on a measure defined on the circuit size

Definition (Formal complexity measure)

A function $\mu : \{\{0,1\}^n \rightarrow \{0,1\}\} \rightarrow \mathbb{N}^+$ s.t.:

1. $\mu(x) \leqslant 1, \mu(\bar{x}) \leqslant 1$
2. $\mu(f \wedge g) \leqslant \mu(f) + \mu(g)$
3. $\mu(f \vee g) \leqslant \mu(f) + \mu(g)$

Trying to find a circuit lower bound for 3SAT...

- If a circuit is complicated, some part of it should be complicated too
- So it is tempting to try to prove circuit lower bounds by induction on a measure defined on the circuit size

Definition (Formal complexity measure)

A function $\mu : \{\{0,1\}^n \to \{0,1\}\} \to \mathbb{N}^+$ s.t.:

1. $\mu(x) \leqslant 1, \mu(\bar{x}) \leqslant 1$
2. $\mu(f \wedge g) \leqslant \mu(f) + \mu(g)$
3. $\mu(f \vee g) \leqslant \mu(f) + \mu(g)$

Trying to find a circuit lower bound for 3SAT...

- If a circuit is complicated, some part of it should be complicated too
- So it is tempting to try to prove circuit lower bounds by induction on a measure defined on the circuit size

Definition (Formal complexity measure)

A function $\mu : \{\{0,1\}^n \to \{0,1\}\} \to \mathbb{N}^+$ s.t.:

1. $\mu(x) \leqslant 1, \mu(\bar{x}) \leqslant 1$
2. $\mu(f \wedge g) \leqslant \mu(f) + \mu(g)$
3. $\mu(f \vee g) \leqslant \mu(f) + \mu(g)$

For example μ (f) = 1 + the smallest formula size for f

Trying to find a circuit lower bound for 3SAT...

- If μ is any formal complexity measure, then $\mu(f)$ is a lower bound on the formula complexity of f (proof by induction)

Trying to find a circuit lower bound for 3SAT...

- If μ is any formal complexity measure, then $\mu(f)$ is a lower bound on the formula complexity of f (proof by induction)
- So it suffices to show that $\mu(3SaT)$ is super-polynomial

Trying to find a circuit lower bound for 3SAT...

- If μ is any formal complexity measure, then $\mu(f)$ is a lower bound on the formula complexity of f (proof by induction)
- So it suffices to show that $\mu(3SaT)$ is super-polynomial
- But this property cannot hold only for one function...

Trying to find a circuit lower bound for 3SAT...

- If μ is any formal complexity measure, then $\mu(f)$ is a lower bound on the formula complexity of f (proof by induction)
- So it suffices to show that $\mu(3SaT)$ is super-polynomial
- But this property cannot hold only for one function...

Theorem

Suppose μ is a formal complexity measure and $\mu(f) \geqslant S$ for some f and some large number S. Then $Pr[\mu(g) \geqslant S/4] \geqslant 1/4$.

Trying to find a circuit lower bound for 3SAT...

- If μ is any formal complexity measure, then $\mu(f)$ is a lower bound on the formula complexity of f (proof by induction)
- So it suffices to show that $\mu(3SaT)$ is super-polynomial
- But this property cannot hold only for one function...

Theorem

Suppose μ is a formal complexity measure and $\mu(f) \geqslant S$ for some f and some large number S. Then $Pr[\mu(g) \geqslant S/4] \geqslant 1/4$.

Proof.

For random g let $h = f \oplus g$ so $f = h \oplus g = (h \wedge \bar{g}) \vee (g \wedge \bar{h})$. If $Pr[\mu(g) < S/4] > 3/4$ then $\mu(h), \mu(g), \mu(\overline{g}), \mu(\bar{h}) < S/4$, so $\mu(f) < S$, but that is absurd. □

Trying to find a circuit lower bound for 3SAT...

So if we prove this way that $\mu(3SaT)$ is super-polynomial we define a natural predicate $\mathcal{P}(f) = 1 \Leftrightarrow \mu(f) > n^c$

Trying to find a circuit lower bound for 3SAT...

So if we prove this way that $\mu(3SaT)$ is super-polynomial we define a natural predicate $\mathcal{P}(f) = 1 \Leftrightarrow \mu(f) > n^c$

Usefulness: $\mathcal{P}(3SAT) = 1, \textit{if}\, g \in SIZE(n^c)\ \textit{then}\, \mathcal{P}(g) = 0$ ✓

Trying to find a circuit lower bound for 3SAT...

So if we prove this way that $\mu(3SaT)$ is super-polynomial we define a natural predicate $\mathcal{P}(f) = 1 \Leftrightarrow \mu(f) > n^c$

Usefulness: $\mathcal{P}(3SAT) = 1, \textit{if } g \in SIZE(n^c) \textit{ then } \mathcal{P}(g) = 0$ ✓

Largeness: we just proved it ✓

Trying to find a circuit lower bound for 3SAT...

So if we prove this way that $\mu(3SaT)$ is super-polynomial we define a natural predicate $\mathcal{P}(f) = 1 \Leftrightarrow \mu(f) > n^c$

Usefulness: $\mathcal{P}(3SAT) = 1, if\, g \in SIZE(n^c)\, then\, \mathcal{P}(g) = 0$ ✓

Largeness: we just proved it ✓

Constructiveness: easy from the truth table ✓

Trying to find a circuit lower bound for 3SAT...

So if we prove this way that $\mu(3SaT)$ is super-polynomial we define a natural predicate $\mathcal{P}(f) = 1 \Leftrightarrow \mu(f) > n^c$

Usefulness: $\mathcal{P}(3SAT) = 1, if\, g \in SIZE(n^c)\; then\, \mathcal{P}(g) = 0$ ✓

Largeness: we just proved it ✓

Constructiveness: easy from the truth table ✓

That means that if one way functions exist we cannot prove P $\neq NP$ that way

Overview

Unnatural proofs

- Can we prove circuit lower bounds using unnatural proofs?

Unnatural proofs

- Can we prove circuit lower bounds using unnatural proofs?
- We can use old simple diagonalization!

Unnatural proofs

- Can we prove circuit lower bounds using unnatural proofs?
- We can use old simple diagonalization!
- Diagonalization is an inherently unnatural technique because it focuses on a specific function, so it violates the largeness condition

Unnatural proofs

- Can we prove circuit lower bounds using unnatural proofs?
- We can use old simple diagonalization!
- Diagonalization is an inherently unnatural technique because it focuses on a specific function, so it violates the largeness condition
- Alternatively, one can also view a diagonalization proof as showing that a function has the property that it disagrees with every small circuit on some input - a property that satisfies largeness but not constructiveness.

Unnatural proofs

- Can we prove circuit lower bounds using unnatural proofs?
- We can use old simple diagonalization!
- Diagonalization is an inherently unnatural technique because it focuses on a specific function, so it violates the largeness condition
- Alternatively, one can also view a diagonalization proof as showing that a function has the property that it disagrees with every small circuit on some input - a property that satisfies largeness but not constructiveness.
- But diagonalization is a relativizing proof technique...

Promise problems

- A promise problem is a partially defined function $f : \{0,1\}^* \to \{0,1,\perp\}$

Promise problems

- A promise problem is a partially defined function $f : \{0,1\}^* \rightarrow \{0,1,\perp\}$
- An algorithm A solves a promise problem f iff $\forall x (f(x) \in \{0,1\} \Rightarrow A(x) = f(x))$ ($\perp$ represents undefined so when f(x) = $\perp$ there is no guarantee for A's output)

Promise problems

- A promise problem is a partially defined function $f : \{0,1\}^* \rightarrow \{0,1,\perp\}$
- An algorithm A solves a promise problem f iff $\forall x (f(x) \in \{0,1\} \Rightarrow A(x) = f(x))$ ($\perp$ represents undefined so when f(x) = $\perp$ there is no guarantee for A's output)
- We can define *promise*$\mathcal{C}$ for every complexity class $\mathcal{C}$

Promise problems

- A promise problem is a partially defined function $f : \{0,1\}^* \to \{0,1,\perp\}$
- An algorithm A solves a promise problem f iff $\forall x (f(x) \in \{0,1\} \Rightarrow A(x) = f(x))$ ($\perp$ represents undefined so when f(x) = $\perp$ there is no guarantee for A's output)
- We can define *promise*$\mathcal{C}$ for every complexity class $\mathcal{C}$

Definition (promiseMA)

Let f be a promise problem. f $\in$ promiseMA if for every $x \in \{0,1\}^*$ $\exists$ polynomials p,q and a polynomial time algorithm A s.t.:

$f(x) = 1 \Rightarrow \exists y \in \{0,1\}^{q(|x|)}, \exists z \in \{0,1\}^{p(|x|)} Pr[A(x,y,z) = 1] \geqslant 2/3$

$f(x) = 0 \Rightarrow \exists y \in \{0,1\}^{q(|x|)}, \exists z \in \{0,1\}^{p(|x|)} Pr[A(x,y,z) = 1] \leqslant 1/3$

Unnatural proofs

Theorem

$PSPACE \nsubseteq SIZE(n^c)$

Unnatural proofs

Theorem

$PSPACE \nsubseteq SIZE(n^c)$

Theorem (R. Santhanam, 2007)

$promiseMA \nsubseteq SIZE(n^c)$

Unnatural proofs

Theorem

$PSPACE \nsubseteq SIZE(n^c)$

Theorem (R. Santhanam, 2007)

$promiseMA \nsubseteq SIZE(n^c)$

Unnatural proofs

Theorem

$PSPACE \nsubseteq SIZE(n^c)$

Theorem (R. Santhanam, 2007)

$promiseMA \nsubseteq SIZE(n^c)$

the proof of the first theorem uses diagonalization and the proof of the second theorem uses the first result

Moral

- We showed that we can use a natural property (one that holds for a nonnegligible fraction of boolean functions and can easily be checked) to distinguish a pseudorandom function from a truly random function

Moral

- We showed that we can use a natural property (one that holds for a nonnegligible fraction of boolean functions and can easily be checked) to distinguish a pseudorandom function from a truly random function
- So, we cannot use natural proofs to prove circuit lower bounds in complexity classes where pseudorandom generators exist, like NC^1(parallel log-time and polynomial number of processors) or TC^0 (constant depth, polynomial size, unbounded-fanin)

Moral

- We showed that we can use a natural property (one that holds for a nonnegligible fraction of boolean functions and can easily be checked) to distinguish a pseudorandom function from a truly random function
- So, we cannot use natural proofs to prove circuit lower bounds in complexity classes where pseudorandom generators exist, like NC^1(parallel log-time and polynomial number of processors) or TC^0 (constant depth, polynomial size, unbounded-fanin)
- It is interesting that we used computational complexity to shed light on a metamathematical question about computational complexity

Moral

- We showed that we can use a natural property (one that holds for a nonnegligible fraction of boolean functions and can easily be checked) to distinguish a pseudorandom function from a truly random function
- So, we cannot use natural proofs to prove circuit lower bounds in complexity classes where pseudorandom generators exist, like NC^1(parallel log-time and polynomial number of processors) or TC^0 (constant depth, polynomial size, unbounded-fanin)
- It is interesting that we used computational complexity to shed light on a metamathematical question about computational complexity
- But remember that we used a condition (existence of one-way functions) that is stronger than $P \neq NP$...

Bios

Bios

Alexander Razborov (1963-) 
Nevanlinna Prize (Approximation method, 1990), Gödel Prize (Natural Proofs, 2007)

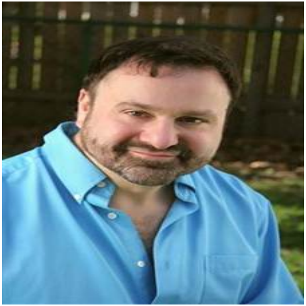
University of Chicago

Bios

Alexander Razborov (1963-) 
Nevanlinna Prize (Approximation method, 1990), Gödel Prize (Natural Proofs, 2007)

University of Chicago

Steven Rudich (1961-)
Gödel Prize (Natural Proofs, 2007)



Carnegie Mellon University

References

A.Razborov, S. Rudich. Natural Proofs, J. Comput. Syst. Sci. 55(1): 24-35 (1997)

References

- **A.Razborov, S. Rudich**. Natural Proofs, J. Comput. Syst. Sci. 55(1): 24-35 (1997)
- **Timothy Y. Chow**. What is a natural proof?, Notices of the AMS Volume 58, Number 11, December 2011

References

- **A.Razborov, S. Rudich**. Natural Proofs, J. Comput. Syst. Sci. 55(1): 24-35 (1997)
- **Timothy Y. Chow**. What is a natural proof?, Notices of the AMS Volume 58, Number 11, December 2011
- **Sanjeev Arora, Boaz Barak**. Computational Complexity: A modern Approach, Cambridge Univeristy Press 2009

References

- **A.Razborov, S. Rudich**. Natural Proofs, J. Comput. Syst. Sci. 55(1): 24-35 (1997)
- **Timothy Y. Chow**. What is a natural proof?, Notices of the AMS Volume 58, Number 11, December 2011
- **Sanjeev Arora, Boaz Barak**. Computational Complexity: A modern Approach, Cambridge Univeristy Press 2009

References

- **A.Razborov, S. Rudich**. Natural Proofs, J. Comput. Syst. Sci. 55(1): 24-35 (1997)
- **Timothy Y. Chow**. What is a natural proof?, Notices of the AMS Volume 58, Number 11, December 2011
- **Sanjeev Arora, Boaz Barak**. Computational Complexity: A modern Approach, Cambridge Univeristy Press 2009

THANK YOU!