Natural Proofs

Alexander N. Rybalov
Omsk State University
Omsk, Russia

January 12, 2007

The P versus NP problem

P – all binary languages recognized in polynomial time.

NP – all binary languages certified in polynomial time.

Is $P = NP$?

Approaches to P vs NP

Diagonalization is from Algorithm Theory (1960-1970s). A weakness of Diagonalization:

$$C_1 \neq C_2 \Rightarrow C_1^O \neq C_2^O$$

for any oracle O. Every such proof can be relativized.

Baker, Gill and Solovay in 1975 constructed oracles A and B such that

$$P^A = NP^A \text{ and } P^B \neq NP^B.$$

Approaches to P vs NP

Combinatorial approach uses Boolean functions and sophisticated combinatorics (1980s).

Razborov and Rudich in 1994 introduced the notion of Natural Proof to explain why this approach cannot separate P and NP.

A.Razborov, S.Rudich. Natural Proofs, Journal of Computer and System Sciences, Vol. 55, No 1, 1997, pages 24-35.

Boolean Circuits

Boolean circuit C of variables $x_1, ..., x_n$ is a finite sequence of assignments of type

$$y_i = u_j \circ u_k, \ \circ \in \{\vee, \wedge\}$$

or

$$y_i = \neg u_j,$$

where u_j, u_k is either some input variable x_j, or some previous intermediate variable $y_j, j < i$.

Circuit C computes a Boolean function

$$f_C : \{0,1\}^n \to \{0,1\}.$$

The size of C is the number of assignments.

Boolean Circuits and P vs NP

A layer f_n of a function

$$f : \{0,1\}^* \to \{0,1\}$$

is the restriction of f on $\{0,1\}^n$.

Theorem (Savage). *If function*

$$f : \{0,1\}^* \to \{0,1\}$$

is computable by a Turing machine within time $t(n)$, then there is a constant C such that every layer f_n is computed by a Boolean circuit of size less than $Ct^2(n)$.

Corolary. *If there is a set in NP with characteristic function f such that every f_n cannot be computed by Boolean circuits of size less than $s(n)$ with some super-polynomial function s, then $P \neq NP$.*

Lower bound problem is to find such a set and prove lower bound on circuit size.

Almost all Boolean Functions are Hard

Theorem (Claude Shannon). *For any constant* $1 < C < 2$ *almost all Boolean function*

$$f : \{0,1\}^n \to \{0,1\}$$

cannot be computed by Boolean circuits of size less than C^n.

$C(n, N)$ — the number of Boolean circuits of length less than N from n variables. The number of different assignments

$$y_i = u_j * u_k, * \in \{\vee, \wedge\},$$

or

$$y_i = \neg u_j,$$

where u_j, u_k are either $x_l, 1 \leq l \leq n$ or $y_l, l < i$, is less than $3(N+n)^2$.

Almost all Boolean Functions are Hard

So $C(n, N)$ is less than the number of all possible sets of N such assignments, i.e.

$$C(n, N) < (3(n + N)^2)^N <$$

$$< (12N^2)^N = 2^{2Nlog(12N)}$$

for $N > n$. Now let $N = C^n$, where $1 < C < 2$ and

$$C(n, C^n) = 2^{2C^n log(12C^n)} = 2^{2nC^n log(12C)}.$$

The number of all boolean functions of n variables F_n is 2^{2^n}, so

$$\frac{C(n, C^n)}{F_n} \to 0.$$

Known Lower Bounds

The best known now lower bound for problems in NP is only $O(n)$:(

Lower bounds of restricted Boolean circuits are much better:

- Razborov in 1985 proved a super-polynomial lower bound for monotone circuits (without negation),

- Ajtai in 1983 and Furst, Saxe and Sipser in 1984 proved exponential lower bound for circuits with restriction on depth.

Usual Strategy of Lower Bound Proving

- To find some complexity measure of Boolean functions.

- To prove by induction on construction of circuit that small (polynomial-sized) circuits can only compute functions with "low" (polynomial bounded) complexity.

- To show that layer functions of characteristic function of some problem (from NP) have "hight" (exponential) complexity.

A Toy Example

Parity function

$$p_n(x_1, \ldots, x_n) = x_1 \oplus \ldots \oplus x_n$$

where $\oplus$ — addition modulo 2.

Complexity measure of a Boolean function f is the number of variables such that f depends on these variables substantially.

f depends on x_i substantially if there are $a_j \in \{0, 1\}, j \neq i$ such that

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \neq$$

$$\neq f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

A Toy Example

1. Any Boolean circuit of size less than n cannot compute a function which substantially depends on n variables.

2. Parity function p_n substantially depends on n variables.

So we have a linear lower bound on circuit size of parity.

Natural Proofs

- Constructivity: Checking that a function has high complexity can be done on the truth table in polynomial time (in $2^{O(n)}$ time, where n – number of variables).

- Largeness: $\frac{C_n}{F_n} \geq \frac{1}{2^{O(n)}}$, where C_n – the set of all "complicated" function of n variables, F_n – set of all boolean functions of n variables. In the other words "complicated" functions don't form a strongly negligible set with respect to the size of truth table.

The Toy Example is Natural

1. To check that a function substantially depends on x_i we can test the equation

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) =$$

$$= f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

for all $a_j \in \{0, 1\}, j \neq i$. And do this for every variable. Of course this algorithm is exponential of n, but it is polynomial of the size of truth table.

2. The number of functions which substantially does not depend on x_i is not greater than $2^{2^{n-1}}$ and therefore the number of functions which substantially depends on some variables is not less than

$$2^{2^n} - n2^{2^{n-1}}.$$

A Weakness of Natural Proofs

Theorem (Razborov, Rudich). *If there exists a natural proof of exponential lower bound for some NP-problem, then there is no strong pseudorandom function generator. In particular, there exists a polynomial algorithm for factoring and discrete logarithm.*

Pseudorandom Function Generators

Pseudorandom function generator is a polynomially computable function

$$f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$$

with the following property. There is a constant $c > 0$ such that for any sufficiently large n for any Turing machine M running in time $2^{O(n)}$

$$|Pr(M(f_n) = 1) - Pr(M(f_n(k,\cdot)) = 1)| < \frac{1}{2^{n^2}}$$

if a "key" $k \in \{0,1\}^{n^c}$ is chosen randomly and uniformly and f is chosen randomly and uniformly from the set of all boolean functions of n variables.

Pseudorandom Function Generators

Informally, no one polynomial Turing machine on truth tables of functions f and $f(k,\cdot)$ (that machine running in time $2^{O(n)}$) can separate with a good probability truly random function f from pseudorandom function $f(k,\cdot)$.

Examples of such pseudorandom function generators were constructed in 1980s by M.Blum, O.Goldreich, S.Goldwasser and S.Micali assuming the existing of one-way functions with a complexity 2^{n^c} of invertion for some $c > 0$. It is widely believed that discrete logarithm and factorization problems can provide such one-way functions.

A Sketch of Proof of Razborov's and Rudich's Theorem

Suppose a Turing machine M (polynomial in size of truth table, i.e. in time $2^{O(n)}$) decides the set of all complicated functions C. Note that for any fixed key $k \in \{0,1\}^{n^c}$ for any sufficiently large n the layers of $f(k,\cdot)$ can be computed by polynomial-sized circuits . It implies that for any sufficiently large n and for any $k \in \{0,1\}^{n^c}$ functions $f_n(k,\cdot) \notin C$. So $M(f_n(k,\cdot)) = 0$ and $Pr(M(f_n(k,\cdot)) = 1) = 0$. But $M(f_n) = 1$ for every function from C and it follows from the largeness property that

$$Pr(M(f_n) = 1) \geq Pr(f \in C_n) \geq \frac{1}{2^{O(n)}}.$$

Hence

$$|Pr(M(f_n) = 1) - Pr(M(f_n(k,\cdot)) = 1)| \geq \frac{1}{2^{O(n)}}$$

— a contradiction.

How natural is largeness?

Formula is a Boolean circuit such that $j \neq k$ in any its assignment $y_i = y_j * y_k$, $* \in \{\vee, \wedge\}$.

$\mu : F_n \to \mathbb{N}$ is a formal complexity measure if

- $\mu(f) \leq 1$ if $f(\bar{x}) = x_i$ or $f(\bar{x}) = \neg x_i$,
- $\mu(f * g) \leq \mu(f) + \mu(g)$, where $* \in \{\vee, \wedge\}$.

Lemma. *$\mu(f)$ is a lower bound on formula size for f.*

How natural is largeness?

Theorem. *If $\mu(f) = T$ for some function of n variables then at least for $\frac{1}{4}$ fraction of all boolean functions g of n variables $\mu(g) \geq \frac{T}{4}$.*

Let g be arbitrary boolean function of n variables and $h = g \oplus f$ then

$$f = h \oplus g = (\neg h \wedge g) \vee (h \wedge \neg g)$$

and

$$\mu(f) \leq \mu(h) + \mu(g) + \mu(\neg h) + \mu(\neg g).$$

If set $\{g : \mu(g) < \frac{T}{4}\}$ contains more than $\frac{3}{4}$ of all functions then we can choose g such that

$$\mu(g), \mu(\neg g), \mu(h), \mu(\neg h) < \frac{T}{4}.$$

Hence $\mu(f) < T$, a contradiction.