Polynomial complexity classes over real algebras with nilpotent elements

Alexander N. Rybalov, Omsk Branch IM SB RAS

доказал теорему ферма тчк
икс в степени эн переносим
в правую часть тчк
подробности письмом тчк

Телеграмма в адрес Академии Наук

Generalized Computability and Complexity

- Blum, Shub and Smale, 1989: computability over rings (BSS-model).
- Ashaev, Belyaev and Myasnikov, 1992: computability over the list superstructure (ABM-model).
- Hemmerling, 1996: computability and complexity over algebraic structures (based on BSS-model).
- Rybalov, 2002: complexity over the list superstructure (based on ABM-model).

BSS-model (Hemmerling version)

Computational model — a generalization of Turing machine to a ring $\langle R, +, \times, 1, 0\rangle$. BSS-machine consists of:

- an finite tape, every cell of the tape contains an element from R,
- a finite number of pointers p_i on cells of the tape,
- a program consisting of finite number of numerated commands

BSS-program

- $right(p_i)$ $(left(p_i))$ — to move pointer p_i to the right (left) cell,
- $p_i = p_j \circ p_k$ $(\circ \in \{+, \times\})$ — to write in cell p_i the sum or the product of cells p_j and p_k,
- $p_i = 0$, $p_i = 1$ — to write a constant in cell p_i,
- $stop$ — the halting command,

- $if\ p_i = p_j\ goto\ q$ — if cells p_i and p_j contain the same element then go to command q, else to the next command,

- $lapp(p_i)$ — to append a cell at left from p_i (if p_i points on the most left cell),

- $rapp(p_i)$ — to append a cell at right from p_i (if p_i points on the most right cell),

- $del(p_i)$ — to delete cell p_i (if p_i points on one of end cells), after that p_j (and all other pointers pointing on this cell) will point on adjacent cell.

BSS-model

BSS-machine M computes some function

$$f_M : R^* \to R^*$$

in the following way. The input string w of elements from R is written on the starting tape. After start numerated commands of M are performed one-by-one (goto command may change the order) until the stop command. After halting the string $f_M(w)$ is written on the tape. If M is not halting then $f_M(w)$ is not defined.

Having this definition we can develop a computability and complexity theory over R.

The size of input w is just the length $|w|$.

Some Features

- If ring R is binary field $\langle 0, 1, +, \times, / \rangle$ then we have the classical Turing computability over binary strings.

- Example of recursive set over field $\mathbb{C}$:

$$(a_1, \ldots, a_n) : \exists I \subseteq \{1, \ldots, n\} \sum_{i \in I} a_i = 0.$$

- Examples of not-recursive sets over field $\mathbb{C}$: integers $\mathbb{Z}$, Mandelbrot and Julia fractals (Blum, Shub, Smale).

NP-complete problems

Satisfability problem over ring R:

$$(f_1(\bar{x}), \ldots, f_n(\bar{x})) : \exists \bar{a} \in R^*$$

$$f_1(\bar{a}) = 0 \wedge \ldots \wedge f_n(\bar{a}) = 0$$

is NP-complete (Blum, Shub, Smale).

List superstructure

Introduced by Goncharov and Sviridenko. $\langle HL(A), \sigma^* \rangle$— list superstructure of structure $\langle A, \sigma \rangle$. Here $HL(A)$ is

$$L_0 = A, L_{n+1} = L_n \cup F(L_n)$$

$$HL(A) = \cup_{n=0}^{\infty} L_n,$$

where $F(B)$ is the set of all finite lists over B.

$$\sigma^* = \sigma \cup \{head^{(1)}, tail^{(1)}, cons^{(2)}, nil\}$$

- $tail(\langle a_1, a_2, \ldots, a_n \rangle) = \langle a_2, \ldots, a_n \rangle, \qquad head(\langle a_1, a_2, \ldots, a_n \rangle) = a_1$
- $cons(\langle a_1, a_2, \ldots, a_n \rangle, b) = \langle a_1, a_2, \ldots, a_n, b \rangle, \qquad nil = \langle \rangle$

ABM-model

Machine M has a finite number of registers $R_1, \ldots, R_n$, in which elements of $HL(A)$ are stored. Program of machine consists of commands of the types:

- $R_i = R_j$
- $R_i = c$, where c is a constant from σ^*
- $R_i = f(R_{i_1}, \ldots, R_{i_k})$, where f is a function from σ^*
- $if\ P(R_{i_1}, \ldots, R_{i_k})\ goto\ q$, where P is a predicate from σ or equality

ABM-model

The first register R_1 contains initial data. The commands are executed in a natural way. After halting R_1 contains the result. So machine M computes a function

$$f_M : HL(A) \to HL(A).$$

Theories of computability and complexity were developed in these frameworks. The size of input is the size of list defined as

$$size(a) = 1,\ a \in A,$$
$$size(\langle a_1, \ldots, a_k \rangle) = \sum_{i=1}^{k} size(a_i).$$

Some Features

- For functions $f : A^* \to A^*$ ABM-model is equivalent to BSS-model (Rybalov).

- Interesting types of sets (recursive, halting, output) have a natural description in so-called logic of computable disjunctions (Ashaev, Belyaev, Myasnikov).

- A theory of NP-completeness was developed (Rybalov).

Polynomial Classes over Structures

$\mathfrak{A} = \langle A, \sigma \rangle$ — some structure.

$P_{\mathfrak{A}}$ — class of subsets of A^*, recognized in polynomial time by **deterministic** BSS-machines.

$DNP_{\mathfrak{A}}$ — class of subsets of A^*, recognized in polynomial time by BSS-machines with **nondeterministic branches**.

$$if\ ?\ goto\ q$$

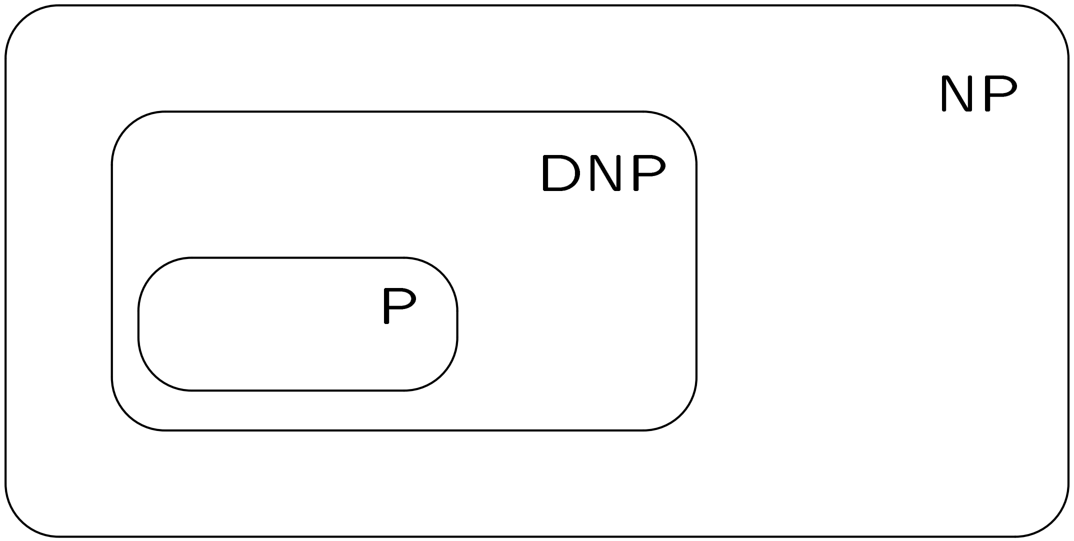
$NP_{\mathfrak{A}}$ — class of subsets of A^*, recognized in polynomial time by BSS-machines with **nondeterministic guesses**.

$$p_i = guess$$

P versus NP

Lemma. $P_{\mathfrak{A}} \subseteq DNP_{\mathfrak{A}} \subseteq NP_{\mathfrak{A}}$

Question. *Is* $P_{\mathfrak{A}} = DNP_{\mathfrak{A}}$*? Is* $DNP_{\mathfrak{A}} = NP_{\mathfrak{A}}$*?*



P versus NP over Some Structures

- $DNP = NP$ over any finite structure and $PvsNP$ is equivalent to classical $PvsNP$.

- $P \neq DNP$ over $\langle \mathbb{R}, + \rangle$ (Meer, 1992).

- $P \neq DNP$ over $\langle \mathbb{R}, +, \leq \rangle \Leftrightarrow P \neq NP$ in classics (Koiran, 1996).

- $P \neq DNP$ over infinite abelian groups (Gassner, 2002).

- $DNP \neq NP$ over of integers $\langle \mathbb{Z}, +, -, \times, 0, 1 \rangle$ (Hemmerling, 1995).

P versus NP over Some Structures

- $P \neq DNP$ over infinite Boolean algebras (Prunescu, 2003).
- $P \neq DNP$ over real and complex matrix rings (Rybalov, 2004).
- $DNP \neq NP$ over unordered field $\mathbb{R}$ (BSS + Cucker 199?).
- $DNP \neq NP$ over field $\mathbb{Q}$ (Malajovich, 199?).
- Hemmerling in 2005 constructed a structure where $P = NP$.

P versus NP over $\mathbb{R}$ and $\mathbb{C}$

Question. *Is* $P \neq DNP$ *and* $DNP \neq NP$ *over* $\langle \mathbb{C}, +, -, \times, /, 0, 1 \rangle$*?*

Question. *Is* $P \neq DNP$ *and* $DNP \neq NP$ *over* $\langle \mathbb{R}, +, -, \times, /, \leq, 0, 1 \rangle$*?*

- If $BPP = P$ then classical $P = NP$ implies $P = NP$ over $\mathbb{R}$ (BSS, 199?)

- Oracles: $P^{\mathbb{Z}} \neq DNP^{\mathbb{Z}}$ over $\mathbb{C}$ (Rybalov,2004)

$P \neq DNP$ over $\langle \mathbb{R}, + \rangle$

Theorem. $P \neq DNP$ *over* $\langle \mathbb{R}, + \rangle$

We prove that the following set from DNP

$$NULLSACK = \{(a_1, \ldots, a_n) : \exists I \subseteq \{1, \ldots, n\} \sum_{i \in I} a_i = 0\}$$

does not belong to P. Suppose there is a BSS-machine M, recognized $NULLSACK$ with polynomial time bound $p(n)$. Let's try to cheat M.

How to cheat polynomial machines?

- Fix a size n such that $2^n - 1 > p(n)$
- Put $\alpha = (a_1, \dots, a_n)$ to M with a_i linearly independent over $\mathbb{Z}$
- $\alpha \notin NULLSACK$ and M outputs NO
- In computation on α M has $N \leq p(n) < 2^n - 1$ tests of type

$$l_i(a_1, \dots, a_n) = 0, i = 1, \dots, N \quad (*)$$

where l is a linear combination with integer coefficients. All non-trivial tests give inequations because a_i are independent over $\mathbb{Z}$.

How to cheat polynomial machines?

- Now put to M input $\beta = (b_1, \ldots, b_n)$ such that $\beta \in NULLSACK$ but for all non-trivial tests in (*) $l_i(\beta) \neq 0$.

- It's possible because $N < 2^n - 1$ planes

$$l_i(x_1, \ldots, x_n) = 0, i = 1, \ldots, N$$

cannot cover $2^n - 1$ planes of NULLSACK

$$\sum_{i \in I} x_i = 0, I \subseteq \{1, \ldots, n\}$$

So M on β has the same computational path as on α and outputs NO!

$P \neq DNP$ over algebras with nilpotent elements

Theorem. *$P \neq DNP$ over $\mathcal{A}$, where $\mathcal{A}$ is a real algebra with nilpotent elements.*

Theorem. *$P \neq DNP$ over $\mathcal{A}$, where $\mathcal{A}$ is an algebra over field of characteristics 0 with nilpotent elements.*

$P \neq DNP$ over ring $\langle \mathbb{R}, +, -, \times, 0, 1 \rangle$

A problem with similar scheme of proof for ring $\mathbb{R}$: surfaces

$$f_i(x_1, \ldots, x_n) = 0, i = 1, \ldots, N < 2^n - 1$$

with polynomials f_i can cover $NULLSACK$. Actually one surface

$$F(\bar{x}) = \prod_{I \subseteq \{1,\ldots,n\}} (\sum_{i \in I} x_i) = 0$$

covers $NULLSACK$.

But can a polynomial machine get such "big"polynomial F in its computation?

Algebraic Circuits

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

$$y_i = u_j \circ u_k, \ \circ \in \{+, -, \times\},$$

where u_j, u_k is either some input variable x_j, or some previous intermediate variable $y_j, j < i$, or constant 1.

Circuit C computes a polynomial of variables $x_1, ..., x_n$ with integer coefficients. The size $\tau(C)$ of C is the number of assignments.

$$\tau(f) = \min_C \{\tau(C) : \text{C computes f}\}$$

An Example

A polynomial of very big power can be computed by a small circuit. Consider a sequence of polynomials $f_n(x) = x^{2^n}$. It is easy to see that it is computed by the following circuit

$$y_1 = x \cdot x, y_2 = y_1 \cdot y_1, \ldots, y_n = y_{n-1} \cdot y_{n-1}.$$

The size of this circuit n is logarithmic of the power of polynomial x^{2^n}. Moreover it is a well-known fact that any polynomial x^n can be computed by a circuit of size $O(logn)$ - corresponding algorithm is used for encoding and decoding in RSA.

Algebraic Circuits and P vs NP over ring $\mathbb{R}$

Shub-Smale tau conjecture: There exists a constant $C > 0$ such that any polynomial with integer coefficients $f(x)$

$$\tau(f) > Int(f)^C$$

where $Int(f)$ is the number of different integer roots of $f(x)$.

Theorem (BBS + Cucker). *If Shub-Smale tau conjecture is true then $P \neq DNP$ over $\mathbb{R}$.*

Algebraic Circuits and P vs NP over ring $\mathbb{R}$

Suppose some polynomial BSS-machine M decides $NULLSACK$. Then M can computes in polynomial time

$$F_n(\bar{x}) = \prod_{I \subseteq \{1,\ldots,n\}} \Big(\sum_{i \in I} x_i \Big).$$

Then

$$f_n(x) = F_{n+1}(x, 1, 2, 2^2, \ldots, 2^{n-1}) =$$

$$= (2^n - 1)! \prod_{i=1}^{2^n - 1} (x + i)$$

can be computed by polynomial sized circuits.

Algebraic Circuits and Factorization

Moreover if polynomials

$$f_n(x) = (x+1)(x+2)\dots(x+n)$$

can be computed by circuit of size $O(logn)$ for all n (that is a contradiction to Shub-Smale tau conjecture), then there exists a polynomial-time algorithm for integer factorization.